## Predicting the Optimal Computing Platform for Climate-Driven Ecological Forecasting Models

#### Scott Sherwin Farley

#### Master’s Thesis

#### Advisor: John W. Williams

### Introduction

Human-induced global environmental change, including climate warming, anthropogenic land use, and the spread of invasive species, threatens to severely alter biodiversity patterns worldwide in the coming century (Lowe et al., 2011; Root & MacMynowski, 2005; Thuiller, 2007; Thuiller et al., 2008a). Habitat degradation and fragmentation, expansion of invasive species, and a loss of climatically suitable areas are expected to result in large-scale biotic reorganizations, including the extinction of over one-quarter of all species (Thomas, 2010). Climate exerts significant control over species distributions, particularly over those of vascular plants (Salisbury, 1926; Woodward, 1987), implying that anthropogenic perturbations to atmospheric CO2 concentrations can have significant impact on species ranges (Lowe et al., 2011). Statistical methods that quantify species responses to climatic gradients, known as species distribution models (SDMs), can then be used to forecast future biotic assemblages under different warming scenarios (Clark et al., 2014; Guisan & Thuiller, 2005a; Guisan & Zimmerman, 2000; Guisan et al., 2013; Maguire et al., 2015; Wilfried Thuiller et al., 2008a). A growing volume of ecological data, both contemporary and fossil, is available to inform these models, leading to improved forecasting and mitigation of potentially irreversible changes. Environmental monitoring efforts, such as the Long Term Ecological Research program (LTER), (Hobbie et al., 2003), the National Ecological Observatory Network (NEON)(Schimel et al., 2009), community curated databases, including the Neotoma Paleoecological Database (http://neotomadb.org) and the Paleobiology Database (PBDB, http://paleobiodb.org), and modern biodiversity occurrence databases, such as the Global Biodiversity Information Facility (GBIF, http://www.gbif.org), organize, store, and distribute large amounts of information to researchers attempting understand and forecast responses to perturbations in the earth system (Brewer et al., 2012; Michener & Jones, 2012). However, though collections of modern and historical biodiversity data have the potential for forecasting studies, their volume and heterogeneity can make their uptake challenging for ecologists (Hampton et al., 2013). At the same time, cloud computing may offer a technological solution to some of the problems posed by the increasing ‘Bigness’ of ecological data (Hampton et al., 2013; Michener & Jones, 2012), allowing users to easily provision and use configurable and on-demand computer resources. The present study develops a method for identifying the optimal computer hardware on which to run SDMs. The method is highly skillful and can be used to inform model users in computing resource provisioning strategies, as well as model developers in future priorities.

With over 700 million occurrence records in Neotoma and GBIF, traditional statistical methods for analyzing and forecasting ecological processes often cannot be applied without compromising analysis scope. Many SDM methods, though popular in the literature and highly skillful, are not designed to take advantage of parallel processing or distributed computing, and are therefore not scalable to huge datasets. Other scientific fields, including bioinformatics (Schatz et al., 2010), genomics (Stein, 2010), climate analytics (Schnase et al., 2014), as well as private industry (Mosco, 2014), have adopted techniques to cope with large datasets. As the volume of ecological data increases and the need for high resolution, accurate projections of biotic distributions becomes more pressing, reducing project scope (e.g., Bolker et al., 2009a) can no longer be considered a valid option. However, cloud computing may offer a technological solution to some of these problems (Hampton et al., 2013; Michener & Jones, 2012), by providing “ubiquitous, convenient, and on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort” (Mell & Grance, 2012). With the rapid commercialization of cloud computing and the widespread availability of public cloud providers like Amazon Web Services (AWS) and the Google Cloud Compute Engine (GCE), scientists now have a seemingly unlimited supply of computing resources at their disposal.

Although the cloud offers a promising technological solution to the computational demands of ecological forecasting, there are few guiding principles on when, if ever, the benefits in reduced computing time outweigh the financial costs of a cloud-based solution. The cloud introduces a novel expense model for computing by charging users for the use of virtual machines (VMs) rather than the purchase of physical hardware (Hassan, 2011), letting consumers can scale their resources depending on computational demand (Armbrust et al., 2009). While the exact costs of migrating to a cloud environment are difficult to estimate (but see Sun & Li, 2013), the computational time gains achieved by running models on high performance virtual instances can be significant.

I hypothesize that, for any SDM modeling experiment, there exists an optimal data-hardware configuration that maximizes SDM accuracy while jointly minimizing the time and cost of modeling. In this thesis, I develop a framework for predicting this optimal. I gather a large empirical dataset on the cost, runtime, and accuracy of four highly popular of SDMs under different parameterizations and on different computing hardware. Subsequently, a Bayesian learning model was used to predict the execution time of future modeling scenarios and identify the optimal hardware for the task. My findings suggest that if SDMs are to benefit from cloud computing, future effort should be directed towards developing models that more explicitly take advantage of parallelism and distributed computing frameworks.

## Background and Previous Work

#### Big Data in Ecology

The vast expansion of scientific data has necessitated the development of new techniques for data management, analysis, and accessibility (Schaeffer et al., 2008). Worldwide data volume doubled nine times between 2006 and 2011, and successive doubling has continued into this decade (Chen et al., 2014). With the contemporary influx of massive geonomic sequences, long term monitoring projects, phylogenetic histories, and biodiversity occurrence data, robust, expressive and quantitative methods are essential to the future of the biological sciences (Schaeffer et al., 2008). Significant challenges, including inability to move datasets across networks, increased metadata requirements for storage and data discovery, and the need to support novel uses for the data, are common (Schnase et al., 2014).

Ecological occurrence data are records of presence, absence, or abundance of individuals of a species, clade or higher taxonomic grouping that are fundamental to biodiversity analyses, ecological hypothesis testing, and global change research. These data are increasingly being stored in large, dedicated databases like Neotoma, GBIF, PBDB. Since the early 1990s, the internet and associated IT and an increased willingness to share primary data between scientists precipitated rapid influxes of digital occurrence records. While there are known problems with the quality and consistency of data records in large occurrence databases (Soberón et al., 2002), they provide a low-friction way of consuming large amounts of data that would otherwise be prohibitively time consuming to derive from the literature or in the field (Beck et al., 2014; Grimm et al., 2013). Entire new fields, namely ‘Biodiversity Informatics’ (Soberon & Peterson, 2004), ‘Ecoinformatics’ (Michener & Jones, 2012), and ‘Paleoecoinformatics’ (Brewer et al., 2012) have been developed and delineated to address the growing challenges and opportunities presented by the management, exploration, analysis and interpretation of primary data regarding life, particularly at the species level, now housed in biodiversity databases (Soberon & Peterson, 2004).

The term Big Data is typically used to describe very large datasets, whose volume is often accompanied by lack of structure and a need for real-time analysis. Big Data, while posing significant management and analysis challenges, can provide new insights into difficult problems (Chen et al., 2014). Though the precise definition of Big Data is loose, there are two prominent frameworks for discriminating Big Data from traditional data. One characterizes Big Data as “data sets so large and complex that they become awkward to work with using standard statistical software” (Snijders et al., 2012). This ambiguous delineation is echoed in the advertising and marketing literature that accompanies products, like cloud computing, that facilitate Big Data analysis. For example, Apache Hadoop, a popular distributed computing framework, describes Big Data as “datasets which could not be captured, managed, and processed by general computers within an acceptable scope” (Chen et al., 2014).

Under this framework, the Bigness of the data is specific to both the time of analysis and the entity attempting to analyze it. Manyika et al. (2015) suggest that the volume of data required to be Big can change over time, and may grow with time or as technology advances. Furthermore, the criteria for what constitutes Big Data can vary between problem domains (Chen et al., 2014), the size of datasets common in a particular field, and the kinds of software tools that are commonly used in that industry (Manyika et al., 2015). The Big Data label is most often applied to datasets between several terabytes and several petabytes (240 to 250 bytes). Because of ecology’s traditional lack of experience with massive datasets and limited analysis software in the discipline, ecological occurrence data may be characterized as Big Data.

The recent development of complex relational databases that store spatiotemporal occurrence records and their metadata suggests that traditional methods of data handling were not sufficient for modern ecological analyses. While the datasets are not particularly large in storage volume, they are composed of millions of heterogenous records with complex linkages. Further developments, like application programming interfaces and language specific bindings, supplement the tasks of accessing, filtering and working with the large occurrence datasets (Goring et al., 2015; “Package paleobioDB,” 2016). While occurrence data does not require the disk space of popular commercial applications, it has recently demonstrated a need for new, custom built tools to store, analyze, and use.

A second important framework by which to assess Big Data is the ‘Four V’s Framework’. First introduced by IBM and used by large technological companies in the early 2000’s to characterize their data, it is now a popular and flexible framework under which to describe Big Data. Under this framework, a dataset’s Bigness is described by its Volume, Variety, Veracity, and Velocity. Yang & Huang (2013) describe this framework, suggesting that “volume refers to the size of the data; velocity indicates that big data are sensitive to time, variety means big data comprise various types of data with complicated relationships, and veracity indicates the trustworthiness of the data” (Yang & Huang, 2013 p. 276.).

Since the late 1990s, the scale of biodiversity information has become challenging to manage. Figures 1a and 1b track the growth in collections of Neotoma and GBIF through time. In 1990, only 2 of the records now stored in Neotoma were in digitized collections. Today, there are over 14,000 datasets containing over 18 million individual occurrence records, and associated spatial, temporal, and taxonomic metadata, corresponding to an average growth rate of 1,825 occurrences per day. Nearly all records in Neotoma are derived from sedimentary coring or macrofossil extraction efforts, data gathering techniques that require large expenditures of time and effort (Davis, 1963; Glew et al., 2002). GBIF houses digital records of well over 600 million observations, recorded specimens (both fossil and living), and occurrences described in the scientific literature. Since its conception in the early 2000s, the facility’s holdings have grown nearly 300%, from about 180 million records in 2001 to nearly 700 million records in 2016. GBIF’s efforts to digitize existing specimens allow its holdings to precede its origin in 2001.

Biodiversity data is highly diverse with many complicated interrelationships, indicative of its variety. As shown in Figure 2a, Neotoma’s holdings feature 23 dataset categories, including X-ray fluorescence (XRF) and isotopic measurements, macro fossils of both vertebrates and plants, modern and fossil pollen records, and freshwater diatom and water chemistry series. Similarly, in GBIF, there are 9 distinct record types, including human observations, living and fossil specimens, literature review, and machine measurements (Figure 2b). Though the records coexist in the databases, they are distinctly different, derived using different protocols by different communities of researchers. Moreover, the data’s spatial and temporal nature causes complex interrelationships between data entities. All of Neotoma’s records and 87.6% of GBIF’s records are georeferenced[[1]](#footnote-1). Digital representations of spatial phenomena must grapple with data that is a discrete representation of a continuous physical feature, correlations between parameters, space and time, and processes differences at heterogenous spatiotemporal scales (Yang et al., 2011), making storage and management difficult. Finally, occurrence data represents the work of many dispersed individual researchers and research teams. The controlled vocabularies and organizational structures of aggregating databases helps to efficiently assimilate large numbers of records, however, nearly every record was collected, analyzed, and published by a different scientist. While some scientists have contributed many datasets to occurrence databases, most have only contributed one or two. The median number of datasets contributed to Neotoma is 2 and the third quantile value is just 7 datasets. Each researcher is likely to use different equipment, employ different lab procedures, and utilize different documentation practices, contributing to a high variation between datasets.

Biodiversity data also has high levels of uncertainty associated with it. Some of the sources of uncertainty in the data, including spatial or temporal positional uncertainty can be estimated (Wing et al., 2005) or modeled (Blaauw, 2010). Both Neotoma and GBIF show high levels of quantifiable uncertainty. Of a random sample of 10,000 records of the genus *Picea* from GBIF, over half did not report spatial coordinate uncertainty. Of the 4,519 records that did, the average spatial uncertainty was 305 meters, and the maximum was 1,970 meters. Such uncertainty may be problematic for modeling studies (Beck et al., 2014). Temporal uncertainty is high as well. Out of a sample of 32,341 age controls in the database, only 5,722 reported any age uncertainty at all. The remaining records indicate a mean uncertainty of ± 130 years of the actual date. Other sources of uncertainty have yet to be quantified, such as inter-researcher identification differences, measurement errors, and data loss incurred between field, lab, and database. Expert elicitation was recently used to quantify differences between known events in a pollen record. High variability was seen between researchers, suggesting that “expert knowledge of the region and the late-Holocene history of the site” is an important factor (Dawson et al., 2016). Moreover, some contextual information is undoubtedly lost in the aggregation process in the dataset. Though some procedural information accompanies the data records to the database, process details are difficult to coerce into metadata fields, and probably more importantly, contextual details are often lost.

The final piece of the Big Data framework is the dataset’s velocity, which characterizes the dataset’s sensitivity to time. High velocity data must be analyzed in real time as a stream to produce meaningful insights. Tweets, for example, are analyzed for trends as they are posted. Users are drawn to participation in up-to-the minute discussion, and significant effort has been put towards the development of sophisticated algorithms that can detect clusters and trends in real time (Bifet et al., 2011; Kogan, 2014). Unlike in many private sector applications, there is little incentive to researchers to immediately analyze new biodiversity records. Moreover, automated analyses of distributional data have been warned against, due to the overall poor data quality (Soberón et al., 2002) and high uncertainty.

While not time sensitive, ecological occurrence data requires advanced, sophisticated techniques to store and analyze, and demonstrates high volume, low veracity, and significant variety, and should therefore be considered Big Data. Traditional statistical analysis techniques and storage methods for occurrence data may begin to suffer because they were not designed to handle Big Data. Both GBIF and Neotoma are experiencing sustained and increasing growth that has not diminished since the early 1990s. To fully and accurately derive value from new data being added to these databases, novel techniques for modeling and analyzing this data are required.

## Cloud Computing in the Sciences

In recent years, large technology companies have promoted cloud computing as a way of overcoming the computational challenges associated with Big Data. The cloud model leverages distributed networks of physical machines to create a computing utility. The cloud model provides a pay-as-you-go business model and large economies of scale (Armbrust, 2009; Hassan, 2011), that delivers abstract resources and services, in addition to storage and compute resources (Foster et al., 2008). While some organizations and universities have developed ‘private clouds’ -- large collections of virtualized servers not made available to the general public, similar to computing grids -- many researchers have recognized the potential for incorporating public clouds -- utility computing provided as a service by a cloud provider -- into their workflows. With this technology, scientists with little or no computational infrastructure can access scalable and cost-effective computational resources (Hsu et al., 2013). Major scientific organizations in the United States, including the NSF and NASA, have actively promoted cloud computing. Spurred by the U.S. Office of Management and Budget’s 2010 “25 Point Plan to Reform Federal Information Technology Management” (Kundra, 2010), federal agencies are now required to adopt a “Cloud First” policy when “contemplating IT purchases and evaluate secure, reliable, and cost-effective cloud computing alternatives when making new IT investments” (“NASA’s Progress in adopting cloud-computing technologies,” 2013). The federal plan also created programs to help agencies adopt cloud solutions, reducing the effort needed to screen cloud providers for data security policies and enable rapid procurement of cloud services (Kundra, 2010). In 2013, the NSF announced a $20 million dollar solicitation for supporting “research infrastructure that enables the academic research community to develop and experiment with novel cloud architectures addressing emerging challenges, including real-time and high-confidence systems”[[2]](#footnote-2). Public cloud providers, such as Amazon Web Services and Google Cloud Compute, support scientific enterprise by providing large, open-access datasets for public consumption, including Landsat images, real-time NEXRAD radar, and the 1000 Geonomes project, and by soliciting grants for researchers incorporating cloud computing into their research. Amazon Web Service’s website claims that many prominent research institutions, including the NASA Jet Propulsion Laboratory, rely heavily on their products and services[[3]](#footnote-3).

Cloud technology, both public and private, has been extensively lauded for its application in many fields, including bioinformatics (Hsu et al., 2013, Issa et al. (2013); Stein et al., 2007) and climate analytics (Lu et al., 2011; Schnase, 2015; Schnase et al., 2014). Cloud based solutions for bioinformatics research relieve the large memory requirements often present in geonomics and drug-design data (Hsu et al., 2013), and have resulted in low latency, streaming methods for data analysis (Issa et al., 2013) and biology-specific operating systems for protein analysis (Kaján et al., 2013). Contemporary climate analytics requires massive datasets too large to be transferred across a network. Schnase et al. (2014) describes the development of Climate Analytics as a Service, an effort to integrate data storage and high performance computing to perform data-proximal analytics (Schnase, 2015; Schnase et al., 2014).

Cloud services have also been used in the geosciences, and in ecological modeling problems specifically. Yang et al. (2011) suggest that geoscientific problems are still limited by computational ability and argue that the cloud provides a means of overcoming these challenges by leveraging distributed computational resources without increasing the carbon footprint or financial budget of research (Yang et al., 2011). In practice, cloud-optimized implementations of numerical models, such as real-time dust storm forecasting, have improved model performance significantly (Yang et al., 2011). Environmental models can also be run in the cloud (Granell et al., 2013). Candela et al. (2013) describe a novel platform that enables cloud-based SDM workflows, suggesting that an integrated cloud-based approach can aid in data discovery and increase processing capabilities.

#### Species Distribution Models

Species Distribution Models (SDMs) are a widely used class of statistical models that quantify the relationships between a species and its environmental range determinants (Svenning et al., 2011). While these models sometimes include mechanistic or process components, they most often refer to correlative models (Elith & Leathwick, 2009a), using supervised statistical learning algorithms to approximate the functional relationship between species occurrence and environmental covariates. Used extensively in both academic and management contexts, SDMs have been shown to provide reliable estimates of climate-driven range shifts when compared to independent datasets (Guisan & Zimmerman, 2000; Guisan et al., 2006). Widespread availability of statistical software and large databases of environmental and occurrence data, have increased the uptake of these techniques substantially in recent years (Franklin, 2010; Svenning et al., 2011). Citations for SDMs outpaced the field average (National Science Board, 2016) by 3.8% per year between 204 and 2014, according to a Web of Science search (Figure 3).

SDMs use a learning algorithm, along with occurrence records and environmental covariates, to approximate the functional form of the species niche that can be used to test ecological hypotheses or to predict to future scenarios. Hutchinson (1957) characterized a species’ fundamental niche as an n-dimensional hypervolume that defines the environmental spaces where the intrinsic population growth rate of the species is positive (Williams & Jackson, 2007). The realized niche describes the subset of environmental space that the species actually occupies at some point in time, and is smaller than the fundamental niche due to competing biotic interactions with other species. Most scholars argue that SDMs approximate the species’ realized niche (Guisan & Zimmerman, 2000; Miller et al., 2007; Soberón & Peterson, 2005), though the inclusion of fossil data in the model fitting process can increase the likelihood that calibration captures the fundamental niche by exposing the model to states of the climate system not present on Earth today (Veloz et al., 2012).

SDMs rely on three important assumptions. First, as a fundamental justification for applying predictions across space and time, SDMs assume niche conservatism -- that the niche of the species remains constant across all spaces and times (Pearman et al., 2008). Though niche evolution is not accounted for in the models, Peterson et al. (1999) suggest that species typically demonstrate niche conservatism on multi-million year time scales. Second, SDMs rely on the assumption that species are at equilibrium with their environment (Nogués-Bravo, 2009), being present in all environmentally suitable areas while being absent from all unsuitable ones. Given dispersal limitations and biotic interactions between species, this may rarely be the case. For example, Svenning et al. (2008) showed that many European tree species are still limited by postglacial migrational lag. Finally, SDMs must account for extrapolation to novel and no-analog climates for which there is no modern or fossil data. Inductive learning is severely impacted when it is used to predict onto future cases not within the range of values provided in the training set. Williams & Jackson (2007) note the high likelihood of encountering novel and no-analog climates in the near future. Fitting the models with fossil data increases the likelihood that climatic assemblages will be included in the training data, however, given rapid and highly uncertain climate change, the problem of projecting models onto unseen climates is a major limitation on their application.

Despite strong assumptions, SDMs have been used for a variety of paleo and contemporary studies of geographic and environmental distribution. SDMs are often used to confirm ecological hypotheses by comparing hindcast projections with the fossil record; for example, supporting hypotheses on the extinction of Eurasian megafauna (Nogués-Bravo et al., 2008), identifying late-Pleistocene glacial refugia (Fløjgaard et al., 2009; Keppel et al., 2011; Waltari et al., 2007), and to assess the effect of post-glacial distributional limitations and biodiversity changes (Svenning et al., 2008). SDMs are sometimes combined with genetic, phylogeographic, and other methods to develop a complete assessment of a species biogeographical history (e.g., Fritz et al., 2013). In an anthropogenic climate change context, SDMs have been used to assess the effectiveness of modern reserve planning (Araújo et al., 2004), predict the distribution of both endangered (Thuiller et al., 2005) and invasive species (Ficetola et al., 2007; Václavík & Meentemeyer, 2009), and ecosystems (Hamann & Wang, 2006), and evaluate the effectiveness of conservation planning for the future (Loiselle et al., 2003).

#### A Taxonomy of Species Distribution Models

SDMs range in complexity from simple algorithms that characterize a ‘climate envelope’ for a species (Guisan & Zimmerman, 2000) to multivariate bayesian techniques that use Markov Chain Monte Carlo simulations (MCMC) to develop probability distributions around projections and parameters. While all aim to model responses to climatic gradients, (Franklin, 2010) suggests grouping modeling algorithms into data-driven and model-driven algorithms. The data-driven/model-driven dichotomy is introduced in Hastie et al. (2009) and is often employed when differentiating between ‘statistical’ (model-driven) and ‘machine learning’ (data-driven) algorithms. I add the burgeoning set of methods that employ stochastic, probability-based Bayesian methods to this taxonomy due to their recent uptake, high accuracy, and explicit modeling of uncertainty. No individual method or class of methods has consistently outperformed any other (Araújo & New, 2007; Elith et al., 2006; Veloz et al., 2012), though many scholars have attempted to assess variation between models (Araújo & Guisan, 2006; Elith et al., 2006) and parameterizations (Araújo & New, 2007; Thuiller et al., 2008a; Veloz et al., 2012).

Supervised learning techniques use a set of training examples, , where both *x* and *y* are known, to approximate the real relationship between the two, with a function, . The learned approximation minimizes a loss function based on the difference between the real and predicted value, . Each training example is composed of a -dimensional vector of covariates, , . Models can either make *a priori* assumptions about the form of the input-output relationship (model-driven) or adapt to fit any given matrix of training examples (data-driven).

Model-driven learners fit parametric statistical models to a dataset, making assumptions about how inputs and outputs are related, including linearity and error distribution. While these techniques can make poor predictions if the assumptions are not upheld, they were the first to see substantial use in SDM applications and continue to be widely used because of their strong statistical foundations and ability to realistically model ecological relationships (Austin, 2002). These models include boxcar algorithms, which build multidimensional bounding boxes around species presence in environmental space (Guisan & Zimmerman, 2000), as well as more complex methods such as generalized linear models (Guisan et al, 2002; Vincent & Haworth, 1983), and linear and logistic regression for abundance and presence-absence outputs (Franklin, 2010).

The increase in available computing power has spurred the development and application of non-parametric, data-driven learning algorithms. While they do not rely on any stringent assumptions about the form of the relationship, any particular portion of parameter space depends on only a handful on input points, making the models highly sensitive to small changes in the input data (Hastie et al., 2009). In some cases, these models been shown to out perform their model-driven counterparts (Elith et al., 2006), and include genetic algorithms (Elith et al., 2006), classification and regression trees (Elith, Leathwick, & Hastie, 2008), artificial neural networks (Hastie et al., 2009), support vector machines (Drake et al., 2006), and maximum entropy techniques (Elith et al., 2010; Phillips & Dudík, 2008). MaxEnt, a maximum entropy algorithm for SDM and associated Java-based runtime environment, is widely used and has demonstrated its ability to perform consistently even on small sample sizes (Elith et al., 2010; Phillips & Dudík, 2008; Phillips et al., 2006). SDM literature analysis suggest that MaxEnt is the most popular SDM method in use today, and appearing in over 20% of all SDM studies published after 2008. Recent critiques of MaxEnt, however, suggest that its performance may be questionable when compared with other SDM algorithms (Fitzpatrick et al., 2013). Data-driven models are often more computationally intensive than their model-driven counterparts because they usually take at least two passes over the input dataset to process the data and build the model. Furthermore, data-driven learners are often combined with techniques like bagging -- building a collection of models based on random subsets of the input data -- and boosting -- combining many weakly predictive models into a single, highly predictive ensemble – which can further increase computational intensity.

Bayesian methods have also been used to develop the relationship between environmental predictors and species presence. Advantages of the Bayesian approach include the ability to include prior ecological knowledge in model formulation (Ellison, 2004) and the ability to estimate model uncertainty without the need for bootstrapping procedures (Dormann et al., 2012; Elith & Leathwick, 2009b). With improved computational infrastructure and better MCMC sampling algorithms, Bayesian methods have become increasingly popular in recent years (Hegel et al., 2010). Golding & Purse (2016) introduce SDMs that incorporate Gaussian processes, which they claim demonstrate both high predictive accuracy and ecologically sound predictions. Clark et al., (2014) use the joint probability distribution of all taxa in an ecosystem to model both the climatic range limitations of a species and its biotic interactions with other species. Though it can be challenging for ecologists trained in the frequentist perspective to transition to a Bayesian approach (Ellison, 2004; Hegel et al., 2010), software packages are in development for implementing Bayesian models in languages like R (e.g., Vieilledent et al., 2012). MCMC methods are computationally very expensive, though numerical approximations and analytical solutions can sometime reduce computational burden (Golding & Purse, 2016).

A review of recent literature suggests that the majority of SDM users employ data-driven models. Of 100 randomly sampled papers from recent publications in Web of Science that met the query “*(Species Distribution Model) OR (Ecological Niche Model) OR (Habitat Suitability Model)*”, the overwhelming majority utilized data-driven models. Of 203 modeling runs described, 38 were model-driven, 131 were data-driven, and 1 was Bayesian (Figure 4). 33 additional experiments used unsupervised clustering analyses not suitable for prediction. Of all algorithms, MaxEnt was the most popular (64 experiments). Models in the model-driven category included generalized linear models (15), logistic regression (5) and multiple linear regression (2). Data-driven modeling applications included boosted regression trees (16), generalized additive models (11), genetic algorithms (11), random forests (8), artificial neural nets (6), and multivariate adaptive regression splines (4). Figure 4 shows the results of the literature meta analysis.

Because of the overwhelming propensity of scholars to employ methods in the second category, I focus my analyses on this class of algorithms. Many authors have alluded to the limitations imposed by computational complexity, though few have estimated or tested those limits explicitly. Elith et al. (2006) recorded the execution time of the runs they used in their often-cited review of novel SDM techniques, noting execution times of up to several weeks for some modeling algorithms (Elith et al., 2006). Popular learning models, including boosted regression trees (80 h), generalized additive models (17h), generalized linear models (17h), and MaxEnt (2.75 h), were all shown to be extremely computationally intensive. The authors suggest that performance could be improved if model building was split over multiple processing cores. While processor speeds have increased since their 2006 analysis, models are still often unable to leverage multiple processors.

Methodological papers often advise against large modeling studies due to computational limitations. A 2009 review suggests that, when fitting a generalized linear mixed model (GLMM), if a user encounters insufficient computer memory or time limitations, the user should reduce model complexity, perhaps using a subset of the original dataset (Bolker et al., 2009b). Many authors warn of the computational expense of running SDMs, for example, noting that “considerable computational capacity is necessary for the development of models even for a single species” (Peterson, 2003). Thuiller et al. (2008a) cautions “limits to the broad application of this approach may be posed … by the computational challenges encountered in the statistical fitting of complex models.” While modern computing infrastructure may alleviate some of this problem, the computation intensity of SDMs can cause challenges that are difficult to resolve without reducing model complexity.

#### Algorithm Execution Time: Drivers and Measurement

It is possible to theoretically estimate the upper, lower, and average run times using asymptotic complexity analysis. In this exercise, the order of growth of an algorithm’s runtime is determined as its input is increased to infinity, so that only first order terms are relevant (Knuth, 1976). The algorithm that is more efficient asymptotically will typically be the best choice for all but very small inputs (Cormen, 2009). An estimate of the slowest run time (Big-O), can usually be obtained by inspecting the structure of the algorithm and counting how many operations are required when the inputs is sufficiently large (Cormen, 2009). The theoretical complexity of an algorithm is most often considered when considering an algorithm’s scalability (Goldsmith et al., 2007), though the actual runtime will vary with real-world inputs (Cormen, 2009; Goldsmith et al., 2007).

Empirical complexity studies have attempted to bridge the gap between asymptotic theory and real programs. These studies use observations of algorithm runtime under different parameterizations and inputs to build models that predict the run time of future model runs, seeking a method “with the generality of a big-O bound by measuring and statistically modelling [sic] the performance … across many workloads” (Goldsmith et al., 2007). Brewer & Brewer (1995) describes an initial attempt to develop a statistical model for the run and compile time of algorithms in a C library. While most contemporary empirical runtime models use data-driven pattern recognition, linear regression between input size and execution time has been shown to perform well in some cases (Fink, 1998). Empirical complexity models have recently become an important subfield of artificial intelligence and have important applications to algorithm selection (Hutter et al., 2014a). Algorithms for solving very difficult (-Hard) combinatorial problems, can exhibit high runtime variance between different problem instances. Empirical runtime modeling can be used to select the model that will most efficiently reach a solution (Hutter et al., 2014a; Leyton-Brown et al., 2003). Hutter et al. (2014a) outline a comprehensive analysis of strategies and methods for empirical runtime models in the context of algorithm portfolio optimization. Parameterized algorithms can be treated the same way as nonparametric algorithms, by including model parameters as input features in the execution time model (Hutter et al., 2014a). Nonlinear, tree based methods for empirical performance modeling, including random forests, were shown to be superior to other methods because of their ability to group similar inputs together and fit local responses, so that some large outliers do not interfere with the predictions of other groups (Hutter et al., 2014a; Hutter et al., 2014b).

Concurrently running programs, operating system tasks, and other processes may affect the execution time of a real computer program at any point in time. Changes in dynamic system state are stochastic and can cause unpredictable, non-linear and non-additive changes in program runtime (Jones & Kalibera, 2013; Lilja, 2009). Random variation in system state makes deterministic statistical modeling of hardware’s influence on execution time difficult. These variations are a result of the way in which memory access patterns differ in space and time when small changes are made to the operating system state, timing device, or algorithm and its inputs (Lilja, 2009), and few attempts have been made to model them explicitly. Jones & Kalibera (2013) suggest that models based on benchmarked runtime may provide an accurate estimate of an upper bound of execution time, though due to potentially large, nondeterminstic, system-induced variance in empirical results, it is important to perform the benchmarking experiment many times. Dongarra et al., (1987) suggest that a failure to properly characterize the workload, running benchmarks that are too simplistic, or running benchmarks in inconsistent environments can lead to meaningless results.

#### Theoretical Problem Formulation

To conceptualize the optimal configuration for an SDM modeling workflow, it is useful to conceptualize the modeling process using the following framework. Here, the workflow is presented as a series of steps that advance a user towards her goal of obtaining scientific insight from an SDM model. As a rational consumer (Simon, 1986), the modeler will wish to minimize her costs, in both time and money, while maintaining the maximum accuracy under given budgetary constraints.

1. Consider a pool of computing resources, , that is characterized by its hardware capabilities, such as memory and CPUs.
2. Consumers of computing services are part of a market driven by supply and demand, and face costs set by computing providers that are proportional to the computing power provided: Figure 5 demonstrates an instance of a Google Cloud Computing Engine’s cost surface as a function of memory and CPUs.
3. Every user of a modeling application has a particular set of goals for using it in the first place (Norman, 1984). We can conceptualize, for a given model, a finite set of use cases for that model that fall within the bounds of existing or expected use (Carroll, 1999; Rosson, 2002). Let be a vector of characteristics that fully describe the user’s goals in the scenario. The components of include user traits, such as experience with the model and interface, motivation, skill, and desired accuracy, as well as the number and parameterizations of each modeling run required by the modeler.
4. Assume the time to compute a given algorithm with a given set of inputs is proportional to , the effective computing power. In addition to computing the model, however, the user must also undertake a number of other pre- and post-processing steps. The total time elapsed during a modeling experiment can be expressed as

In this formulation, represents the portion of time that is spent by user gathering the resources needed to model, such as time needed to find and download occurrence points and covariates. is a function of the computing resources available to the user (how fast can data be downloaded?) and the experiment (what is the data?). represents time required by the modeler to prepare the data for entry into an algorithm, including data cleaning, projection, and conversion, and can vary widely between modelers, data source and quality, and user skill and motivation. Elith et al., 2006 notes the potential impact of user experience on the modeling time and results. represents time to transfer the output from the computation to the user, which may be non-trivial if the model is run on a set of remote resources and the network downloads are included. Finally, represents the amount of time spent by the user evaluating model output and determining whether her goals were met during the modeling process. Like , this term may be highly variable between model users and applications.

1. Single experiments can be combined together to form workflows, so that a user’s time-to-goal for a workflow of modeling experiments can be expressed as a function of the experiments and the computing resources on which they are run.
2. Combining equations from (2) and (5), the total time for a modeling experiment is the sum of total time of spent modeling and the total monetary cost is the cost of provisioning computing resources for this time. The total workflow cost is then a function of the user and their required modeling experiments, the computing resources, and the cost surface that dictates the cost of these resources. Therefore, a multivariate cost function for all potential user activities is:
3. Each user-based scenario, , will have its own cost curve that’s subject to both the particular characteristics of the workflow and the cost surface imposed by the computing provider. If we select a single use case from the finite set of all possible ’s, and call it , we can obtain a unique cost function for this workflow that depends only on the computing resources used to fit the model. The minimum along this curve corresponds to the optimal hardware for use in this modeling scenario.
4. Multiple experiments may meet the user’s goals, but have different optimal costs. The optimal workflow for a user to pursue is that which jointly maximizes model accuracy while minimizing the cost of provisioning the computing resources for the time required. A set of weights could be applied to preferentially weight one or more of these dimensions. Moreover, if a user faces constraints on time (e.g., latency requirements) or money (e.g., budget requirements), these can be incorporated to find the optimal configuration within the allowable space.

### Methods

#### Data Collection

#### SDM Data Preparation

I systematically collected data on the execution time and accuracy of four SDM algorithms that have shown competitive accuracy results in the literature: multivariate adaptive regression splines (MARS, Leathwick et al., 2006), gradient boosted regression trees (GBM-BRT, Elith et al., 2008; Friedman, 2001; Natekin, 2013), generalized additive models (GAM, Guisan et al., 2002; Yee & Mitchell, 1991), and Random Forests (Breiman, 2006; Elith & Graham, 2009). All of the SDMs were executed using the R statistical environment (R Core Team, 2016) using implementations in the standard packages for SDM. GBM-BRT tree models were fit using the dismo package version 1.1-1 (Hijmans et al., 2016), GAMs using the gam package, version 1.12 (Hastie, 2015), MARS using the earth package version 4.4.4 (Hastie, 2016), and random forests with the randomForest package (Liaw & Wiener, 2002).

Each SDM was fit using fossil pollen data obtained from the Neotoma Paleoecological Database in April 2016. All records for the genera *Picea* (spruce), *Quercus* (oak), *Tsuga* (Hemlock), and *Betula* (birch) were downloaded using the neotoma R package (Goring et al., 2015), and filtered to only include those dated within the last 22,000 years and located in North America. For each record, the latitude, longitude, age, and relative abundance of the taxon was retained and stored using a comma separated value format.

Climatic covariates were obtained from downscaled and debiased Community Climate System Version 3 (CCSM3) model simulations for North America (Lorenz et al., 2016a). The post-processed model output was obtained in NetCDF format with a 0.5-degree spatial resolution and decadal temporal resolution for the last 22,000 years. Bioclimatic variables (BV) (O’Donnell & Ignizio, 2012) were calculated for each timestep using the biovars function in the dismo R package (Hijmans et al. 2016). BV values were extracted for the space-time location of each fossil occurrence. The dataset was then filtered to include only the six least correlated BV covariates, using the variance inflation factor (VIF, O’brien, 2007). The variables retained were BV2 (mean diurnal temperature range), BV7 (annual temperature range), BV8 (mean temperature of westtest quarter), BV15 (precipitation of warmest quarter), BV17 (precipitation of warmest quarter), and BV18 (precipitation of driest quarter).

Future climate layers for AD2100 were obtained from the CMIP project, HadCM3 climate model. These layers represent modeled climate variables under the UN IPCC RCP 8.5, a scenario that assumes high population, moderate economic growth, and a sustained dependence on fossil fuels (Riahi et al., 2011). These layers were processed as above.

#### Computing Infrastructure

The Google Cloud Compute Enginge (GCE) infrastructure was used for all model runs. Google’s platform was chosen over other public cloud vendors because of its ability to create ‘custom’ instance types that adhear to user-defined specifications. Other vendors (e.g., Amazon Web Services) provide a larger number of predefined instance types, but do not support the creation of an instance with an arbitrary hardware configuration. Debian Linux virtual machines were provisioned and released as needed. Git was used to transfer required files to computing nodes. Experimental results were stored in a MySQL relational database for further analysis.

#### SDM Model Protocol

For a virtual machine of a given hardware configuration, data parameters (number of covariates and number of training examples) were communicated to the computing node via a configuration file. Pre-processed occurrence points were partitioned into a testing set (20%) and a training set (80%) of the number of rows (training examples) and columns (covariates). The SDM model was fit using the training data, projected onto the modeled future climate, and assessed for accuracy. SDM accuracy was evaluated using the testing set and quantified using the AUC statistic. Learning (hyper)parameters were initially evaluated as a potential predictor, but were discarded as uninformative. Timing was done within R using the proc.time function. No database I/O was done inside of the timing script, so network connection speed is not expected in influence the results.

Various data and hardware configurations were considered. In total, 26,730 runs were made. Configurations were chosen to maximize the parameter space covered in the analysis while maintaining at least three replicates per configuration. At least three replicates of each data-hardware configuration was made to ensure robust results, though where feasible (see “Assumptions and Limitations”), more replicates were made.

#### Modeling Performance and Accuracy

Once data collection had been completed, SDM runtime and accuracy were modeled using Bayesian additive regression trees (BART), fit with the bartMachine R package, version 1.2.3 (Bleich & Kapelner, 2016). By employing a Bayesian context, the response at each leaf node is a probability distribution, rather than the single maximum likelihood estimate. For more details about the model and its implementation, see Bleich & Kapelner (2016). Runtime and accuracy were modeled separately for each SDM. Runtime was modeled on a log scale, which has been shown to improve prediction high-variance datasets {Hutter:2014ci}. The response data was randomly split into a training set (80%) and testing set (20%). The training set was fit using the default priors suggested by the package authors. 1250 MCMC iterations were performed, each of which built an entire additive model of 50 trees. The first 250 iterations were discarded as burn-in, leaving 1000 posterior samples.

#### Model Evaluation

The predictive skill of the each model was evaluated using the mean squared error (MSE), the statistic between observed and predicted values from the mean of the posterior distribution, and the standard deviation of the distribution. Visual assessment of model results was done by plotting the predicted values against the observed data and qualitatively assessing deviations from the y=x line.

Strength of model drivers was evaluated by cross validating the performance of a separately built model using four of the five model predictors. Each predictor was in turn left out of a model, and the of the subset model was evaluated and compared to the of the full model. A reduction in is interpreted as the reduction explanative power by the model when that predictor is removed.

#### Optimal Prediction

Prediction of the optimal data-hardware configuration for an SDM consisted of a four-step process. First, the accuracy model was used to identify the accuracy-maximizing data configuration. Second, the performance model was used to predict the execution time of the accuracy-maximizing model run on a large set of hardware configurations. Third, multidimensional hierarchical clustering was used to assemble groups of configurations similar to one another. Finally, the hardware cluster with the lowest runtime, cost, and uncertainty was selected as optimal.

To predict the accuracy-maximizing configuration of data for an SDM, the corresponding accuracy model was used to predict the accuracy of a finite number of regularly spaced training example-covariate configurations. This set of predictions included every 100 training exmples between 1 and 10,000 and every number of covariates between one and five. Hardware configuration was included, as hardware should not affect SDM accuracy. Predictions were sorted, first by descending order of accuracy, then by ascending order of number of training examples, and finally by ascending number of covariates. Training examples are costly in time, thus, given equal accuracy, the configuration that requires the least number of training examples was preferentially chosen.

The accuracy-maximizing data configuration, and thus expected accuracy, was held constant and used as the data input for the performance model. This model was used to predict the runtime of an experiment with the given number of training examples and covariates on a set of 287 CPU and memory configurations. Candidate configurations were chosen from GCE allowable custom instance types, and covered the parameter space between 1 and 25 cores and 1 and 25 GB of memory. Each runtime prediction was evaluated as a distribution of 1000 posterior samples. The mean of the distribution was used to calculate runtime cost, using GCE custom instance type rates[[4]](#footnote-4), and the standard deviation was used as a measure of prediction uncertainty.

The runtime predictions were then clustered using complete linkage hierarchical clustering on runtime, run cost, and prediction uncertainty. Prior to clustering, the dataset was centered and scaled using the R scale function. The results were plotted as a dendrogram and cut into clusters using the silhouette rule for maximizing within-cluster homogeneity while maximizing out-of-cluster variance {Rousseeuw:1987gv}.

Finally, the clusters were plotted in time-cost-uncertainty space. The ideal scenario would involve no time, no cost, and no uncertainty, which occurs at the origin of these three axes. The Euclidean distance between the centroid of each cluster and the origin was calculated and the cluster with the smallest distance to the origin was identified as the optimal set of hardware configurations for that SDM.

#### Extensions

Several extensions to this optimization are possible, including weighted optimization and constrained optimization. The optimization described above weights each dimension equally, assuming equal importance for run time, run cost, and model uncertainty. However, the routine could easily accommodate user-specific weighting on these components by using multipliers applied after the runtime predictions have been made. This would allow a user to preferentially increase or decrease the effect of a component during further analysis.

Two forms of constrained optimization are also possible, as modifications to the procedure described here: data-constraint and time-constraint. The unconstrained analysis above assumes infinitely large datasets are available to the model user, which is rarely the case. Indeed, many SDM analyses consider datasets of less than 100 occurrences. A data constrained optimization would allow such small datasets to be considered. In this analysis, the space of potential data configurations is first subset to include only those for which data exists. The accuracy-maximizing point is then selected from the subspace, rather than the full space, to reflect a point feasible within the data limitations. The optimization then continues as above.

A second form of constrained optimization is a time-constrained optimization, where runtime must fall below a certain threshold to meet user-defined standards. First, a large set of realistic configurations is selected and the runtime for each experiment is predicted using the performance model. Those configurations that have predicted runtimes that fall within the constraint are fed into the accuracy model. These results are then sorted, yielding an estimate of the highest-accuracy configuration that would fall below the user-defined threshold on runtime.

#### Limitations and Assumptions

This approach has several important limitations. Perhaps most importantly, while a real SDM workflow contains terms other than those directly related to model computation, I focus here on only computing time. User skill and motivation and can depend on many factors that may be difficult or impossible to model. Furthermore, limit my analysis to include only two hardware components, CPU cores and memory, to keep the project to a manageable scope, effectively eliminating Tinterp and Toutput. While excluding terms improves the tractability of the problem, predictions of the true optimal value are no longer possible. Future work could be pointed towards modeling these factors in a form that could be incorporated into the predictive models described here.

A second limitation of the approach described here is that my analysis is limited to virtual computing instances hosted on Google Cloud Compute Engine (GCE), rather than real-world physical machines, limiting the conclusions I am able to draw about the relative benefits of physical hardware and virtual machines. However, this experimental design adds validity to the benchmarks of computing time, by providing them with a consistent environment unaffected by other tasks or concurrent programs (Dongarra et al., 1987). Nonetheless, because real-world machines do have many concurrent and interacting processes running at any one time, this may bias the results. Moreover, by using GCE, I am unable to experimentally vary the CPU clock rate as a hardware component. By using GCE instances, I am limited to only the CPUs provided by Google, which may be updated or changed as they wish. At the current time, GCE provides only one processor type, a 2.6 GHz Intel Xeon E5 processor. Fortunately, my results will not be biased by using machines with different CPU rates, as might be the case if I collected data on physical machines instead of virtual instances.

I limit my work to the analysis of data-driven SDMs, as systematic literature review suggests that a majority of SDM users use these methods. Furthermore, I focus on the most popular R implementations of these SDMs. While there are known limitations to the language design and speed of R (e.g., {Morandat:2012fw}), the platform is the most widely used for SDM analysis. Maxent, is excluded because (1) it is written in Java, with only R bindings linking it to the R platform and (2) it is not open source, it is distributed as a black-box algorithm.

Finally, I was strongly limited by computational cost. In order to gather enough data to develop a robust predictive model, I limited the number of models running longer than several hours. Similarly, I limited my experimentation on virtual servers with expensive hardware configurations to allow for more experimentation on less costly servers. More data collected in all areas, particularly on virtual instances with high memory and many CPUs may improve the robustness of the results presented here.

### Results

#### Model Performance

Predictive models of performance and accuracy were skillful when compared to a holdout testing set. While results varied across SDM classes, all models explained more than 50% of the variance in the data (. The most explanative performance models were for GBM-BRT and MARS, with an of approximately 0.96, and an MSE of approximately 0.05 log-seconds2. The estimates from all models are tightly constrained, as indicated by the standard deviation of the posterior distribution. The posterior SD for GAM, GBM-BRT, and MARS were between 0.01 and 0.035 log-seconds -- small when compared to the mean of the prediction. The standard deviation of the RF posterior is less well constrained, showing the highest uncertainty on the predictions from the model. The full results are presented as Table 1.

The GAM and RF models were less explanatory than the other two SDMs. The GAM runs tended to converge within several seconds (maximum 10.3s), potentially exposing it to a stronger influence by low-level system processes not explicit in the model, resulting in a higher variance dataset, and lower model explanatory power. In contrast, the other three SDM types typically take several hours to converge (maximum GBM-BRT 5,285.0s). Another possible reason for the poorer model performance is a smaller training dataset. Both the GAM and RF models are fit with smaller datasets than the the GBM-BRT and MARS models, which may partially explain their loss of predictive skill. With addition data in the training set, covering more parameterizations, these models would be likely to perform more skillfully. Nonetheless, all models explained a majority of the variance in the SDM runtime. While stochastic variance in system processes is often suggested to impede modeling algorithm execution time, the results shown here indicate that a skillful model can be produced without accounting for this source of error.

The accuracy models were more skillful than the performance models, despite the large number of hyperparameters suggested to enhance SDM accuracy. All accuracy models explained more than 87% of the variance in the data. The RF model demonstrated an of of 0.98 and a MSE of less than 3.5x10-5 AUC. All of the models show well-constrained posterior estimates on their predictions, though the GAM model indicates that its predictions have higher uncertainty associated with them than the other three SDMs. Full results of the assessment are presented in Table 2.

#### Model Drivers

Data volume, in both number of training examples and number of predictors, is the strongest predictor of both SDM runtime and SDM accuracy, across most SDM classes. The influence of data volume on GBM-BRT, GAM, and RF runtime is similar, showing a relationship with the underpinnings of how these models are computed. In theory, these models rely heavily on creating structure from the given input dataset, and their runtime should be roughly proportional to the number of training examples asymptotically {Hastie:2009et}. Indeed, in the GBM-BRT model, the removal of the number of training examples reduces model by over 0.7. MARS and RF also show large reductions in skill if this term is removed (0.36 and 0.38, respectively). Surprisingly, the number of environmental covariates does not appear to be a very important predictor of runtime; only GBM-BRT is influenced by this predictor, and only slightly. Theoretical complexity analyses suggest that learning algorithms are often asymptotically influenced by both training examples and covariates. However, the leave-one-out validation performed here does not suggest that the number of covariates has a strong influence on runtime under real workloads. GAM appears to be controlled by a different set of processes, namely the number of cells onto which the fitted model is projected. This SDM’s fitting time, as described above, is often trivial, instead, nearly all of the total time during each experiment was spent on predicting the fit model onto the novel climate scenario. Without this term, the model would have very little skill beyond random.

Importantly, with the exception of RF, the computing variables, memory and CPUs, have nearly no influence at all on execution time. For GAM, GBM-BRT, and MARS, CPU capacity accounts for less than 1 percent of the total variance. Similarly, for all models, memory accounts for less than 3% of the structure in the data. Random forests, as expected, show a slight dependence on CPU cores, as their computations can be spread across multiple processing units. However, the overall trend of minimal dependence on hardware capabilities is surprising. These findings indicate that, without additional optimization, SDMs should run at roughly the same speed on all computers, from a shared laptop to a high-throughput supercomputer.

MARS shows a surprising and dependence on memory. Upon further investigation, it appears that an error in sampling design may have artificially biased this model to more heavily weight memory. Nearly ten times as many experimental runs two instances -- a one core, 4 GB memory virtual machine and a four core, 16 GB memory virtual machine -- than other instance types. While the parameter space was fully covered, I believe that the additional influence of these data points may have caused biased results, indicated in the model drivers and clearly seen in the optimal results below.

The accuracy of SDM models is also closely tied to the amount of data being used to fit the model. Together, the data configuration accounted for over 50% of the model skill. In particular, RF and GBM-BRT, this can account for nearly 80% of the model’s total predictive skill. Surprisingly, the GAM accuracy is only strongly controlled by the number of covariates in the training set. As theory suggests, hardware has a minimal influence on accuracy.

#### Optimization

In the unconstrained optimization procedure, the algorithm inputs, and therefore the accuracy, were fixed before optimizing the hardware configuration. Each model presented a different accuracy surface. In all the models, the accuracy-maximizing point lies at the right-hand size of the training example-covariate space. However, the individual surface determines the accuracy and the point’s exact position. Most points lie close to the very top right of the space, indicating that many training examples and many covariates lead to the highest accuracy, which we would expect from the investigation of the accuracy model drivers as well. Of course, this space was limited due to experimental design constraints, so if the accuracy-maximizing point lies at the point with the highest data requirements, it may not truly be optimal, but instead require additional data. More data will nearly always produce a better model. However, given my review of the SDM literature, in practice the limits I’ve artificially imposed seem like reasonable limits beyond the data inputs of most SDM studies.

The GAM model fixed inputs at the lowest predicted accuracy, with a maximum predicted accuracy of only 0.71. MARS required the least data, with only 1000 training examples needed to reach the maximum accuracy. The MARS accuracy surface is interesting in that after ~1000 training examples, in which accuracy increases quite quickly, only the addition of more covariates can increase accuracy. This trend is also seen, to a lesser degree, in the GAM model. However, in the RF and GBM-BRT models, additional training examples continue to increase model accuracy throughout the surface. In all cases, additional covariates continued to increase accuracy significantly up to the five covariates included in this analysis.

Once an accuracy-maximizing point was developed for each SDM class, we can proceed to identify the optimal hardware configuration for running this set of algorithm inputs. Because of the different levels of influence of hardware components on each model type, the optimal for each class different slightly. Using the Bayesian posterior distribution of the predictions, rather than the posterior means, means that there are multiple configurations that were statistically indistinguishable from each other. These clusters were taken together as the optimal if their dissimilarity of mean and standard deviation was less than 1.

Only the GBM-BRT model had a clearly defined optimal solution for this problem, with only one distribution significantly different than the rest. The GAM class also called for a cluster of only three similar solutions. RF and MARS, however, each had larger clusters, 9 and 10, respectively. The optimal solution for GAM and GBM-BRT each called for very low hardware requirements, with only two cores and two GB memory. The RF model, as expected because of its ability to run on parallel cores, suggests an optimal with between 7 and 16 cores, though with little memory required. The MARS model, oddly, requires a high number of cores, even though it runs sequentially, and a high amount of memory. The other models all suggest that 2GB of RAM is sufficient to run the model optimally.

#### Discussion

##### Establishing the Accuracy Maximizing Point

Establishing the accuracy-maximizing point under unconstrained optimization yields an interesting perspective on the factors that contribute to SDM accuracy. Curiously, there are significant differences between models. Without an empirical dataset, it would be logical to assume that all models would become more accuracy with more data. Given more predictors and more training examples, the model would be able to better characterize the system which they are describing, thus being better able to predict future situations. However, the dataset does not suggest that this is the case for all models. GBM-BRT and RF require all 10,000 training examples and five covariates to maximize accuracy. By using all data to maximize accuracy, it appears that these two models require additional training examples and covariates, and that accuracy could be further increased by adding more data. However, given the experimental limitations of data collection and the realities of SDM modeling, five covariates and 10,000 training examples is likely close to the maximum amount of data that would be used to fit a typical model. GAM and MARS both require less than the full dataset to maximize accuracy. Both suggest that all five covariates are important to the maximization, but need less training examples. When testing against an independent holdout testing set, overfitting is a problem, because the data may be too closely fit to the training dataset. However, overfitting, while possible, is often associated with complex models, where the number of parameters or terms is high relative to the number of training examples. It is unusual to overfit a model due to adding additional training examples.

Figure [X] demonstrates the accuracy surfaces faced by each optimization as a function of the algorithm inputs covariates and training examples. This figure provides an interesting complementary view to Figure [x] that describes the drivers of model accuracy. This figure establishes the accuracy substitution rate (ASR) between algorithm inputs that characterizes the amount of one input that can be substituted to get to the accuracy obtained by an increase in the other input. The rate clearly changes between models and at different points in the training examples-covariates space. The models all show a rapid increase in accuracy that corresponds to the first ~1000 training examples, after which the number of covariates is the more important term in increase accuracy. GAM, GBM-BRT, and MARS all show nearly vertical contours of accuracy after approximately 2500 training examples, meaning that additional observations are unlikely to significantly affect model accuracy. Instead, to get a higher accuracy, more covariates are needed. GBM-BRT indicates a shift back to a more horizontal contour in the high training examples-high covariates portion of the space, though many thousands of training examples are required to increase accuracy in that way. RF shows an oblique pattern of accuracy contours throughout, indicating that training examples can be easily substituted for covariates, and visa versa, if sufficient data exists. Like GBM-BRT, a flattening of the accuracy contours indicates that the ASR decreases, so that many thousands of training examples or needed to increase accuracy, though adding more covariates is unlikely to significantly affect accuracy either.

The ASR is an important quantity for three reasons. First, it provides the model user a guide on how to achieve the maximum accuracy with the given data. With very little data, there is no reason to expect very high accuracy, but knowing how the accuracy surface lies lets the model user estimate the accuracy before they begin the modeling process. Second, it is important in helping the model user identify when additional data is no longer necessary. Take the MARS model, for example. There is no reason to find more than 1000 training examples if sufficient covariates are on hand. The time and effort going to collecting these training examples would essentially be wasted, since they will not go towards increase accuracy. The researcher could better spend her time finding relevant covariates. Third, the drivers of accuracy are not the same as the drivers of model execution time. The execution time models respond differently to increased numbers of training examples than to numbers of covariates. Therefore, knowing the ASR helps optimize the execution time model. If the same accuracy can be be achieved at many covariates and few training examples, or few covariates and many training examples, if the data is available, it makes more sense in the optimization framework to complete the experiment in the way that minimizes execution time. All of the models respond much more strongly to increase training examples than to increased numbers of covariates. Thus, it make sense to choose the method of obtaining maximum accuracy with the minimum number of training examples. The ASR formalizes the tradeoffs between the two parameters so that this choice can be modeled.

##### Sequential Model Hardware Responses

GAM, GBM-BRT, and MARS are all fit sequentially in the chosen R implementations. The empirical models support the theoretical claim that hardware should minimally affect the execution time of these models. The optimal cluster of posterior distances either contains a wide range of optimal machine types, or a clear preference for few cores and low memory. In either case, there is no demonstrated advantage to a more powerful hardware configuration. The MARS optimal configuration cluster calls for all configurations between 2 and 16 cores. Statistically, there is no quantitative difference between provisioning and running the model on a computer with two cores and one with 16 cores. THE GBM-BRT optimal cluster has a clear preference for a single core configuration. No optimal configurations are suggested with multiple cores. However, the optimal cluster suggests between 4 and 22 GB of RAM. Again, because all configurations fall within the same cluster, the same optimal result is expected on both the configuration with one core and 4 GB as the one with one core and 22 GB. Finally, the GAM SDM shows a small range of core types and a preference for 2 GB of RAM. In this cluster, all configuration with 2, 3, or 4 cores and 2GB of RAM are statistically similar.

It is important to recognize the different scale of response times predicted by the optimal cluster in each SDM class. The GAM algorithms converges within about 5 seconds, incurring only fractions of a cent each time the model is run. The GAM model is not strongly influenced by either algorithm input or hardware, making in only sensitive to the regional extent of the model run. MARS also terminates quite quickly, typically within 16-18 seconds under the accuracy-maximizing algorithm inputs, which only results in 15-20 cent model runs. GBM-BRT SDMs however, are much slower to converge, taking over 2500 seconds to fit the accuracy-maximizing inputs. Of course, these models are significantly more expensive due to their increased time, with several configurations in the optimal cluster costing over $5. While the time and money expended on these runs are significantly more than the GAM and MARS runs, the expected accuracy is significantly higher. GAMS and MARS both show maximum expected accuracy in the 0.7 range, while GBM-BRT maximum accuracy is closer to 0.85. This accuracy increase takes the model results from qualitatively ‘fair’ to ‘good’ (Araújo, Pearson, Thuiller, & Erhard, 2005; Swets, 1988). The specific application may dictate whether or not the tradeoff between an increase in accuracy is worth the 10+ fold increase in execution time and cost.

The models also have varying degrees of uncertainty associated with their prediction. The GAM SDM predictions have a mean posterior standard deviation of distance of about 0.5, which is approximately 10% of their total runtime. Therefore, we are relatively certain in these predictions. Our certainty diminishes, however, with both MARS and GBM-BRT. These model predictions have larger uncertainties as a percentage of their posterior mean, both approaching 50%. The difference between the three model classes in posterior variance is likely due to the fact that GAMs converge so quickly, so all models are relatively homogeneous in run time. MARS and GBM-BRTs have much larger variance overall in the experimental dataset, so the corresponding models place less confidence in each prediction. However, the clustering algorithm takes both posterior spread and posterior mean into account when determining the optimal cluster, so these predictions are the ones that have both the lowest mean and lowest posterior standard deviation.

The models do not appear to respond to changes in memory. The GAM and GBM-BRT optimal clusters require very small amounts of memory. This is consistent with the relative size of the input datasets to the instances’ main memory. MARS, however, suggests an optimal of 16GB of RAM. This appears to be statistically significant, because there is no intra-cluster variation of memory prediction. There is no clear reason why the MARS models should require greater memory than the other sequential algorithms. In addition, these models fail under very large dataset sizes. When the datasets exceeded several hundred thousand rows (100MB), the algorithms would fail to properly manage in-memory storage and crash. R is known for poor memory management and for making many copies of data objects within built-in and external packages (Johnson, 2012). When datasets become large, making in-memory copies is not possible, resulting in program termination. Some programmers have addressed these problems with packages that can handle datasets too large to fit into an instance’s main memory (https://cran.r-project.org/web/views/HighPerformanceComputing.html accessed October 10, 2016). However, popular SDM packages require modification before they can take advantages of many of these implementations.

##### Random Forest Model Hardware Responses

Random Forest SDMs are embarrassingly parallel, and so can easily leverage additional cores in powerful hardware configurations. The accuracy maximum is on-par with the GBM-BRT SDM runs, but in a fraction of the time. The RF accuracy-maximizing point is the same as that for GBM-BRT, with 10000 training examples and 5 covariates. However, random forests can be fit in between 10% and 30% of the time to fit a GBM-BRT model. While the GBM-BRT took between 150 and 550 seconds to converge, the member of the RF optimal cluster all converge between 46 and 49 seconds. This results in a cost between 13 and 19 cents, closer to the cost of MARS and GAM models. Because the accuracy is so high, the research seeking the truly optimal configuration would rationally be better off choosing the RF model, run in parallel across many cores, rather than waiting for the GBM-BRT to converge sequentially.

The Random Forest optimal cluster shows an interesting trade off between monetary cost and time. Statistically, there is no euclidean distance difference between the member of the cluster, with a mean distance of 53 and standard deviation of 15.6. However, because the algorithm can make effective use of the additional hardware, more powerful hardware is accompanied by an decreased execution time but an increased rate. Conversely, the cheaper machines with less cores take additional time to fit, but have a cheaper rate. In total, the two really balance each other out, clearly illustrating the tension between hardware cost and time when considering algorithms that can effectively leverage multiple cores. In the unconstrained optimal accuracy-maximizing configuration, everywhere between 9 and 17 cores can be considered optimal. However, because the floor is so high – 9 cores – it is statistically significant that the algorithm requires additional cores to be optimal.

The random forest, like all algorithms able to be run in parallel, is subject to diminishing speed returns as it is spread across additional cores(Gustafson, 1988). Because the algorithm must run, at least in part, sequentially, when setting up and combining the trees in the ensemble, it is theoretically impossible to obtain an infinite speedup. Because of the challenge of managing inter-core communication, as more cores are added, it becomes more and more difficult to outweigh the benefits of parallelization with the increased overhead. Figure [X] shows the diminishing marginal returns that are demonstrated by the RF algorithm when run on between 1 and 25 cores. The difference between a model run sequentially and one run in parallel but with just two cores is great. However, the marginal benefit of adding the 16th core is small, in comparison. This is often measured as parallel efficiency, which is given by the difference between parallel and serial runtimes of the algorithm, divided by the number of cores the parallel implementation was run on. Efficiency gives a measure of diminishing returns of adding additional cores by penalizing the addition of cores, where speedup simply indicates that the decrease in execution time. The ideal algorithm would show an efficiency of 1, where there are no diminishing marginal returns over all cores. My dataset shows that efficiencies vary on workload size, but range between 0.05 and 0.4 at 25 cores. Far from ideal, but it is important to note that the bigger workloads come closer to the ideal efficiency, at least on this range of cores.

The amount of marginal return when adding additional cores is dependent on workload size (Figure [X]). Small workloads (1000 training examples) experience a great gain in execution time when the first execution time when the first parallel core is added, but then experience a steep linear decrease in marginal decreases in runtime. Large workloads, over 10,000 training examples, experience a much slower decline in marginal returns. Because there is more work to do in fitting the, now more complex, model surface, more work can be done spread across cores, before it is combined and returned to the user. Therefore, these models show a less steep decrease in marginal benefit. The addition of the 14th, 15th, and 16th cores is still meaningful when fitting a large dataset, when these additions might not be worth in when fitting a smaller dataset. As more data becomes available with which to fit SDMs, these results suggest that efficiency will increase when fitting parallel algorithms, making it more worth it to invest in fitting models on powerful configurations.

##### Utility of Constrained Optimization

The unconstrained maximization routine to fix algorithm inputs at the accuracy-maximizing point then sequentially optimize hardware to balance time and cost is useful when neither data nor runtime or cost is limited. However, in many real-world situations, scenarios where neither time, cost, or data is limited are unlikely. The methods described here can be used alone or in conjunction to meet the realities of modeling with real data. The two presented here demonstrate two likely cases when determining the best configuration given the circumstances.

A constraint on the number of training examples or covariates available to the research seems probable. My analysis of the SDM literature shows that a significant portion of recent studies fit models with less than 100 training examples. Given the low-friction availability of biodiversity records, both past and contemporary, it is logical to conclude that these studies were fit with such small datasets due to lack of digital data. A similar analysis shows that many studies also use fewer than five covariates. Therefore, it is clearly important to be able to determine the maximum accuracy theoretically possible given a constraint on the available data, either training examples or covariates, and optimize the hardware when facing a reduced dataset.

Under a severely constrained dataset, with only 45 training examples, the models showed similar patterns as under the unconstrained optimal. GBM-BRT shows a wide spread of both cores and RAM requirements in the optimal cluster. Given the execution time, which is very small when compared to the unconstrained optimal, we are able to conclude that the runtime is just so low that any hardware configuration is just about the when computing it. Similarly, both GAM and MARS show the same optimal clusters for the data-constrained and unconstrained predictions. Because hardware has little bearing on the execution time of these models, it makes sense that if the optimal configuration to compute a hard model was on very weak hardware, the optimal to compute a small problem would be the same. The random forest model shows a requirement for slightly fewer cores (5) than under no constraints, which is consistent when fitting a model with less data.

It can also be useful to place a hard maximum bound on the execution time or cost of the SDM. If there is a finite amount of money that can be allotted to limit each model run to a fraction of the total available funds, allowing the researcher to complete all modeling experiments without going over budget. Here, however, I demonstrate the ability to place a constraint in the execution time on the model, and to come up with the optimal algorithm and hardware configuration under that limitation. This example is best thought of in terms of cloud computing and latency. While most SDMs are currently computed on desktop and laptop personal computers, it is likely that these models could be efficiently computed on a client-server environment. If the servers were distributed on a cloud platform, my framework could be used to automatically provision the optimal configuration that maximized the accuracy of the model results returned to the users, while minimizing the time and cost incurred by the server manager. However, users of interactive web applications are apt to loose interest and turn their attention to other tasks if the application takes too long to respond (Roth, 2013). If the user requests the unconstrained optimal for, say, the GBM-BRT model, they may be waiting over 1 hour for a response. It would benefit the server manager, then, to limit the time it takes to complete the model so that (a) their costs are minimized and (b) they retain the user’s interest and business. By constraining the optimal with a hard maximum bound on the time it takes to compute the model, the server manager can provision the cheapest resources, return results of high accuracy, and continue to keep their audience engaged.

Another way to optimize configurations that was not explored in this work was 3-way joint optimization between accuracy, cost, and time. In my method, accuracy is first maximized to yield the accuracy-maximizing combination of algorithm inputs. Then, hardware is optimized to minimize time and costs. Some limited use cases may wish to explicitly model the multi-directional trade off between these three parameters. This optimization problem could easily be accomplished using euclidean distance minimization, as I’ve discussed, but with the inclusion of a 1-accuracy term in the distance calculation. Then, the three-way joint minimum would be the one that best balanced the complement of accuracy, time and cost. I saw few use cases that would find such a technique useful, since scientists are unlikely to want to explicitly compromise accuracy for a decrease in cost. Instead, it seemed more liekly one would want to set a limit on the maximum allowable time or cost, then find the best accuracy could be obtained out of that. Therefore, the three-way joint optimization was not presented more rigorously, but could easily be implemented using the same dataset.

##### Cloud Computing’s Role

Given the relatively minor contributions from hardware to model execution time, the sequential models are not, in general, good candidates to obtain benefits from transitioning to the pay-as-you-go, utility model of cloud computing. Because GAM, GBM-BRT, and MARS all have an unconstrained optimal cluster of very weak hardware configurations, it does not benefit the user to put these models into a cloud enabled system that can be adapted to very powerful instances. Rather, the user is best off using the cheapest available instances. If they choose to provision additional cores or RAM, they are likely wasting money that is not returned by decreased execution time. If the user designs the cloud-based ecosystem correctly, she may be able to run multiple model instances, effectively in parallel, by simultaneously provisioning multiple isolated computing nodes. Each node could then run independently, fitting the model for a single SDM run. To make this work automatically, efficiently, and without error would take a significant amount of technical skill. Particularly the centralization of results into a cohesive database would be challenging. However, it would allow users of sequential models to benefit from working in the cloud.

Random Forests, however, are a good candidate for transitioning into a full cloud platform, running on a single node with multiple CPU cores. Random forests benefit immensely from parallelization, and therefore require hardware configurations that are more powerful than the most basic requirements of the sequential models. Furthermore, the requirements of the exact number of cores is dictated by the workload, as demonstrated by the data-constrained optimal configuration, which uses a smaller workload, and results in a less-powerful optimal configuration. Thus, the ability to easily scale hardware based on problem specific features would be helpful to RF model users. This ability is provided by cloud computing IaaS providers, where users could put all necessary modeling code and programs (e.g., R) onto a virtual hard disk, and then automatically scale the hardware configuration up or down depending on the difficulty of the problem.

Of course, the difficulty of the transition to a cloud-based solution may outweigh its benefits, particularly for small collections of models. It is not a trivial task to provision and configure virtual hardware and install and prepare modeling software effectively on a cloud instance. When only a few models are being run, it is foreseeable that this step would far outweigh any benefit obtained by proper scaling of hardware capabilities. However, in a large modeling experiment, for researchers experienced with virtual machines, or a server-based SDM enterprise, this benefits of running the models on the optimal balance of cost and execution time may be significantly larger than the configuration time.

Moreover, cloud computing’s role becomes more important as workloads and dataset sizes scale up. As figure [x] shows, RF models with very large input datasets are more efficient to run on many parallel cores than smaller workloads. Problems with large numbers of training examples are therefore better suited to running on the cloud-based environment, because they can be easily scaled up to many cores, rather than being limited to the number of cores on a purchased physical machine. Biodiversity database size trends suggests that much more data will be available to research to fit SDMs over the next decade. Therefore, SDMs, when fit with parallel methods, are likely to see increased benefits of running in a cloud-based environment.

##### Future Model Development

Because the training time of most SDMs, excluding random forests, is not driven by hardware capabilities, they are unlikely to be able to cope with the massive influx of data. (Austin, 2007) has posited that a solid foundation of ecological theory is essential to the correct prediction and interpretation of species distribution models. He notes that the ecological underpinnings of the statistics are, perhaps, more important the statistical method itself. (Elith & Leathwick, 2009a) further suggest that additional improvements in species distribution modeling will come from the incorporation of additional, ecological relevant information in the statistical model itself, and the covariates used to fit it. Indeed, “further advances in SDM are more likely to come from better integration of theory, concepts, and practice than from improved methods per se” (Elith & Leathwick, 2009a).

Given the results presented here, modelers should clearly focus their effort not only on incorporating ecological realism into complex statistical models, but on optimizing these models to take advantage of parallelism, high performance libraries, and cloud computing environments. While ecological datasets may not have been large in the recent past, they can now truly be considered Big Data. Therefore, new model developments should be undertaken to develop models that more effectively leverage advanced computing infrastructure, including multiple cores and more effective memory management. While incorporation of additional ecological relevant information will help to strengthen the the ecological inference made from the model, incorporation of algorithm design techniques that facilitate the inclusion of Big Data will facilitate the inclusion of information from far more data points. More training data is nearly always associated with a better model fit, suggesting that the ability to better leverage large collections of data will improve the robustness of SDM models and their interpretations.

Once the models are fit, prediction, even for large datasets, is not a particularly large problem, rather, it is the model fitting process that must be differently designed. My results indicate that prediction takes only a fraction of the fitting time, even when a high spatial resolution is specified, and many thousands (or even millions) of points must be predicted. In the case of additive tree models, this is only a matter of evaluating the predictor set at each splitting point in the internal nodes of the tree, and averaging the predicted response of each tree in the ensemble. So even with very large datasets, this term is relatively small. The more important term is the fitting time. Random forests are a model that is considered embarrassingly parallel. Each tree can be grown in isolation, and the only step that must be done sequentially is when the trees are combined into the ensemble. Conversely, additive regression trees (GBM-BRT) are exceptionally difficult to parallelize, since each tree must be built in sequence, so it can develop a model that reduces residual variance. The overhead of cross-core communication is prohibitively high if the trees are built in parallel. While it is difficult, some attempts to build these models in parallel have been successful (Tyree, Weinberger, Agrawal, & Paykin, 2011). This study used very large datasets, and novel implementations of the regression tree approach to parallelize it.

##### Framework Extensibility

Though these results are specific to choice of SDM classes and parameterizations chosen here, the framework could, theoretically, be applied to any class of predictive modeling workflow. My modeling approach relies on algorithm inputs and hardware capabilities and has no intrinsic relationship to species distribution modeling. To extend the framework to additional classes and scenarios, it would first be necessary to consider the set of factors that could affect accuracy. If the model can fairly be characterized in terms of the number of observations and the number of features (covariates) in each training examples, this framework would likely be a good fit. If there are many tuning parameters that are hypothesized to significantly control accuracy, the framework could still be used, but may need some modification to optimize accuracy in more than two dimensions. If, on the other hand, a particular predictive model’s accuracy is solely controlled by parameter settings, or the problem domain does not support different training set dimensions, this framework is likely not suitable without heavy modification.

One of the main limitation of utility of the results discussed here is that they will quickly become outdated as the computing landscape changes. Hardware capabilities continue to improve, yielding faster processors, better cores, and higher performance memory cells. Furthermore, prices for computing resources continue to decrease, particularly as cloud computing becomes mainstream. This framework should be relatively robust to both changing hardware capabilities and cost surfaces. To adapt to more performant hardware, additional data can be collected using similar methods, the execution time models refit, and predictions re-run. If the clock rate on the processor is the same, this can be done directly. If not, as is likely, the new data would have to be corrected, or the new data may bias the results. To incorporate a new pricing scheme, the optimality prediction and clustering could simply be re-run using the new price surface. The existing data and models would not need to be altered. The optimization that balances time and cost would be reassessed to determine a new optimal cluster given the updated pricing information.

My results suggest fairly short runtimes and very low costs, for all models. The longest observations of runtime are only several hours, far short of the several weeks suggested by Elith et al. (2006). However, as data volumes grow and modeling approaches become more common, even these short runtimes become formidable. Many studies now model hundred or even thousands of species (e.g., (Rezende, Oliveira-Filho, Eisenlohr, Kamino, & Vibrans, 2015)). Some large scale projects attempt to model entire biotas in a single exercise. Candela et al. (2013) describe a processes for routinely modeling and publishing maps of distributions over 11,000 marine species. Nearly all contemporary SDM studies in the literature use multiple model classes, predict to multiple time periods, and/or fit with several climate model scenarios. When doing large scale biotic modeling with multiple models to be fit for each taxon, spending several hours for each model is no longer feasible. Even if models fit in tens of seconds, this may add up to a large time sink and big computing bill. Therefore, while the individual results presented here may seem trivial, they add up quickly when considering modeling activities typical in contemporary SDM literature.

### Conclusion

In this thesis, I presented a framework for approaching the tradeoff between accuracy, time, and money when considering the provisioning of computing resources for species distribution models. A theoretical model for assessing the time to reach a user’s goal when using a model was developed using a hypothetical user and scenario. Empirical observations of the run time and accuracy of over 30,000 simulations for four SDM classes were collected under systematically varied inputs, yielding a robust experimental dataset. The dataset was used to fit a Bayesian regression tree model for time and accuracy, and these models were used to (1) evaluate the drivers of SDM accuracy and execution time and (2) optimize algorithm inputs and hardware capabilities in order to maximize accuracy while jointly minimizing execution time and cost. The unconstrained optimal in the presence of no limit on data or time for all model classes was determined. Finally, demonstrative examples of optimizing the SDM configuration under a time limit and data limit were worked out to show the ability to find optimal solutions under various constraints. The posterior distributions of execution time and cost from the Bayesian prediction models were used, in conjunction with a hierarchical clustering method, to determine a set of statistically similar configurations that had low prediction uncertainty.

The models and method show considerable skill in their predictive ability, and can be used to improve researchers allocation of time and money. My findings suggest that contemporary models are unable to handle large datasets or effectively leverage high performance computing resources, and are likely to struggle under the massive influx of data into global biodiversity databases. Models that can be parallelized show promising ability to balance accuracy and cost while using multiple cores. Future model development should be directly towards enabling state-of-the-art SDM algorithms to effectively run in the cloud by modifying current implementations to support very large datasets and split workloads across multiple computing cores.

1. As of November, 2016 [↑](#footnote-ref-1)
2. http://www.nsf.gov/pubs/2013/nsf13602/nsf13602.htm [↑](#footnote-ref-2)
3. https://aws.amazon.com/government-education/research-and-technical-computing/. Accessed 18 September, 2016. [↑](#footnote-ref-3)
4. https://cloud.google.com/compute/pricing#custommachinetypepricing [↑](#footnote-ref-4)