## 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; Wilfried 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). Assuming uniformitarianism (Williams & Jackson, 2007), statistical methods that quantify species responses to climatic gradients can then be used to forecast future biotic assemblages under different warming scenarios (Clark, Gelfand, Woodall, & Zhu, 2014; Guisan & Thuiller, 2005a; Guisan & Zimmerman, 2000; Guisan et al., 2013; Maguire, Nieto-Lugilde, Fitzpatrick, Williams, & Blois, 2015; Wilfried Thuiller et al., 2008a). Despite the potential for widespread, irreversible changes to the Earth’s biosphere (Barnosky et al., 2012), a growing volume of ecological data puts contemporary global change researchers in a position to forecast and mitigate impending catastrophic changes. Environmental monitoring efforts, such as the Long Term Ecological Research program (LTER, (Hobbie, Carpenter, Grimm, Gosz, & Seastedt, 2003)), National Ecological Observatory Network (NEON, (Schimel, Keller, Duffy, Alves, & Aulenbach, 2009)), community curated databases, like 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), form a network of informatics infrastructure for supporting global environmental change research. New cyberinfrastructures for ecoinformatics organize, store, and serve a tremendous amount of information to researchers attempting understand and forecast responses to perturbations in the earth system (Brewer, Jackson, & Williams, 2012; Michener & Jones, 2012). Paleoenvironmental proxy data, including fossil pollen, macrofossils, and freshwater and marine diatoms, can be an important supplement to modern biodiversity data and are an important part of these informatics efforts. When forecasting to future climate spaces, paleoecolgical proxies can enhance understanding climatic changes by providing information about bioclimatic gradients not found on Earth today (Veloz et al., 2012). Understanding responses to past climates can improve prediction of distributions under climates with no modern analog that are likely to emerge in the near future(Williams & Jackson, 2007). However, though collections of modern and historical biodiversity data have the potential for forecasting studies, their volume, uncertainty, and heterogeneity can make their uptake challenging for ecologists (Hampton et al., 2013). Once a dataset has been assembled, statistical methods can be used to mine it for new insight, extract parameters, and simulate future scenarios, though these techniques can be computationally demanding, particularly on larger datasets (Huang et al., 2013). Many contemporary Big Data applications, such as the popular microblogging service Twitter (http://twitter.com), require distributed, parallel, streaming methods to identify key analytical trends in real time (e.g., Bifet, Holmes, Pfahringer, & Gavalda, 2011). With the recent growth in monitoring and occurrence data, some ecological analysts apply similar, data-driven machine-learning techniques to ecological forecasting problems and have seen significant increases in predictive skill (Elith et al., 2006). Climate-driven ecological forecasting models, variously called ecological niche models (Peterson, 2003), predictive habitat distribution models (Guisan & Zimmerman, 2000), and species distribution models (SDMs)(Guisan & Thuiller, 2005b), have seen extensive application in ecology, global change biology, evolutionary biogeography (Araújo, Whittaker, Ladle, & Erhard, 2005; Wilfried Thuiller et al., 2008a), reserve selection (Guisan et al., 2013), and invasive species management (Ficetola, Thuiller, & Miaud, 2007). SDMs characterize a species’ response to biospatial environmental gradients (Franklin, 2010), and use these responses to forecast potential future distributions under climate change scenarios. Recently, studies with an large number of occurrence records, along with computationally intensive modeling approaches, including nonparametric data-driven techniques (Elith et al., 2006) and Bayesian methods that rely on repeated sampling of full joint probability distributions (Clark et al., 2014; Dawson et al., 2016), have become commonplace.

With over 1 billion occurrence records in Neotoma and GBIF, traditional statistical methods for analyzing and forecasting ecological processes often cannot be applied without compromising analysis scope. Most leading SDM methods, though popular in the literature and highly skillful, are not scalable to very large datasets because they are not designed to take advantage of parallel processing or distributed computing. With so much data, ecological modelers are likely to need to adopt Big Data techniques, including distributed computing and ensemble parallelism, common in fields like bioinformatics (Schatz, Langmead, & Salzberg, 2010), geonomics (Stein, 2010), climate analytics (Schnase et al., 2014), and private industry (Mosco, 2014). 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.

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). Cloud computing is a architectural design pattern that provides “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” (Hassan, 2011; Mell & Grance, 2012; Vaquero, Rodero-Merino, Caceres, & Lindner, 2008). 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 have a seemingly unlimited supply of computing resources at their disposal. The cloud has been touted by many of the largest players in Silicon Valley, and is credited with Obama’s 2012 presidential election win, Netflix’s ability to provide streaming entertainment to millions of consumers, and Amazon’s massive success in online retailing ((Mosco, 2014)). In 2010, the U.S. Federal Government embraced the efficiency, speed, and economy of the cloud, requiring all federal agencies to adopt a “Cloud-First” policy when considering new information technology (IT) developments (Kundra, 2010). Accordingly, the National Aeronautics and Space Administration (NASA) and the National Science Foundation (NSF) have both officially endorsed cloud technology (Mosco, 2014). Moreover, researchers in many disciplines have posited that the cloud as the key to solving future computing and modeling challenges (Hsu, Lin, Ouyang, & Guo, 2013; e.g., Yang et al., 2011).

Although the cloud offers a promising technological solution to the computational demands of ecological forecasting, there are few guiding principles on when 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 partitioning large physical resources into smaller virtual machines (VMs), cloud providers can charge consumers for the use of computing resources, rather than their purchase (Hassan, 2011). Under this new operational expense model, consumers are no longer tied to a single hardware configuration, rather they can scale their virtual instances up or down depending on computational demand (Armbrust, Fox, Griffith, Joseph, & Katz, 2009). Despite being more complex to set up and maintain, Cloud-based solutions can offer significant advantages over traditional desktop computing. 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 measured empirically and combined with cost estimates to provide guidance on when a cloud based solution would be economically rational.

In this thesis, I develop a framework for predicting the optimal computing solution for a set of SDM activities in order to better understand limits to the application of climate-driven forecasting models and improve large scale modeling of biodiversity shifts in response to climate change. Treating workflow characteristics as model parameters, I posit a theoretical model for a scenario-specific optimal computing solution that minimizes computational time and financial cost. I gather data on the runtimes of four different classes of species distribution models under different parameterizations and on different computing hardware. Using this empirical dataset, I fit a nonparameteric learning model to this data to assess the drivers of SDM runtime and predict the execution time of future modeling scenarios. My findings suggest that if SDMs, and biodiversity more generally, 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.

## Research questions

My research questions aim to develop a theoretically sound and empirically grounded understanding of the drivers of SDM execution time, and put this understanding towards predicting the computing solution with the lowest cost to the researcher. In particular, my research questions are:

1. With what skill can the runtime of climate-based SDMs be predicted?
2. Can an optimal computing solution for a given modeling scenario be predicted with confidence that this prediction is better than a null model that suggests no change in execution time between SDM scenarios.
3. What are the drivers of SDM execution time, and how sensitive is the runtime to changes in these drivers?

## Background and Previous Work

#### Big Data in Ecology

The vast expansion of data the sciences has necessitated the development of revolutionary measures for data management, analysis, and accessibility (Schaeffer, Pierre, Twigger, White, & Rhee, 2008). Worldwide data volume doubled nine times between 2006 and 2011, and successive doubling has continued into this decade (Chen, Mao, & Liu, 2014). With the large influx of massive geonomic sequences, long term environmental 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). As datasets become larger, significant challenges are encountered, including inability to move datasets across networks, necessity of high performance and high throughput computing techniques, increased metadata requirements for storage and data discovery, and the need to respond to new uses for the data (Schnase et al., 2014).

Ecological occurrence data are records of presence, absence, or abundance or 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, community-curated 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, Arriaga, & Lara (2002); ], they provide a low-friction way to consume large amounts of data that would otherwise be prohibitively time consuming to derive from the literature or in the field (Beck, Böller, Erhardt, & Schwanghart, 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 centralized 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 “term used to describe data sets so large and complex that they become awkward to work with using standard statistical software” (Snijders, Matzat, & Reips, 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, has described 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, Chui, Brown, Bughin, & Dobbs (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 particularly industry, 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 (2^40 to 2^50 bytes). However, because of ecology’s lack of experience with massive datasets and limited analysis software in the discipline, ecological occurrence data clearly falls under the banner of 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. Consider the complexity of the relationships between different data records, for example. Figure 1 shows the Neotoma relational table structure, and the complicated web of relationships between each entity. 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, 2016). While occurrence data does not require the disk space of popular commercial applications like Twitter and Youtube, it has recently demonstrated a need for new, custom built tools to store, analyze, and use large numbers of records.

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 alone has become challenging to manage. Figures 2a and 2b 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 [XXX] individual occurrence records, and associated spatial, temporal, and taxonomic metadata, corresponding to an average growth rate of 1.4 records [[[XXX 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, Smol, & Last, 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 at the turn of the century, the facility’s holdings have grown nearly 300%, from about 180 million records in 2001 to over 614 million records today. GBIF’s reliance on literature and museum specimens allow its holdings to precede its origin in 2001, however, it is within the last 15 years that the data’s volume has become clearly apparent.

The second characteristic of Big Data in the four V’s framework is the Variety of the data, and its ‘various types with complicated relationships’ (Yang & Huang, 2013). Biodiversity data is highly diverse with many very complicated relationships and interrelationships. As shown in Figure 3a, 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 3b). Though the records coexist in large biodiversity database, they are distinctly different, derived using different protocols by different communities of researchers.

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 to specific places on the earth’s surface. The spatial information in these databases is supplemented by other fields that describe the location of the observation, including depositional setting, lake area, and site altitude, to improve contextual interpretation of occurrence data. Managing data with a spatial component is nearly always more challenging than managing data without it. 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, Wu, Huang, Li, & Li, 2011). Furthermore, occurrence data represents the work of many dispersed individual researchers and research teams. The controlled vocabularies and organization 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 only 2 and the third quantile value is just 7 datasets. Each researcher is apt to use different equipment, employ different lab procedures, and utilize different documentation practices, contributing to a highly variable dataset.

Biodiversity data also has high levels of uncertainty associated with it, comprising the third of the Four V’s. Some of the sources of uncertainty in the data, like spatial or temporal positional uncertainty can be estimated (Wing, Eklund, & Kellogg, 2005) or modeled (Blaauw, 2010). Other sources of uncertainty have yet to be quantified, for example inter-researcher identification differences, measurement errors, and data lost in the transition from field to lab to database. A recent paper by the Paleon working group used expert elicitation to quantify the differences between the dates assigned to European settlement horizon, a process they argue varies between sites, and depends on the “temporal density of pollen samples, time-averaging of sediments, the rapidity of forest clearance and landscape transformation, the pollen representation of dominant trees, which can dampen or amplify the ragweed signal, and expert knowledge of the region and the late-Holocene history of the site”(Dawson et al., 2016). The findings of this exercise suggest that paleoenvironmental inference from proxy data is highly variable between researchers. Moreover, some information is undoubtedly lost in the process of going from a field site through a lab workflow to being aggregated in the dataset. Though some procedural information accompanies the data records in the database, not all process details can be incorporated into database metadata fields, and probably more importantly, contextual details essential to proper interpretation of the data often gets lost on aggregation.

Both Neotoma and GBIF show high levels of quantifiable uncertainty, and are likely to show high levels of unquantifiable uncertainty as well. 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 uncertainty was 305 meters, and the maximum was 1,970 meters. Clearly, such high levels of uncertainty might be problematic for modeling efforts (Beck et al., 2014). Neotoma records show a similar uncertainty in their temporal information. Neotoma records each have a minimum, maximum, and most likely age for each age control point (e.g., radiocarbon date). Out of a sample of 32,341 age controls in the database, only 5,722 reported any age uncertainty at all. The summary statistics for these age controls suggest that the median age control has a temporal uncertainty of 260.0 years. The 25% percentile is an uncertainty of 137.5 years and the 75% 751.2 years, suggesting that dates are only identifiable down to ± 130 years of the actual date, on average. [NEOTOMA UNCERTAINTY THROUGH TIME]. Considering sediment mixing, laboratory precision, and other processes at work this is a relatively minor uncertainty, but it certainly contributes to occurrence data’s lack of veracity.

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. User’s 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). The rate of increase in data volume in both Neotoma and GBIF is not fast enough to invalidate the results from previous analyses, suggesting that it’s velocity is not high enough to warrant streaming techniques. Neotoma’s growth rate of approximately 1.4 new datasets each day (1990-2016 average) and GBIF’s daily growth rate of about 59,000 records (2000-2015 average) are small compared to the total number records in the database. Unlike in many private sector applications, there is little incentive to researchers to immediately analyze new biodiversity records, since all new findings will be reported in the academic paper cycle, typically several months to years. Moreover, automated analyses of distributional data have been warned against, due to the overall poor data quality (Soberón et al., 2002) and high levels of 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 fall under the auspices of 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 the early 1990s. To fully and accurately derive value from new data being added to distributional databases, novel and advanced 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. Like grid computing, the cloud model standardized requests for computer resource to leverage distributed networks of physical machines to create a computing utility. As Foster, Zhao, Raicu, & Lu (2008) suggest,

“Cloud Computing not only overlaps with Grid Computing, it is indeed evolved out of Grid Computing and relies on Grid Computing as its backbone and infrastructure support. The evolution has been a result of a shift in focus from an infrastructure that delivers storage and compute resources (such is the case in Grids) to one that is economy based aiming to deliver more abstract resources and services (such is the case in Clouds).”

Grid computing never achieved large-scale success in industry, though it did succeed in forming massive federated systems providing computing power and data to scientists that still exist today, such as the Earth System Grid (Foster et al., 2008). Unlike the grid’s collective and project based utility-style, the cloud provides a pay-as-you-go business model and large economies of scale (Armbrust, 2009; Hassan, 2011). While some organizations and universities have developed ‘private clouds’, large collections of virtualized servers not made available to the general public, 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 have access to scalable and cost-effective computational resources(Hsu et al., 2013).

Major scientific organizations in the United States, including the NSF and NASA, have made major pushes to promote cloud computing in their own operation. 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” (“NASAS PROGRESS IN ADOPTING CLOUD-COMPUTING TECHNOLOGIES,” 2013). The federal plan also provisioned programs to help agencies adopt cloud technologies, reducing the effort needed to screen cloud providers for data security policies and enable rapid procurement of cloud services (Kundra, 2010). NASA developed its own high performance open-source cloud stack, Nebula, before concluding that its own needs were better served by public cloud providers (“NASAS PROGRESS IN ADOPTING CLOUD-COMPUTING TECHNOLOGIES,” 2013), though private clouds are still a popular method for large academic organizations (Huang et al., 2013). 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”[[1]](#footnote-30). The public cloud now provides several large open-access datasets for public consumption, including Landsat images, real-time NEXRAD radar, and the 1000 Geonomes project, and claims that many research institutions, including the NASA Jet Propulsion Laboratory, among others, use their products and services[[2]](#footnote-31).

Cloud technology, both public and private, has been extensively lauded for its application to the traditional Big Data fields of bioinformatics (Hsu et al., 2013, Issa et al. (2013); Stein, Isakov, Godowitch, & Draxler, 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). Biomedical scientists have developed a variety of public-cloud based applications including low latency, streaming methods for data analysis (Issa et al., 2013) and biology-specific operating systems that support protein analytics out of the box (Kaján et al., 2013). Climate analytics and reanalysis are also demonstrative of fields adopting cloud computing technology early in their history. 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) note that despite recent advances in computing, geoscientific problems are still limited by computational ability, including data volume bottlenecks, processing limitations, multi-user concurrency, and spatiotemporal velocity of data (Yang et al., 2011). Many scholars suggest that the cloud provides a means of overcoming these challenges by leveraging distributed computational resources without increasing the carbon footprint or financial budget (Yang et al., 2011). Several large spatial data infrastructure (SDI) projects are currently hosted on public clouds, though geospatial algorithms are more difficult to implement on public cloud infrastructure than on more traditional computing environments like grid computing (Huang, Yang, Nebert, Liu, & Wu, 2010). Numerical models, such as real-time dust storm forecasting, have seen significant performance increases when run on high performance cloud VMs (C. Yang et al., 2011). Environmental models, when implemented in a consumption-oriented way, can also be run in the cloud (Granell, Díaz, Schade, Ostländer, & Huerta, 2013). Candela, Castelli, Coro, Pagano, & Sinibaldi (2013) describe a novel ‘hybrid’ platform that supports SDM specifically, suggesting that a cloud-based approach can aid in data discovery and increase processing capabilities. OpenModeller (Souza Muñoz et al., 2009), while not cloud-specific, offers a generic interface for running multiple SDM algorithms on a remote server.

#### Species Distribution Models

Species Distribution Models (SDMs) are a class of statistical models that quantify the relationships between a species and its environmental range determinants (Svenning, Fløjgaard, Marske, Nogués-Bravo, & Normand, 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 its environmental or climatic covariates. Used in a variety of global change-related fields, the models have been shown to provide reliable estimates of climate-driven range shifts when compared to independent datasets (Guisan & Zimmerman, 2000; Guisan et al., 2006). With the widespread availability of statistical software and machine learning code libraries and lower barriers to access to environmental and occurrence data, the utilization of these techniques has grown substantially in recent years (Franklin, 2010; Svenning et al., 2011), shown in Figure 4. Science and engineering as a whole had a 7% growth rate between 20013 and 2013 (National Science Board (U.S.), 2016), while SDM literature far outpaced this with a 10.8% average growth rate over the same period [Don’t know how or if to cite this].

SDMs work by approximating the species niche and then applying it 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 come close to approximating the species’ realized niche (Guisan & Zimmerman, 2000; Miller, Franklin, & ASPINALL, 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. As a fundamental justification for applying predictions across space and time, all SDMs assume niche conservatism, i.e., that the niche of species remains constant across all spaces and times (Pearman, Guisan, Broennimann, & Randin, 2008). Wilfried Thuiller et al. (2008b) argues that SDM fitting with fossil data also lessens effect of assuming niche conservatism through time. Though speciation and evolution are not accounted for in the SDM paradigm, Peterson, Soberon, & Sánchez-Cordero (1999) suggests 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, being present in all environmentally suitable areas while being absent from all unsuitable ones (Nogués-Bravo, 2009). Given dispersal limitations and biotic interactions between species, this is rarely the case. Svenning, Normand, & Skov (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 that were 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 limit to their application.

Despite their strong assumptions, SDMs have been used for a wide variety of paleo and contemporary studies of geographic and environmental distribution. SDMs are very often used to confirm ecological hypotheses, comparing hindcast projections with the fossil record. In this context, SDMs have been used to support hypotheses on the extinction of Eurasian megafauna (Nogués-Bravo, Rodríguez, Hortal, & Batra, 2008), identifying late-Pleistocene glacial refugia (Fløjgaard, Normand, & Skov, 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 often combined with genetic, phylogeographic, and other methods to develop a complete assessment of a species biogeographical history (Fritz et al., 2013). In an anthropogenic climate change context, SDMs have been used to assess the effectiveness of modern reserve planning (Araújo, Cabeza, Thuiller, Hannah, & Williams, 2004), predict the distribution of both endangered (Thuiller, Lavorel, & Araújo, 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 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 have the same fundamental goal of characterizing responses to climatic gradients, (Franklin, 2010) notes a conceptually meaningful way of grouping modeling algorithms into data-driven and model-driven algorithms. The data-driven/model-driven dichotomy is introduced in Hastie, Tibshirani, & Friedman (2009) and often employed when differentiating between ‘statistical’ and ‘machine learning’ algorithms. I add the burgeoning set of methods that employ stochastic, probability-based Bayesian methods to this taxonomy due to their recent uptake and high accuracy. No individual method or class of methods has consistently outperformed any other (ARAUJO & NEW, 2007; Elith et al., 2006; Veloz et al., 2012), though many scholars have attempted to assess interclass variation (Araújo & Guisan, 2006; Elith et al., 2006), and variation between different parameterizations of the same model class (ARAUJO & NEW, 2007; Wilfried Thuiller et al., 2008a; Veloz et al., 2012).

Supervised learning techniques map inputs to outputs by using a set of training examples, , where both values are known, then approximating the the real relationship between the two , with a function ,, that minimizes a loss function based on the difference between the real and predicted value, . Each training example may be composed of a -dimensional vector of predictors, $\pmb{x\_i} = x\_{i1}, x\_{i2}, ..., x\_{ip}, p=1, 2, ..., p$. Models can either make *a priori* assumptions about the form of the input-output relationship or adapt to fit any given matrix of training examples. The former class of models, model-driven models, exhibit low bias but high variance and can make poor predictions if the assumptions are not upheld. The latter, data-driven models, do not rely on any stringent assumptions about the form of the relationship, but any particular subregion depends on only a handful on input points, making the models highly sensitive to small changes in the input data (Hastie et al., 2009).

Model-driven learners fit parametric statistical models to a dataset, making assumptions about how inputs and outputs are related, including linearity and error distribution. These models 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 simple 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, Edwards, & Hastie, 2002; Vincent & Haworth, 1983). Other model-driven techniques include variants of 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. These models, have, in some cases, been shown to significantly out perform their model-driven counterparts (Elith et al., 2006). These models 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, RANDIN, & Guisan, 2006), and maximum entropy techniques (Elith et al., 2010; Phillips & Dudík, 2008). Since its introduction in 2006, MaxEnt, a maximum entropy algorithm and Java-based runtime environment has seen widespread use in SDM and has demonstrated its ability to perform consistently even on small sample sizes (Elith et al., 2010; Phillips & Dudík, 2008; Phillips, Anderson, & Schapire, 2006). Trends in the SDM literature 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 evaluations of MaxEnt, however, claim that its performance may be questionable when compared with other SDM algorithms (Fitzpatrick, Gotelli, & Ellison, 2013). Data-driven models can be 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, which builds a collection of models based on subsets of the input data, and boosting, which combines many weakly predictive models into a single, highly predictive ensemble. A common application of boosting in SDM is in boosted regression trees (Elith et al., 2006, 2008), where stochastic gradient descent is used to sequentially combine many small regression trees together into an ensemble that minimizes model deviance.

Models in the third category utilize Bayesian methods 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, and have been used in several recent SDM studies (Hegel, Cushman, Evans, & Huettmann, 2010). Golding & Purse (2016) introduce SDMs that incorporate Gaussian processes, a flexible method that demonstrates both high predictive accuracy and ecologically sound predictions. While most model- and data-driven SDMs use only abiotic environmental factors as their predictors, using a joint probability distribution of all entities in an ecosystem, joint SDMs can model both the climatic range limitations of a species and its biotic interactions with other species (Clark et al., 2014). Though it can be challenging for ecologists trained in the frequentist perspective to transition to a Bayesian approach (Ellison, 2004; Hegel et al., 2010), some software packages are in development for implementing Bayesian models out-of-the-box in languages like R (e.g., Vieilledent, Latimer, Gelfand, & Merow, 2012). Though MCMC methods are computationally very expensive, numerical approximations and analytical solutions can, when available, significantly reduce computational burden (Golding & Purse, 2016).

A review of recent literature suggests that the majority of ecologists are using nonparametric models for SDM. Of 100 randomly sampled papers from the most recent 4,000 citations in Web of Science that met the query “(Species Distribution Model*) OR (Ecological Niche Model*) OR (Habitat Suitability Model\*)“, the overwhelming majority utilized techniques that were data-driven. Out of 203 modeling experiments described, 38 were model-driven, 131 were data-drive, and 1 was Bayesian. 33 additional experiments used unsupervised clustering analyses that are not capable of predicting future presence. Figure 5 shows the distribution of the 47 different models, and their aggregation into the categories described above. Of all algorithms, MaxEnt was the most popular (64 instances). 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).

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 those limits precisely. 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). Their longest running model is the genetic algorithms for ruleset production, which they claim took six weeks to converge. Other 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 the 2006 analysis, models are still often built sequentially, unable to leverage multiple processors.

In some cases, methodological papers advise against large modeling efforts due to computational limitations. A 2009 review paper in *Trends in Ecology and Evolution* 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, noting that “considerable computational capacity is necessary for the development of models even for a single species” (Peterson, 2003). Wilfried Thuiller et al. (2008a) cautions that “limits to the broad application of this approach may be posed … by the computational challenges encountered in the statistical fitting of complex models.” While better computing infrastructure may alleviate some of this problem, the the computation intensity of SDMs can cause challenges that are difficult to resolve without reducing model complexity.

#### Algorithm Execution Time: Drivers and Measurement

Analyzing the constantly evolving and multifaceted dimensions of computer performance has posed problems for analysts and scholars since the advent of modern computing (Nordhaus & Yale University. Cowles Foundation for Research in Economics, 2001). While some figures, such as millions/billions of floating point operations per seconds (MFLOPS/GFLOPS), are used in the supercomputing literature, a computer’s performance depends on the way in which it is used (Lilja, 2009), rendering such universal metrics incomparable. While it is difficult to effectively and meaningfully quantify a computer’s performance, both empirical and theoretical methods of estimating algorithm execution time exist.

Theoretically, it is possible to determine 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). Asymptotic analysis is useful, because the algorithm that is more efficient asymptotically will typically be the best choice for all but very small inputs (Cormen, 2009). The time complexity of an algorithm is most often applied when considering an algorithm’s scalability (Goldsmith, Aiken, & Wilkerson, 2007). An estimate of worst case 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). For any given input, however, particularly on real-world programs, the actual runtime will vary (Cormen, 2009; Goldsmith et al., 2007).

Empirical complexity studies have attempted to bridge the gap between asymptotic characterization and real programs. These studies use observations of algorithm runtime under different sized parameters and inputs to build models that predict the run time of future model applications and parameterizations. These techniques seek 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, Xu, Hoos, & Leyton-Brown, 2014). Algorithms for solving very difficult (-Hard) combinatorial problems, can exhibit high variance between different problem instances. When a collection of different algorithms are available, empirical runtime modeling can be used to select the model that will most efficiently reach a solution (Hutter, Hoos, & Leyton-Brown, 2014; Leyton-Brown, Nudelman, & Andrew, 2003). Hutter, Xu, et al. (2014) outlines a comprehensive analysis of strategies and methods uses 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, “notwithstanding the fact that they describe the algorithm rather than the problem instance, and hence are directly controllable by the experimenter” (Hutter, Xu, et al., 2014). 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, Hoos, et al., 2014; Hutter, Xu, et al., 2014).

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, Martin, & Worlton (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.

Despite the challenges, several empirical runtime studies have attempted to include computer hardware’s effects on algorithm runtime and have achieved highly accurate results. Wu & Datla (2011) contend that the execution time of a program depends on the complexity of the algorithm and its input data, the static hardware configuration of the resource (e.g., amount and type of RAM, CPU clock rate), and the dynamic system state (e.g., number of processes competing for resources). Sadjadi et al. (2008) model execution time as a linear combination of contributions from elements of the computer’s hardware characteristics, though multivariate nonlinear systems may be more appropriate to capture some of the complex changes possible in the empirical data (Wu & Datla, 2011).

Previous attempts to model runtimes have relied on the subset of computing components thought to directly affect performance. Sadjadi et al. (2008) include CPU rate (GHz) and number of CPU cores, though other studies have also included memory amount and type, buffer size, and CPU cache size (Wu & Datla, 2011). While increasing the clock rate of a CPU is nearly guaranteed to improve execution time, since it can increase the number of operations able to be processed per unit time, the number of CPU cores can also affect execution time by allowing multiple programs to execute in parallel. To take advantage of multiple processors, algorithms must be specifically designed to partition their execution across multiple cores. Moreover, the addition of processor cores will only improve performance up to a point, after which all benefits of load sharing will be outweighed by cross-core communication (Gustafson, 1988). Computer memory can also affect a program’s runtime by reducing the number of times a computer must retrieve data from the physical storage device (e.g., hard disk) and can improve the the amount of concurrent work able to be done on a machine (Wu & Datla, 2011).

#### 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. The use of computing resources is essential towards this end, and the cost of these resources is proportional to their power. Computing prices are set by an external computing provider. As a rational consumer, 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, . As posited by (Wu & Datla, 2011), at any time , the effective processing power of is related to both the static and dynamic configuration of its hardware and software. Thus,
2. in which is the effective processing power, represents the static, hardware capabilities of the machine that do not change with time and represent the portions of the system that do vary with time. Execution times can vary non-deterministically with hardware due to stochastic changes in system state, so represents process uncertainty and natural system variability that cannot be included in the model.
3. Consumers of computing services are part of a market driven by supply and demand, and face a costs set by computing providers dictated by the effective computing power provided. Particularly when operating in the cloud computing paradigm, computing providers can be seen as a utility provider (Foster et al., 2008), while in a traditional desktop computing model, the provider of the computers can be seen as a typical rational producer. Figure [X] demonstrates an instance of a Google Cloud Computing Engine’s cost surface as a function of memory and CPUs. Notice that as the hardware capabilities of the virtual machines increase (i.e., more CPUs, more memory), the cost to a user of provisioning these resources is increased.
4. 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). For example, consider a hypothetical scenario that could apply to a typical species distribution model user (use case adapted from (Smith et al., 2013)):

*Jessica Smith is a land manager at Yellowstone National Park, interested in understanding how Mountain Pine Beetle infestations may change under different anthropogenic climate change scenarios. Dr. Smith primarily wishes to characterize how the beetle range might change under the three different IPCC emissions pathways (Moss et al., 2010), rather than differences between algorithms or characterization of modeling uncertainty.*

From this brief scenario, we are able to infer that the model goal is to model a single taxonomic group (*Dendroctonus ponderosae*), in a single area of known size (Yellowstone National Park, ~3,500 mi^2), under three climate scenarios. She requires only a single modeling algorithm, as she is not interested in assessing the differences in the outputs from multiple SDM models.

To generalize this example, I introduce – a vector of characteristics that fully describe the user’s goals in the scenario. The components of include a number of experiments to undertake, as well as user traits, such as experience with the model and familiarity with the interface employed, motivation, skill, and accuracy required. also contains a list of experiments desired by the user, which specifies the number and character of modeling runs the users wants to do. Each element in the experiments list contains enough information to specify the inputs for a single model run, such as modeling algorithm, spatial resolution, amount of data to be used as training data, availability of environmental covariates, and number of past or future time periods to project onto.

1. The time to compute a given algorithm with a given set of inputs is proportional to , the effective computing power. However, in addition to computing the model, 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
2. In this formulation, represents the portion of time that is spent by user gathering the resources needed to model. In a species distribution modeling context, this term represents the time needed to find and download occurrence points and find and download predictor variables. can be thought of as a function of computing resources available to the user (how fast can data be downloaded?) and the experiment (what is the data?). is the time required by the modeler to prepare the data for entry into an algorithm. In this case, time might include data cleaning, projection, and conversion, as well as setting up and configuring computing platforms and environments, like R. This component can vary widely between modelers and between model applications, based on data source and quality, user skill and motivation, and the interface and equipment user has on hand. (Elith et al., 2006) notes the potential impact of how experienced a user is with a model on the modeling time and results. includes the time it takes to return 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 output must be downloaded over a network to reach the client’s machine. Finally, represents the amount of time spent by the user evaluating model output and determining whether her goals were met during the modeling process.
3. 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
4. Combining equations from (2) and (5), we find the total time cost of 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. Thus, we derive a multivariate cost function for a modeling scenario that takes into account both time spent modeling and the cost of provisioning resources:
5. Each user-based scenario, , will have its own cost curve that’s subject to both the particular characteristics of the workflow and the specific cost surface imposed by the computing provider. If we select a single element from the finite set of all possible ’s, and call it , we obtain a unique cost function for this set of activities that depends only on the computing resources used to fit the model. The cost surface, , is defined for the set of all real computing solutions, however, some may be suboptimal for that particular experiment. Indeed, the optimal solution for is multidimensional minimum of that maximizes model accuracy subject to any constraints imposed by the user. Such constraints may include limited budget, minimum required accuracy, or maximum number of data points.

Without any user constraints, we can determine the unconstrained optimal configuration using a three step process. First, we conceptualize all inputs into the cost equation as orthogonal axes in -dimensional space, where is the number of characteristics in plus the number of hardware variables in . Using the model for computing and a regular sampling method, we calculate the time to compute each experiment within a finite bounded hypercube. Because the computing resources, , should have no impact on the accuracy obtained from the SDM model, we can estimate the accuracy of the model , the algorithm inputs, for every experiment contained within the hypercube. We then search those results for the combination of elements that will result in the highest accuracy.

Second, we slice the hypercube on this set of values that will result in the highest accuracy model. We are left with a hypercube of reduced dimensionality, who axes are the orthogonal components of . For each of potential combinations of , which may be limited by the computing provider and physical reality, we calculate the cost of that experiment in both time and money, using .

Finally, we put each of the points calculated in the second step onto two new orthogonal axes, time and money. The ideal model experiment would lie at the origin of these two axes: no time, no money. However, it is economically and physically unrealistic to think that any model will actually lie at this ideal origin: no computing time is free and every modeling experiment can be broken down into simple instruction sets that must be executed by the computer. Therefore, no matter how fast the computer is, some time will still be expended while executing these commands. Thus, the optimal experiment configuration is the one that lies closest to this unobtainable optimal point. We find this point by calculating the distance between the origin and all candidate resource sets, and select the candidate with the smallest distance to the origin.

If a user constraint is placed on the optimization *a priori*, the same technique can be applied to subset the hypercube to contain only candidate solutions that meet the user’s criteria. For example, if the user has a minimum accuracy under which the results of the model are not acceptable, the first step proceeds as described above. The hypercube is then searched for all potential configurations that might result in an accuracy greater than or equal to the user defined threshold. If no such experiment exists, then the user must relax her constraints or adjust her modeling protocol. If one or more solutions exist, the hypercube is subset to include only these candidates. Then, the second and third steps of optimizing to maximize accuracy while minimizing cost proceeds.

Similarly, a user may place a constraint on the total amount of money or time may be spent on a modeling workflow. In this case, the user would then wish to identify the configuration that maximizes the accuracy while not exceeding the monetary to time threshold. First, the cost of all experiments in the hypercube is estimated using . The set of candidate solutions is truncated to include only those that fall within the user’s predefined limits, if any exist. Finally, the reduced-space hypercube is searched for the configuration that yields the highest accuracy.

### Methods

#### Approach

My methods employ a relatively simple methodology to develop an empirical dataset that can be used to predict the optimal configuration for a SDM workflow. First, using the Google Cloud Computing Engine, a large number of SDM simulations were run under systematically varied inputs and on systematically different hardware. In total, I collected data for four SDM classes and approximately 2600 configurations, resulting in nearly 27,000 model simulations. The data was the used to fit predictive models for accuracy and execution time for each model class, using the characteristics of the configuration as features in model training. The models were validated using several metrics of model skill and interrogated for insight into the drivers of the runtime and accuracy for each SDM class. Finally, the models were used in associated with a large regular grid of potential configurations to form the hypercube, which is subsequently used to predict the optimal configuration for a given SDM scenario.

#### Limitations and Assumptions

This methodological approach has several important limitations. While a real SDM workflow contains many important terms other than the time and money spent modeling (the terms in the equation above), I focus in this thesis on only the computing time in my development on the optimal modeling configuration. The addition of the time to gather and prepare the data and interpret the results are excluded here because they are highly variable based on user skill and motivation and can depend on many factors that may be impossible to model. Furthermore, I exclude the and terms, to keep this project to a manageable scope. While excluded terms other than the computing time improve the analytical tractability of this problem, this exclusion prevents the prediction of the true optimal value. Future work could be pointed towards modeling these additional factors so that they could be effectively incorporated into the optimal prediction model.

A second limitation of the approach described here is that my analysis is limited to virtual computing instances hosted on Google Cloud Computing Engine (GCE), rather than real-world physical machines. This experimental design was developed to add 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). However, because real-world machines do have many concurrent and interacting processes running at any one time, this may bias the results as a source of inference for real-world workflows. It is difficult to characterize whether a particular user will use a dedicated physical machine for developing her models or, instead, use a single machine for both modeling and other purposes, running concurrent programs while waiting for an SDM to converge. While the use of dedicated virtual instances may fast-bias the results, it simplifies the problem into one that can be more easily modeled. Moreover, by using GCE, I am unable to systematically vary the CPU clock rate as a hardware variable. The runtime of any algorithm is directly related to the number of cycles a processor can complete in a given amount of time. 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 state-of-the-art 2.6 GHz Intel Xeon E5 processor. While I am not able to alter the clock rate experimentally, my results will not be biased by using machines with different CPU rates, as might be the case if I used physical machines instead of virtual instances.

I limit my work to the analysis of model classes in the data-driven tier of SDMs, and do not attempt to characterize the computational time of either parametric or Bayesian approaches. Systematic literature review suggests that this is a reasonable simplification to make, because a majority of SDM users utilize these machine learning methods. Using the same logic, I limit my analyses to the R implementations of these SDMs, which ensures that all code libraries are well documented and open source. While there are known limitations to the language design and speed of R, the platform is the most widely used for SDM analyses. Maxent, the most popular modeling algorithm in recent years, 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 for SDM.

Finally, my methodological approach was strongly limited by computational cost, both in financial and time contexts. Each SDM configuration was tested between five and ten times to ensure robust results. Experiments in my set ranged from less than five seconds to greater than five hours. In order to gather enough data to develop a robust predictive model, I limited the number of very long running models. Similarly, I limited my experimentation on virtual servers with very high vCPU counts or memory allocations. The cost of these instances was more than an order of magnitude of larger than smaller instances (> $1/hr), so experimentation was shifted to less costly servers. More data collected in all areas, particularly on virtual instances with high memory and many CPUs may improve the robustness of my results, particularly for very data-intensive models or severs with advanced hardware capabilities. This limits the ability to predict into very advanced configurations. The optimization routines I develop here typically call for more data, which is not available, since my training data was truncated due to these limitations.

#### 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, Elith, & Hastie, 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 run in the R statistical environment (R Core Team, 2016) with the standard packages for fitting these models. GBM-BRT tree models were fit using the dismo package version 1.1-1 (**???**), GAMs using the gam package, version 1.12 (Hastie, 2015), MARS using the earth package version 4.4.4 (Hastie & wrapper, 2016), and random forests with the package randomForest (Liaw & Wiener, 2002). These standard packages were chosen for their popularity in the field, and are designed to represent normal use cases in species distribution modeling workflows, however, it is likely that the results are highly sensitive to specific implementation details.

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). Global Quaternary occurrence records were filtered to only include those in the last 22 kya 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.

The SDM climatic covariates were downscaled and debiased Community Climate System Version 3 (CCSM3) model simulations for North America (David J Lorenz et al., 2016). 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 (D J Lorenz et al., 2016). Bioclimatic variables (BV) (O’Donnell & Ignizio, 2012) were calculated for each timestep using the biovars function in the dismo R package (**???**). For each fossil occurrence, BV values were extracted for that space-time location.

The occurrence-climate datasets were then filtered to include only the six least correlated BV predictors, a common practice when applying learning algorithms and SDMs. Collinearity among predictors can decrease model performance, can cause highly volatile model results, and “in truly extreme cases, prevent the numerical solution of a model” (Obrien, 2007). The Variance Inflation Factor (VIF) was calculated and used to determine variable collinearity. VIF quantifies the expected amount of variance in a regression coefficient that is dues to collinearity in its predictors, lower bounded by 1 (no inflation) with no upper bound. Variance inflation was calculated using the usdm package. Based on the result of this analysis, I retained the six least intercorrelated variables, leaving a maximum correlation of 0.51. The variables I 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 converted to bioclimatic variables and resampled to various resolutions for their use as output layers in different experiments.

#### Computing Infrastructure

I used the Google Cloud Compute Enginge (GCE) to complete all of my experiments. A popular Infrastructure-as-a-Service (IaaS) provider (Hassan, 2011), the platform provisions a wide array of virtual server instances, from the most basic (1 CPU, 0.6 GB RAM) to exceptionally powerful (32 CPU, 208 GB RAM). The platform also provides a set of application programming interfaces (APIs) to promote workflow automation on the virtual instances, as well as a graphical user interface (GUI) for interactive resource provisioning.

Google’s IaaS 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, some even more powerful than Google’s top-end, but do not allow you to create an instance with an arbitrary number of processors and memory. By providing the ability to create custom types, Google’s service fits well into my experimental design, and lets me avoid using software solutions to artificially alter hardware parameters.

I set up a distributed computing system to complete my experiments, featuring one centralized database node and multiple distributed computing nodes. Fault tolerance was important, because I utilized Google’s less expensive ‘preemptible’ resources, which function like normal instances, but can be shutdown at any time if other customers require additional computational power. One Master Node hosts a MySQL database and a control script (written in python), and a pool of computing nodes that are fault tolerant and designed only for computing are provisioned and decommissioned as needed. The compute nodes, which run a Debian Linux operating system, are given only enough information to complete a given SDM, and the Master Node control script manages the progress of the project as a whole. A node.js script provides programmatic access to real-time database content and is used as the linkage between the master node and the worker nodes.

An outline of the system is described below and are illustrated in Figure [X].

1. First, a pool of computing nodes is assembled. The Master Node control script queries the central database for experiments that have yet to be completed or threw an error the last time they were run. The database responds, via the node API, with a JSON object that contains the hardware parameters required by the next experiment. The python script parses the response and uses the ‘gcloud’ tools associated with the GCE to create a pool of virtual instances that have the necessary amount of memory and number of CPUs.
2. Each node in the pool automatically begins running a startup script that begins the modeling process. First, a number of system-wide software packages, including R and Git are installed on the new instance. Git is used to clone the most recent version of the project repository (hosted as a private repository on GitHub) which contains all files necessary to compute an SDM. Once all packages have been installed, the timing script is initialized as a new R session. The R script queries the central database, identifying itself as a computing node with cores and memory. The database responds with the parameters needed to run a single experiment on the given infrastructure. The script then loads the necessary variables and runs the SDM. When finished, it reports its results to the database and marks the experiment as completed. Experiments are continued until there are no more experiments that can be computed on an instance of cores and memory. If an instance is preempted by the system or otherwise crashes, a shutdown script will be executed, marking the in-progress experiment as interrupted – signaling other worker nodes that it should be attempted again later.
3. Because Google charges by the minute for the use of their virtual machines, instances must be torn down as soon as possible. As the computing nodes execute the experiments, the Master Node repeatedly polls the central database to determine the current position within the experiment table, determining the percentage completion of the current group of experiments. Once the group is complete, Master Node will use the gcloud tools to decomission the individual instances, and delete the instance pool and the instance template that was used to create each virtual worker node. After this, the Master Node is the only instance that remains online. At this point, Master Node returns to Step 1 to build a new pool of instances for a new hardware configuration.

#### SDM Model Protocol

The experimental parameters are communicated to the worker node by the central database. The computing node parses the database’s response and starts a new experiment session. First, the set of occurrences corresponding with the species to be modeled is loaded from the disk. It is then randomly partitioned two disjoint subsets, a training set of occurrences, and a testing set of 20% of the total number (*Picea* = 9935, *Quercus* = 8953, *Betula* = 10226, *Tsuga* = 7140). All examples were converted to binary presence-absence values using the Nieto-Lugilde (Nieto-Lugilde, Maguire, Blois, Williams, & Fitzpatrick, 2015) method for determining local presence from fossil pollen records. The training set is then sent to the specified SDM model where a learner is fit using predictors. The model is then used to predict that taxon’s range in 2100 AD under the RCP 8.5 scenario. The holdout testing set is used to evaluate the model’s ability to discriminate presence-absence from the predictors. During the execution, separate times are recorded for model fitting, prediction onto the gridded surface, and accuracy calculation, as well as the total time, which is the sum of the three components. Furthermore, the Area Under the Receiver Operator Curve (AUC) accuracy metric, a popular method of evaluating logistic output (but see (Lobo, Jiménez-Valverde, & Real, 2008)), is computed and stored with the timing results. There is no database I/O inside of the timing script, so results should not be slow-biased by network connection or context switching. Learning parameters (learning rate, number of trees, tree complexity, etc) are held constant for all runs except a small subset which were designed specifically to determine sensitivity to these parameters. Timing was done within R using the proc.time function.

Due to budgetary constraints, and thus inability to cover all possible parameterizations, experiments were divided into several categories in which different variables were systematically altered to determine sensitivity. This compromise aims to capture as much within-parameter variance as possible while simultaneously capturing the influence of interactions between variables. One basic series of experiments was run for all SDMs using a small subset of algorithm inputs (training examples and memory) on a wide variety of VM types (core/memory combinations). Five additional, separate analyses were completed to determine the sensitivity to specific parameterizations. Because execution time can vary non-linearly when the hardware parameters are changed, I tested as many combinations of memory and CPUs as possible. On each computer, a standard set of 160 experiments were run for each sequential SDM (MARS, GBM-BRT, GAM), including four spatial extents, four training example sets, and 10 replicates of each. All experiments were done on the *Picea* data set.

Individual, target experiment sets were done to assess the contribution and sensitivity of individual or sets of parameterizations. While no theoretical difference would suggest that execution times should vary between different taxa, a set of 191 model runs were done to evaluate whether differences exist in practice. The model uses aspatial input sets, which suggests that geographic range, abundance, or taxon-specific patterns should not bias the results of the experiments. Using all four taxa on six different VM instance types, inter-taxonomic sensitivity was recorded. The number of training examples and spatial resolutions was held constant for these runs.

Empirical Performance Models theory suggests that specific algorithm parameterizations will take longer to execute than others (Cannon & John, 2007). SDMs have a large number of potential parameters with which to alter, though many ecologists use the defaults, or packages that make it difficult to change the default parameter values (e.g., dismo (**???**)). To assess the magnitude of changes in execution time due to different parameterizations, the GBM-BRT was tested on a set of 70 different combinations of tree complexity and learning rate. The learning rate parameter of the GBM-BRT model is a shrinkage parameter that reduces the impact of each additional fitted tree, motivated by the boosting paradigm of fitting a model with many small models rather than fewer large trees. If one of the greedy iterations does not improve model fit, the contribution of that iteration can be easily reversed in the subsequent iterations (Natekin, 2013). The tree complexity parameter controls whether interactions between predictors are fitted. If tree complexity is 1, the tree will be an additive model with no interactions. A tree complexity of will produce a model with -way interactions between variables(Elith et al., 2008). These two variables together control the total number of trees needed to fit the model (Elith et al., 2008).

To assess the relative performance of parallel methods over sequential models, RF SDMs were fit both sequentially and in parallel on instances up to 24 CPU cores. Spatial resolution, memory, and taxon were held constant while number of ensemble members and number of training examples were systematically altered for each core. In total, 3576 random forests were fit using the randomForest function with the foreach package providing parallelization support. Sequential runs were fit using the same function but with number of cores set to only 1.

In addition to performance gains made by increasing the number of CPU cores and leveraging parallel methods, increasing instance memory should improve performance for very large datasets. Using simulated datasets, I attempted to assess the performance of the model when faced with more than 1 million input examples. R has little support for high memory tasks, and these tests routinely crashed the computer when trying to fit the SDM due to inability to allocate memory space.

Finally, I evaluated the effect of varying the number of predictors on the execution time of the algorithm. The literature on theoretical complexity of algorithms (e.g., (Hastie et al., 2009)) often characterize the complexity of machine learning algorithms in terms of both number of training examples and number of features in each example. I systematically modified the number of training examples between 1000 and 11000 and the number of predictors on all SDM classes, to get an understanding of how the two parameters interact. Because both of these algorithms run serially, they can safely be run on a single processor without the need for estimating the effects of additional cores.

#### Modeling Execution Time and Accuracy

To model algorithm execution time, I utilized a Bayesian additive regression tree framework, which provides the ability to investigate the mean prediction for each node, as well as gain an entire probability distribution for each prediction point. Tree based models, including additive regression trees and random forests, have seen successful use in the past for empirical performance models, as they are able to effectively model local responses to specific combinations of variables (Hutter, Xu, et al., 2014). Additive regression trees have even greater than single trees to capture interactions between variables and both linear and non-linear effects in the estimated function(Kapelner & Bleich, 2016). By placing the model-building in a Bayesian context, the response at each leaf node will be a probability distribution, rather than the single maximum likelihood estimate. As Kapelner & Bleich (2016) describe, the goal is to estimate from a training set of data $ $. To do this, using Bayesian additive regression trees, approximate the function as

In this framework, is the training set of predictor features, is the unexplained variance remaining. The model is composed as a sum of trees, , with parameters at each leaf node given by , and the tree structure . The tree structure contains information about the splitting rules at each internal (non-leaf) node. is the tree’s terminal node parameters, , where is the total number of terminal nodes in the tree. Each observation’s value is found by navigating through each splitting rule at all internal nodes, summed over all trees. As a Bayesian model, there is a prior distribution placed on the leaf parameters and tree structures and the noise term, which limits a single model term from dominating the model fit. The prior placed on tree structure enforces that trees must be shallow, with few internal nodes, which is ideal when building additive models, because it can reduce the risk of overfitting. The leaf node parameters have a normal prior on them which regularizes the predicted response by drawing the predictions towards the response mean. The noise variance prior is chosen so that 90% of it’s probability mass lies below the expected mean squared error from an ordinary least squared regression, which encourages the error term to be smaller than in an ordinary regression.

The regression trees were fit with the bartMachine R package, version 1.2.3 (Bleich & Kapelner, 2016). For each SDM class, the collected data was randomly split in a training set (80%) and testing set (20%). The training set was used to fit the Bayesian regression tree model, using the default priors suggested by the authors (Kapelner & Bleich, 2016). The analysis performed 1250 Monte Carlo simulations, of which the first 250 were discarded as burn-in, leaving 1000 posterior samples. Each Markov Chain Monte Carlo iteration built an entire additive model of 50 trees.

The log transform of SDM execution time was used a response variable, used because the transform does not allow for negative predictions, which are possible under non-transformed inputs but physically impossible for runtime, and have been shown to be more accurate when observed responses span large ranges (Hutter, Xu, et al., 2014). In the MARS, GAM, and RF SDM classes, the predictor set contained five features: number of training examples, number of cells in the raster grid used for prediction, number of CPU cores, gigabytes of RAM, and number of environmental covariates. In the GBM-BRT SDM case, two additional features were used: tree complexity and learning rate of the algorithm. The accuracy models were fit using the same predictor features set, and the response was the area under the receiver operator curve (AUC). While other accuracy metrics are encountered in the literature, the AUC is a particularly common method of evaluating presence-absence classification accuracy.

#### Model Evaluation

Models were evaluated using the 20% holdout set that was not included in model fitting. Using this testing set, the predictive skill of the execution time and accuracy models were evaluated using mean squared error (MSE) and the model’s value between observed and predicted values. The value can be interpreted as the percentage of variance that is explained using the model, and the MSE is a metric of the average deviation of the predicted values from the observed, real values. The model’s fit was evaluated primarily using the posterior mean returned from the bartMachine predict function. The expected variance in the predictions was also considered, by assessing the standard deviation of the posterior. This can be interpreted as the confidence in the position of the mean used in the previous two metrics. Finally, visual assessment of model results was done by plotting the predicted values against the observed data, which, if perfectly predicted, should fall along the y=x line.

Model drivers were evaluated by cross validating the performance of a separately built model using only a subset of the predictors. Each predictor was left out of a model fit, and the of the subset model was evaluated in its absence. The reduction in model skill by removing the additional term is interpreted as the importance of that variable to the model. Variables with a large reduction in model skill are very important, while variables with small reductions are only slightly important to the overall model fitting. A reduction in is then simply interpreted as the reduction in data variance explained by the model when that predictor is removed.

#### Optimal Prediction

##### Unconstrained Optimization

To predict the optimal configuration, the runtime and accuracy prediction models for each SDM class were used to generate predictions of execution time and accuracy for a large set of potential experiments. Ideally, the model would predict the value at every real combination of computing hardware and algorithm inputs, essentially describing every possible experiment. Although there are a finite number of hardware configurations on the market today, there are an infinite number of ways to combine these with sets of algorithm inputs, which makes the problem intractable. Therefore, it is essential to reduce the scope of the prediction into a large, but finite, selection of possible experimental configurations. To accomplish this, I regularly sampled values along each orthogonal axis of the experimental components, from very small (near zero) values, to very large, but still reasonable, values. This allowed to to partition the infinite space into a real number of combinations to work with. Each candidate configuration in the reduced space was assigned a dollar-per-hour rate using the Google Cloud Engine pricing scheme, based on its number of CPUs and amount of RAM. Using this scheme allows the prediction to be recomputed should a new pricing scheme become available. For every scenario in the candidate set, the execution time (in seconds) and accuracy (in AUC) was predicted. Finally, the execution time was multiplied by the dollar-per-hour rate to obtain a monetary cost for that experiment.

The candidate experiment generation was done using the expand.grid function in R and the model prediction was completed on a 16-core, 60 GB RAM instance on GCE. Predictions were spread across all 16 cores using the parallel package and the mclapply functionality provided in that package. Once each experiment in the hypercube had been predicted, it was possible to predict the optimal configuration under a variety of optimality criteria.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Orthogonal Axis | Minimum Value | Maximum Value | Step Size | Number of Steps |
| Training Examples | 0 | 500000 | 10000 | 51 |
| Cells | 10000 | 1000000 | 100000 | 10 |
| Number of Covariates | 1 | 5 | 1 | 5 |
| CPU Cores | 1 | 24 | 1 | 24 |
| Memory (GB) | 2 | 22 | 2 | 12 |
| Learning Rate *(GBM-BRT Only)* | 0.001 | 0.11 | 0.005 | 22 |
| Tree Complexity *(GBM-BRT Only)* | 1 | 5 | 1 | 5 |

To calculate the unconstrained optimal configuration, the predicted values for execution time and accuracy were predicted for all values in the hypercube set. Using a bivariate interpolation method, the accuracy for every combiantion of algorithm inputs (training examples and number of predictors) was estimated. This entire space was searched for the minimal set of algorithm inputs that would result in the maximum accuracy the maximum accuracy. The minimal set of inputs was defined as the one with the least training examples and the least number of predictors.

This maximum-accuracy point, because it is only defined in terms of algorithm inputs, can be calculated on any combination of computer hardware. Holding the algorithm inputs, and thus, the accuracy, fixed at the maximum combination means that there are a number of different costs and times that will result when this algorithm is calculated, due to inherent differences in the hardware capabilities of each configuration. Using data on 287 computing configurations, the execution time model was used to estimate the cost, in seconds and dollars, for each. Under the Bayesian framework, the model was run 1000 iterations to get a posterior estimate of predicted time. Each time sample in the posterior was multiplied by the configuration’s rate per second to obtain a cost for obtaining the accuracy maximizing point.

The posterior distributions for time and cost were then plotted out on time and cost as orthogonal axes. Using the dist function in R, the euclidean distance between each sample in the posterior and the origin of the two axes was calculated to arrive at a density of potential distances for each hardware configuration. To assess the potential for similarly-distributed configurations that might compete for the lowest-cost and lowest-time, 25 distributions with the lowest means were selected as potential optimal candidates. Using hierarchical agglomerative cluster (hclust), the posterior distributions of distance were clustered according to their mean and standard deviation. The optimal point should both have a low mean, and a low standard deviation, and thus, low uncertainty. The top-25 potential candidate distributions were clustered hierarchically, and then split into distinct clusters at a dissimilarity of 1. Candidates with dissimilarity in their means and standard deviations larger than one were considered statistically different, and those within a dissimilarity of 1 were considered statistically the same. The cluster with the lowest mean distance to the time-cost origin was selected as the optimal cluster, and each member of that cluster was considered equally optimal.

##### Data-Constrained Optimization

Constrained optimization routines proceeded very similarly to the unconstrained analyses above, but a prior constraint was placed on the space to subset it to only potential scenarios that would meet the constraint. A data-constrained optimization was performed by first slicing the accuracy space, as defined by algorithm variables, into reflect only those combinations of algorithm inputs with enough data to meet the constraint. For example, if the constraint called for a hard limit of 500 training examples, the accuracy maximization search would not search any values beyond 500o training examples. Similarly, the number of predictors could be limited if there were few covariates available for the researcher to utilize. Once the accuracy space had been subsetted, it was searched to find the accuracy-maximizing space within it. Once found, the analysis proceeded as in an unconstrained analysis, by fixing inputs to maximize accuracy and then minimizing time and cost.

##### Cost-Constrained Optimization

To calculate a cost- or time-constrained optimal configuration for a given SDM, first, the execution time model for that SDM was used to calculate a regular set of values for average execution time under various algorithm input combinations. Using bivariate interpolation, these values were interpolated across the entire space so each distinct combination of values had its own predicted time. Since the algorithm inputs, number of training examples and number of covariates are strong model drivers, these predictions give a time surface, which is subsequently truncated at the user’s threshold of time and/or cost. Once the time space has been subset, its boundaries are used to subset accuracy space. Next, the accuracy-maximizing point within the time-subset accuracy space is found. As in the data-constrained optimization procedure, the execution time model is used to find the posterior estimates for each hardware type and the time-and-cost minimizing cluster of points.

### Results

#### Model Performance

While the results varied by SDM class, the predictive models for both time and accuracy showed considerable skill. For execution time, the most sucessful models were those for GBM-BRT and MARS, which both showed values of approximately 0.96. This is a surprising amount of variance able to be explained by a farily simple model with only five predictors. The MSE on both of these models was just over 1 second^2 (~ 0.06 ln(seconds)^2), indicating great ability to correctly predict to the observed value. The GAM and RF models showed slightly poorer performance. The variance explained by the GAM model was only 52.1%, while the RF model explained only 58%. While this is still a significant amount, there is still a large portion the the variance in these models that is not explained by the predictor set included here. Despite this, the GAM model still demonstrated a very low MSE (0.0121 ln(seconds)^2). The RF SDMs executed much more slowly, and thus, showed a larger MSE (~0.64 ln(seconds)^2), indeed, the worst MSE of all the SDMs. The variance of the posterior samples shows that most of these estimates are tightly constrained. The posterior standard deviation for GAM, GBM-BRT, and MARS were between 0.01 and 0.035 log-seconds, which is very small when compared to the mean error on the prediction. In addition to poor predictive skill, the posterior standard deviation of the RF model is less well constrained, showing higher uncertainty on the predictions from the model. This model is closer to 0.1 log-seconds in prediction variance.

One possible reason for the lower predictive skill for the GAM model is that it is relatively fast to converge. Most GAM models converged within 10 seconds. The higher MSE may be due to low-level system processes that cannot be incorporated in the model that increase the variance of execution time at these time scales. At larger time scales, these small-scale processes may begin to have less of an effect (minutes to hours). 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 performance and the RF model’s high posterior variance. With more data, perhaps these models would be able to better model particular variable combinations, and more precisely predict future values.

Nonetheless, these models explained a majority of the variance in the SDM runtime. The stochastic variance in benchmarks that is often suggested to cause problems when modeling algorithm execution time is not a significant barrier to modeling SDM runtime. The combination of algorithm input and hardware capabilities together seem sufficient to yield effective predictions for runtime.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | nTrain | nTest | MSE |  | Posterior SD |
| GBM-BRT | 9256 | 2314 | 0.0646 | 0.9615 | 0.0257 |
| GAM | 2636 | 659 | 0.0121 | 0.5213 | 0.01069 |
| MARS | 6632 | 1657 | 0.0561 | 0.9648 | 0.03397 |
| RF | 2861 | 715 | 0.64824 | 0.5851 | 0.10174 |

*Table X* shows the results of the evaluation of runtime model skill.

Given the large number of parameters that can be tinkered with in most implementations of learning algorithms in R, it is surprising how well SDM accuracy can be modeled using this framework. All models explained more than 87% of the data’s variance. The best performing models was the RF model, with an of of 0.98 and a MSE of less than 3.5x10^-5 AUC. Accuracy for other models performed high skillfully as well. The second best of the four models was the MARS model, which explained 94% of the data’s variance and had an mse of 0.00016. GAM and GBM-BRT models performed similarly, each with an just under 0.9. All of the model show fairly constrained posterior estimates on their predictions. Not only are these models accurate, there is very little uncertainty at each leaf node about the value of the response variable, further supporting that this set of predictors are sufficient to fully model accuracy.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | nTrain | nTest | MSE |  | Posterior SD |
| GBM-BRT | 9256 | 2314 | 0.000245 | 0.8748 | 0.0012 |
| GAM | 2636 | 659 | 0.000718 | 0.8993 | 0.004998 |
| MARS | 6632 | 1657 | 0.000168 | 0.94175 | 0.001421 |
| RF | 2861 | 715 | 0.000034 | 0.9818 | 0.00062 |

*Table X* shows the results of the evaluation of accuracy model skill.

#### Model Drivers

Upon investigation of the drivers of SDM runtime and accuracy, it is clear that data volume can often be significant contributor to both accuracy and long run times.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predictor | GBM-BRT | GAM | MARS | RF |
| Number of Predictors | 0.0138269164 | 0.00002921527 | 0 | 0 |
| Number of CPUs | 0.0007196373 | 0.00357912281 | 0.00360559534 | 0.1100450350 |
| Memory (GB) | 0.0014692250 | 0.00077365222 | 0.02060023091 | 0.0104282285 |
| Number of Training Examples | 0.7459094228 | 0.00056633610 | 0.33670908117 | 0.4200279389 |
| Number of Cells | 0.0592688780 | 0.37449462087 | 0.04663870144 | 0.0004306765 |
| Learning Rate | 0.0003260443 | NA | NA | NA |
| Tree Complexity | 0.0281510594 | NA | NA | NA |
| Parallel | NA | NA | NA | 0.0006440635 |
| Sequential | NA | NA | NA | 0.0021486491 |

*Table X* shows the reduction in explained variance () when a model fit will all terms except one is compared to the full model, for an execution time model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Predictor | GBM-BRT | GAM | MARS | RF |
| Number of Predictors | 0.0973334060 | 0.00002921527 | 0.2875883935 | 0.38990130804 |
| Number of CPUs | 0.0006968174 | 0 | 0.0005195602 | 0 |
| Memory (GB) | 0.0001511123 | 0 | 0.0028887021 | 0 |
| Number of Training Examples | 0.6949723552 | 0 | 0.2311663197 | 0.44445147018 |
| Number of Cells | 0.0002975617 | 0 | 0.0001914743 | 0.00003363167 |
| Learning Rate | 0.0038748993 | NA | NA | NA |
| Tree Complexity | 0.0047865705 | NA | NA | NA |

*Table X* shows the reduction in explained variance () when a model fit will all terms except one is compared to the full model, for an accuracy model.

The drivers of model runtime are similar for GBM-BRT, GAM, and RF, and show at least a small relationship with the theoretical underpinnings of how these models are computed. These models rely in a very large part on the number of training examples used to fit the data. This is very logical, as the more data is used to fit the model, the more computation must be done during training. Indeed, in the GBM-BRT model, the removal of the number of training examples reduces model by nearly 0.75. MARS and RF also show large reductions in skill if this term is removed. Surprisingly, the number of environmental covariates does not appear to be a very important predictor of runtime. Theoretical complexity analyses suggests that learning algorithms are often asymptotically proportional in time to a combination of both training examples and number of features in each training example. However, the leave-one-out cross validation performed here does not suggest that the number of covariates used to fit the SDM has any strong impact on the time to fit it. It is also surprising that GBM-BRT do not show more dependence on the learning rate or tree complexity parameters. These parameters together are thought to control the number of trees that are built during model fittings, so, theory, should have at least a moderate impact on the execution time. However, experimental results indicate that less and 1% and about 3% of data variance is all that is explained by these two parameters, respectively.

GAM runtime appears to be controlled solely by the number of cells used for prediction. These models converge quite quickly so their fitting time is negligible. Instead, nearly all of the total time during each experiment was spent on predicting the fit model onto the novel climate scenario. Without knowing the number of cells to predict onto, the GAM model would have nearly no skill at all.

Importantly, with the exception of RF, the computing variables, RAM and cores, have nearly no influence at all on the execution time of the SDMs. In the first three models, the number of cores accounted for less than 0.5% of the variance. For all models, memory accounts for less than 3% of all variance. This lack of dependence on hardware capabilities is surprising. Without any of the variance in the data being explained in the data by the hardware, the models should run at the same speed on my laptop as on the NCAR yellowstone supercomputer. Random Forests do show a dependence on the number of cores on the computer used to run the model, because their computations are split across multiple cores. In this case, the number of cores accounts for 11% of the RF data’s variance.

The accuracy of SDM models can be most effectively predicted by including the number of environmental covariates and the number of training examples. The number of environmental covariates included in model fitting is also very important. In the cases of RF and GBM-BRT, these two together can account for nearly 80% of the model’s total predictive skill. In the GAM model, this is the only significant term towards predicting accuracy. In all models, no other terms consistently increase model skill. Learning rate and tree complexity, designed to regularize the model to prevent overfitting, do not enhance the skill of the GBM-BRT accuracy model.

#### Optimization

It is instructive to discuss the prediction of the unconstrained optimal point for each SDM type. In addition, a data-constrain optimization will be demonstrated to show its utility in situation were little data is available to model users. Finally, a time- constrained optimization was performed and is discussed.

##### Unconstrained Optimal

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.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Fixed Accuracy | Training Examples | covariates |
| GAM | 0.7131 | 9000 | 5 |
| GBM-BRT | 0.8087 | 10000 | 5 |
| MARS | 0.7722 | 1000 | 5 |
| RF | 0.8523 | 10000 | 5 |

***Table X*** Accuracy-Maximizing Points for each SDM

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.

| Configuration Number | CPUs | RAM | Seconds | Dollars | Mean Distance | Distance SD | | :————- | :————- | :————- | :————- | :————- | :————- | | :————- | |13 | 2 | 2 | 5.302108 | 0.3187628| 5.330181| 0.4410313 | |25 | 3 | 2 | 5.302108 | 0.4781441| 5.342165| 0.4420229 | |37 |4 |2 |5.302108 |0.6375255 |5.358898 |0.4434074 |

***Table X*** GAM unconstrained optimal predictions for computing hardware for achieving the accuracy-maximizing point.

| Configuration Number | CPUs | RAM | Seconds | Dollars | Mean Distance | Distance SD | | :————- | :————- | :————- | :————- | :————- | :————- | | :————- | |13| 2| 2 |1489.020| 89.51986| 1796.690 |1545.021|

***Table X*** GBM-BRT unconstrained optimal predictions for computing hardware for achieving the accuracy-maximizing point.

| Configuration Number | CPUs | RAM | Seconds | Dollars | Mean Distance | Distance SD | | :————- | :————- | :————- | :————- | :————- | :————- | | :————- | |145| 13| 2| 21.86844| 8.545748| 24.84752| 8.630945| |157| 14| 2| 21.78000| 9.165896| 25.03071| 8.703743| |169| 15| 2| 21.71175| 9.789829| 25.19974| 8.699931| |133| 12| 2| 22.60012| 8.152314| 25.38249| 8.675544| |121| 11| 2| 22.74969| 7.522412| 25.38255| 9.059256| |181| 16| 2| 21.73022| 10.451368| 25.52088| 8.824121| |109| 10| 2| 23.31605| 7.008804| 25.83321| 9.269416| |193| 17| 2| 21.50065| 10.987265| 25.85966| 9.837260| |97 |9 |2| 24.22601| 6.554105| 26.64450| 9.645858|

***Table X*** RF unconstrained optimal predictions for computing hardware for achieving the accuracy-maximizing point.

| Configuration Number | CPUs | RAM | Seconds | Dollars | Mean Distance | Distance SD | | :————- | :————- | :————- | :————- | :————- | :————- | | :————- | |81 |7| 16| 9.111372| 10.86422| 14.28928| 1.688315| |93 |8| 16| 9.111372| 12.41625| 15.52028| 1.833760| |105| 9 |16| 9.111372| 13.96828| 16.80676| 1.985762| |117| 10| 16| 9.111372| 15.52031| 18.13693| 2.142924| |129| 11| 16| 9.111372| 17.07234| 19.50185| 2.304192| |141| 12| 16| 9.111372| 18.62437| 20.89470| 2.468761| |153| 13| 16| 9.111372| 20.17640| 22.31026| 2.636013| |165| 14| 16| 9.111372| 21.72844| 23.74446| 2.805468| |177| 15| 16| 9.111372| 23.28047| 25.19413| 2.976750| |189| 16| 16| 9.111372| 24.83250| 26.65673| 3.149560|

***Table X*** MARS unconstrained optimal predictions for computing hardware for achieving the accuracy-maximizing point.

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.

#### Data-Constrained Optimization

To demonstrate the ability to predict the optimal configuration under a data-constraint, it is demonstrated here. This is clearly a very common occurrence in SDM applications, where there is simply no more data with which to fit the model. In this case, we’ll choose a relatively extreme example, and fit an optimal configuration with only 45 training examples. Covariates are still allowed to range between one and five.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Fixed Accuracy | Training Examples | covariates |
| GAM | 0.6776 | 45 | 5 |
| GBM-BRT | XXX | XXX | XXX |
| MARS | 0.69457 | 45 | 5 |
| RF | 0.7863 | 1 | 5 |

***Table X*** Accuracy-Maximizing Points for each SDM under a strict data constraint of 45 training examples.

Once the accuracy is maximized at this low point, a significant decrease in the maximized accuracy is apparent, and expected. This decrease was about 0.05-0.1 points on the AUC, which corresponds with a good predictive model with a fair predictive model (FIND CITATION FOR THIS). As expected, most models require all 45 training points, and all require all five covariates to achieve maximum accuracy. The RF suggestion of accuracy maximization of 1 training example seems like a statistical artifact and should be treated with caution.

Configuration Number | CPUs | RAM | Seconds | Dollars | Mean Distance | Distance SD |  
:————- | :————- | :————- | :————- | :————- | :————- | | :————- |  
13| 2| 2| 5.224851| 0.3141180| 5.257580| 0.4889995|  
25| 3| 2 |5.224851 |0.4711771| 5.269401| 0.4900990|  
37| 4| 2| 5.224851| 0.6282361 |5.285905 |0.4916341|

***Table X*** Optimal Hardware for GAM under a strict data constraint of 45 training examples.

XXXX NEED GBM-BRT TABLE HERE XXXX

| Configuration Number | CPUs | RAM | Seconds | Dollars | Mean Distance | Distance SD | | :————- | :————- | :————- | :————- | :————- | :————- | | :————- | |109 |10| 2 |4.738573| 1.4244151| 5.274748| 2.201935| |97 |9 | 2| 4.807257| 1.3005552| 5.298139| 2.156391| |133| 12| 2| 4.677079| 1.6871158| 5.298231| 2.211611 | |121| 11| 2| 4.732381| 1.5648092| 5.315057| 2.235309 | |145| 13| 2| 4.683376| 1.8301697| 5.353266| 2.217866 | |157| 14| 2| 4.679844| 1.9694656| 5.411921| 2.308578 | |169| 15| 2| 4.885003| 2.2026478 |5.706941| 2.324732 | |61 |6| 2| 5.679106| 1.0242836| 5.988099| 1.690359 | |49 |5| 2| 5.949659| 0.8942338| 6.257268| 1.810588 |

***Table X*** Optimal Hardware for RF under a strict data constraint of 45 training examples.

| Configuration Number | CPUs | RAM | Seconds | Dollars | Mean Distance | Distance SD | | :————- | :————- | :————- | :————- | :————- | :————- | | :————- | |81| 7| 16| 1.796767| 2.142429| 3.580441| 2.695402| |93| 8| 16| 1.796767| 2.448490| 3.888889| 2.927606| |21 |2| 16| 3.274177| 1.115447| 4.152403| 2.109742| |105| 9 |16| 1.796767| 2.754551| 4.211241| 3.170276| |117| 10| 16| 1.796767| 3.060613| 4.544539| 3.421187| |33 |3 |16| 3.380755| 1.727634| 4.577421| 2.314046| |45| 4| 16| 3.380755| 2.303511| 4.932269| 2.493434| |57| 5| 16| 3.211710| 2.735413| 5.080262| 2.563458| |69| 6| 16| 3.211710| 3.282496| 5.530223| 2.790505|

***Table X*** Optimal Hardware for MARS under a strict data constraint of 45 training examples.

The data-constrained optimals showed significant overlap with the unconstrained optimal predictions. One major difference between the two, however, is the decreased costs, in both time and money, of running the constrained experiment, since less data is used to fit the experiment.

In the case of MARS, we see that number of CPU cores is clearly not a big factor in the optimality of the solution, since everywhere between 2 and 10 CPU cores shows a statistically identical distribution. Again, we see that the memory requirement is large compared to the otehr SDM classes, however, this is likely still due to sampling design and unlikely to be a real recommendation. GAM suggested the same optimal set of 2, 3, or 4 cores with 2 GB of RAM. This makes sense that the GAM optimal is the same as from the unconstrained. There is very little hardware influence on the run time of the model, which predicts an unconstrained optimal will use the lowest amount of hardware. So, any experiment that requires less time than the unconstrained optimal would thus also need only the minimum amount of ahrdware. XXX about GBM-BRT. RF showed wide variability between the number of cores that would result in the optimal solution as well. It is likely to involve multiple cores, however, there is high uncertainty about the exact number required. Nevertheless, only two GB of RAM are suggested regardless of the number of CPU cores. This high degree of uncertainty is surprising, since the model shows high predictive skill of the test set, it would appear that the model itself should be able to differentiate between the different number of cores.

##### Cost-Constrained Optimization

A second type of constraint on the hardware configuration choice is a cost-limit or time-limit imposed by the user. In this case, the user has identified a limit that should not be surpassed, and a solution that falls within it should then be identified. In this particular example, the user has an unlimited supply of data with which to fit the model. However, the techniques from this section and the section above could be applied together to create a real-world situation in which both cost and and data are limited. In this example, the user has a limit of 22 seconds in which to run the model. This scenario could occur in a centralized web-based modeling application, in which the application requires latency to be below a certain time, so that user’s don’t get lost or distracted.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Fixed Accuracy | Training Examples | covariates |
| GAM | 0.71311 | 9000 | 5 |
| GBM-BRT | 0.7001 | 110 | 5 |
| MARS | 0.7016 | 130 | 5 |
| RF | 0.8136 | 1951 | 4 |

#### 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.

ARAUJO, M., & NEW, M. (2007). Ensemble forecasting of species distributions. *Trends in Ecology & Evolution*, *22*(1), 42–47.

Araújo, M. B., & Guisan, A. (2006). Five (or so) challenges for species distribution modelling. *Journal of Biogeography*, *33*(10), 1677–1688.

Araújo, M. B., Cabeza, M., Thuiller, W., Hannah, L., & Williams, P. H. (2004). Would climate change drive species out of reserves? An assessment of existing reserve-selection methods. *Global Change Biology*, *10*(9), 1618–1626.

Araújo, M. B., Pearson, R. G., Thuiller, W., & Erhard, M. (2005). Validation of species-climate impact models under climate change. *Global Change Biology*, *11*(9), 1504–1513.

Araújo, M. B., Whittaker, R. J., Ladle, R. J., & Erhard, M. (2005). Reducing uncertainty in projections of extinction risk from climate change. *Global Ecology and Biogeography*, *14*(6), 529–538.

Armbrust, M. (2009). Above the Clouds: A Berkeley View of Cloud Computing, 1–25.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., & Katz, R. H. (2009). Above the clouds: A berkeley view of cloud computing.

Austin, M. (2007). Species distribution models and ecological theory: A critical assessment and some possible new approaches. *Ecological Modelling*, *200*(1-2), 1–19.

Austin, M. P. (2002). Spatial prediction of species distribution: an interface between ecological theory and statistical modelling. *Ecological Modelling*, *157*(2-3), 1–18.

Barnosky, A. D., Hadly, E. A., Bascompte, J., Berlow, E. L., Brown, J. H., Fortelius, M., Getz, W. M., et al. (2012). Approaching a state shift in Earths biosphere. *Nature*, *486*(7401), 52–58.

Beck, J., Böller, M., Erhardt, A., & Schwanghart, W. (2014). Spatial bias in the GBIF database and its effect on modeling species’ geographic distributions. *Ecological Informatics*, *19*(C), 10–15.

Bifet, A., Holmes, G., Pfahringer, B., & Gavalda, R. (2011). Detecting Sentiment Change in Twitter Streaming Data. *WAPA*.

Blaauw, M. (2010). Methods and code for “classical” age-modelling of radiocarbon sequences. *Quaternary Geochronology*, *5*(5), 512–518.

Bleich, J., & Kapelner, A. (2016). Package bartMachine, 1–50.

Bolker, B. M., Brooks, M. E., Clark, C. J., Geange, S. W., Poulsen, J. R., Stevens, M. H. H., & White, J.-S. S. (2009a). Generalized linear mixed models: a practical guide for ecology and evolution. *Trends in Ecology & Evolution*, *24*(3), 127–135.

Bolker, B. M., Brooks, M. E., Clark, C. J., Geange, S. W., Poulsen, J. R., Stevens, M. H. H., & White, J.-S. S. (2009b). Generalized linear mixed models: a practical guide for ecology and evolution. *Trends in Ecology & Evolution*, *24*(3), 127–135.

Breiman, L. (2006). randomForest: Breiman and Cutler’s random forests for classification and regression.

Brewer, E. A., & Brewer, E. A. (1995). *High-level optimization via automated statistical modeling* (Vol. 30). New York, New York, USA: ACM.

Brewer, S., Jackson, S. T., & Williams, J. W. (2012). Paleoecoinformatics: applying geohistorical data to ecological questions. *Trends in Ecology & Evolution*, *27*(2), 104–112.

Candela, L., Castelli, D., Coro, G., Pagano, P., & Sinibaldi, F. (2013). Species distribution modeling in the cloud. *Concurrency and Computation: Practice and Experience*, *28*(4), 1056–1079.

Cannon, A. R., & John, C. H. S. (2007). Measuring Empirical Computational Complexity. *Organizational Research Methods*, *10*(2), 1–10.

Carroll, J. M. (1999). Five Reasons for Scenario-based Design. *HICSS 1999*, *Track3*, 11.

Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications*, *19*(2), 171–209.

Clark, J. S., Gelfand, A. E., Woodall, C. W., & Zhu, K. (2014). More than the sum of the parts: forest climate response from joint species distribution models. *Ecological Applications*, *24*(5), 990–999.

Cormen, T. H. (2009). *Introduction to Algorithms*. MIT Press.

Davis, M. B. (1963). On the theory of pollen analysis. *American Journal of Science*, *261*(10), 897–912.

Dawson, A., Paciorek, C., McLachlan, J., Goring, S., WIlliams, J., & Jackson, S. T. (2016). Quantifying pollen-vegetation relationships to reconstruct ancient forests using 19th-century forest composition and pollen data, 1–76.

Dongarra, J., Martin, J. L., & Worlton, J. (1987). Computer benchmarking: Paths and pitfalls: The most popular way of rating computer performance can confuse as well as inform; avoid misunderstanding by asking just what the benchmark is measuring. *IEEE Spectrum*, *24*(7), 38–43.

Dormann, C. F., Schymanski, S. J., Cabral, J., Chuine, I., Graham, C., Hartig, F., Kearney, M., et al. (2012). Correlation and process in species distribution models: bridging a dichotomy. *Journal of Biogeography*, *39*(12), 2119–2131.

DRAKE, J. M., RANDIN, C., & Guisan, A. (2006). Modelling ecological niches with support vector machines. *Journal of Applied Ecology*, *43*(3), 424–432.

Elith, J., & Graham, C. H. (2009). Do they? How do they? WHY do they differ? On finding reasons for differing performances of species distribution models. *Ecography*, *32*(1), 66–77.

Elith, J., & Leathwick, J. R. (2009a). Species Distribution Models: Ecological Explanation and Prediction Across Space and Time. *Annual Review of Ecology, Evolution, and Systematics*, *40*(1), 677–697.

Elith, J., & Leathwick, J. R. (2009b). Species Distribution Models: Ecological Explanation and Prediction Across Space and Time. *Annual Review of Ecology, Evolution, and Systematics*, *40*(1), 677–697.

Elith, J., H Graham, C., P Anderson, R., Dudík, M., Ferrier, S., Guisan, A., J Hijmans, R., et al. (2006). Novel methods improve prediction of species distributions from occurrence data. *Ecography*, *29*(2), 129–151.

Elith, J., Leathwick, J. R., & Hastie, T. (2008). A working guide to boosted regression trees. *Journal of Animal Ecology*, *77*(4), 802–813.

Elith, J., Phillips, S. J., Hastie, T., Dudík, M., Chee, Y. E., & Yates, C. J. (2010). A statistical explanation of MaxEnt for ecologists. *Diversity and Distributions*, *17*(1), 43–57.

Ellison, A. M. (2004). Bayesian inference in ecology. *Ecology Letters*, *7*(6), 509–520.

Ficetola, G. F., Thuiller, W., & Miaud, C. (2007). Prediction and validation of the potential global distribution of a problematic alien invasive species - the American bullfrog. *Diversity and Distributions*, *13*(4), 476–485.

Fink, E. (1998). How to Solve It Automatically: Selection Among Problem Solving Methods. *AIPS*.

Fitzpatrick, M. C., Gotelli, N. J., & Ellison, A. M. (2013). MaxEnt versus MaxLike: empirical comparisons with ant species distributions. *Ecosphere*, *4*(5), art55–15.

Fløjgaard, C., Normand, S., & Skov, F. (2009). Ice age distributions of European small mammals: insights from species distribution modelling. *Journal of Biogeography*, *36*(6), 1152–1163.

Foster, I., Zhao, Y., Raicu, I., & Lu, S. (2008). Cloud computing and grid computing 360-degree compared. *2008 Grid Computing …*, 1–10.

Franklin, J. (2010). *Mapping Species Distributions*. Spatial inference and prediction (Vol. 44). Cambridge University Press.

Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of statistics*.

Fritz, S. A., Schnitzler, J., Eronen, J. T., Hof, C., Böhning-Gaese, K., & Graham, C. H. (2013). Diversity in time and space: wanted dead and alive. *Trends in Ecology & Evolution*, *28*(9), 509–516.

Glew, J. R., Smol, J. P., & Last, W. M. (2002). Sediment Core Collection and Extrusion. In *Tracking environmental change using lake sediments* (pp. 73–105). Dordrecht: Springer Netherlands.

Golding, N., & Purse, B. V. (2016). Fast and flexible Bayesian species distribution modelling using Gaussian processes. *Methods in Ecology and Evolution*, *7*(5), 598–608.

Goldsmith, S. F., Aiken, A. S., & Wilkerson, D. S. (2007). *Measuring empirical computational complexity*. New York, New York, USA: ACM.

Goring, S., Dawson, A., Simpson, G. L., Ram, K., Graham, R. W., Grimm, E. C., & Williams, J. W. (2015). neotoma: A Programmatic Interface to the Neotoma Paleoecological Database. *Open Quaternary*, *1*(6), 1–17.

Granell, C., Díaz, L., Schade, S., Ostländer, N., & Huerta, J. (2013). Enhancing integrated environmental modelling by designing resource-oriented interfaces. *Environmental Modelling and Software*, *39*(C), 229–246.

Grimm, E. C., Bradshaw, R. H. W., Brewer, S., Flantua, S., Glesecke, T., Lezine, A.-M., Takahara, H., et al. (2013). Databases and Their Application. *Encycolpedia of Quaternary Science*, 1–9.

Guisan, A., & Thuiller, W. (2005a). Predicting species distribution: offering more than simple habitat models. *Ecology Letters*, *8*(9), 993–1009.

Guisan, A., & Thuiller, W. (2005b). Predicting species distribution: offering more than simple habitat models. *Ecology Letters*, *8*(9), 993–1009.

Guisan, A., & Zimmerman, N. (2000). Predictive habitat distribution models in ecology. *Ecological Modelling*, *135*(2-3), 1–40.

Guisan, A., Edwards, T. C., & Hastie, T. (2002). Generalized linear and generalized additive models in studies of species distributions: setting the scene. *Ecological Modelling*, *157*(2-3), 89–100.

Guisan, A., Lehmann, A., Ferrier, S., Austin, M., OVERTON, J. M. C., ASPINALL, R., & Hastie, T. (2006). Making better biogeographical predictions of species’ distributions. *Journal of Applied Ecology*, *43*(3), 386–392.

Guisan, A., Tingley, R., Baumgartner, J. B., Naujokaitis-Lewis, I., Sutcliffe, P. R., Tulloch, A. I. T., Regan, T. J., et al. (2013). Predicting species distributions for conservation decisions. *Ecology Letters*, *16*(12), 1424–1435.

Gustafson, J. L. (1988). Reevaluating Amdahl’s law. *Communications of the ACM*, *31*(5), 532–533.

Hamann, A., & Wang, T. (2006). POTENTIAL EFFECTS OF CLIMATE CHANGE ON ECOSYSTEM AND TREE SPECIES DISTRIBUTION IN BRITISH COLUMBIA. *Ecology*, *87*(11), 2773–2786.

Hampton, S. E., Strasser, C. A., Tewksbury, J. J., Gram, W. K., Budden, A. E., Batcheller, A. L., Duke, C. S., et al. (2013). Big data and the future of ecology. *Frontiers in Ecology and the Environment*, *11*(3), 156–162.

Hassan, Q. (2011). Demystifying Cloud Computing. *CrossTalk*, 16–21.

Hastie, S. M. D. from mda mars by T., & wrapper, R. T. U. A. M. F. utilities with T. L. leaps. (2016). *earth: Multivariate Adaptive Regression Splines*.

Hastie, T. (2015). *gam: Generalized Additive Models*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition. *International Statistical Review*, *77*(3), 482–482.

Hegel, T. M., Cushman, S. A., Evans, J., & Huettmann, F. (2010). Current State of the Art for Statistical Modelling of Species Distributions. In *Spatial complexity, informatics, and wildlife conservation* (pp. 273–311). Tokyo: Springer Japan.

Hobbie, J. E., Carpenter, S. R., Grimm, N. B., Gosz, J. R., & Seastedt, T. R. (2003). The US Long Term Ecological Research Program. *BioScience*, *53*(1), 21–32.

Hsu, C.-H., Lin, C.-Y., Ouyang, M., & Guo, Y. K. (2013). Biocloud: Cloud Computing for Biological, Genomics, and Drug Design. *BioMed Research International*, *2013*, 1–3.

Huang, Q., Yang, C., Liu, K., Xia, J., Xu, C., Li, J., Gui, Z., et al. (2013). Evaluating open-source cloud computing solutions for geosciences. *Computers and Geosciences*, *59*(C), 41–52.

Huang, Q., Yang, C., Nebert, D., Liu, K., & Wu, H. (2010). *Cloud computing for geosciences: deployment of GEOSS clearinghouse on Amazon’s EC2*. Deployment of geoss clearinghouse on amazon’s ec2. New York, New York, USA: ACM.

Hutchinson, G. E. (1957). Concluding Remarks. *Cold Spring Harbor Symposia on Quantitative Biology*, *22*(0), 1–7.

Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2014). Identifying Key Algorithm Parameters and Instance Features using Forward Selection. *High Performance Computing for Computational Science - VECPAR 2012*, *7997*(Chapter 40), 1–15.

Hutter, F., Xu, L., Hoos, H. H., & Leyton-Brown, K. (2014). Algorithm runtime prediction: Methods & evaluation. *Artificial Intelligence*, *206*(C), 79–111.

Issa, S. A., Kienzler, R., El-Kalioby, M., Tonellato, P. J., Wall, D., Bruggmann, R., & Abouelhoda, M. (2013). Streaming Support for Data Intensive Cloud-Based Sequence Analysis. *BioMed Research International*, *2013*(8), 1–16.

Johnson, K. (2012). Evaluating the Design of the R Language. *Designing Language Teaching Tasks*, (Chapter 8), 1–27.

Jones, R., & Kalibera, T. (2013). *Rigorous benchmarking in reasonable time* (Vol. 48). ACM.

Kaján, L., Yachdav, G., Vicedo, E., Steinegger, M., Mirdita, M., Angermüller, C., Böhm, A., et al. (2013). Cloud Prediction of Protein Structure and Function with PredictProtein for Debian. *BioMed Research International*, *2013*(3), 1–6.

Kapelner, A., & Bleich, J. (2016). bartMachine: Machine Learning with Bayesian Additive Regression Trees, 1–40.

Keppel, G., Van Niel, K. P., Wardell-Johnson, G. W., Yates, C. J., Byrne, M., Mucina, L., Schut, A. G. T., et al. (2011). Refugia: identifying and understanding safe havens for biodiversity under climate change. *Global Ecology and Biogeography*, *21*(4), 393–404.

Knuth, D. E. (1976). Big Omicron and big Omega and big Theta. *ACM Sigact News*, *8*(2), 18–24.

Kogan, J. (2014). Feature Selection Over Distributed Data Streams. In *Data mining for service* (pp. 11–26). Berlin, Heidelberg: Springer Berlin Heidelberg.

Kundra, V. (2010). 25 Point Implementation Plan to Reform Federal information Technology Management.

Leathwick, J. R., Elith, J., & Hastie, T. (2006). Comparative performance of generalized additive models and multivariate adaptive regression splines for statistical modelling of species distributions. *Ecological Modelling*, *199*(2), 188–196.

Leyton-Brown, K., Nudelman, E., & Andrew, G. (2003). A portfolio approach to algorithm selection. *IJCAI*.

Liaw, A., & Wiener, M. (2002). Classification and Regression by randomForest. *R News*, *2*(3), 18–22.

Lilja, D. (2009). **Measuring Computer Performance: A Practitioner’s Guide**. Cambridge: Cambridge University Press.

Lobo, J. M., Jiménez-Valverde, A., & Real, R. (2008). AUC: a misleading measure of the performance of predictive distribution models. *Global Ecology and Biogeography*, *17*(2), 145–151.

LOISELLE, B. A., HOWELL, C. A., Graham, C. H., GOERCK, J. M., BROOKS, T., SMITH, K. G., & Williams, P. H. (2003). Avoiding Pitfalls of Using Species Distribution Models in Conservation Planning. *Conservation …*, *17*(6), 1591–1600.

Lorenz, D. J., Nieto-Lugilde, D., Blois, J. L., Fitzpatrick, M. C., & Williams, J. W. (2016). Downscaled and debiased climate simulations for North America from 21,000 years ago to 2100AD. *Scientific Data*, *3*, 160048–19.

Lorenz, D. J., Nieto-Lugilde, D., Blois, J. L., Fitzpatrick, M. C., & Williams, J. W. (2016). *Data from: Downscaled and debiased climate simulations for North America from 21,000 years ago to 2100AD*.

Lowe, C. B., Kellis, M., Siepel, A., Raney, B. J., Clamp, M., Salama, S. R., Kingsley, D. M., et al. (2011). Rapid Range Shifts of Species Associated with High Levels of Climate Warming. *Science*, *333*(6045), 1019–1024.

Lu, S., Li, R. M., Tjhi, W. C., Lee, K. K., Wang, L., Li, X., & Ma, D. (2011). A Framework for Cloud-Based Large-Scale Data Analytics and Visualization: Case Study on Multiscale Climate Data. In *2011 ieee 3rd international conference on cloud computing technology and science (cloudcom)* (pp. 618–622). IEEE.

Maguire, K. C., Nieto-Lugilde, D., Fitzpatrick, M. C., Williams, J. W., & Blois, J. L. (2015). Modeling Species and Community Responses to Past, Present, and Future Episodes of Climatic and Ecological Change. *Annual Review of Ecology, Evolution, and Systematics*, *46*(1), 343–368.

Manyika, J., Chui, M., Brown, B., Bughin, J., & Dobbs, R. (2015). *Big data: The next frontier for innovation, competition, and productivity. 2011*. URL http://www. mckinsey. ….

Mell, P. M., & Grance, T. (2012). The NIST definition of cloud computing. *National Institude of Standards and Technology*, 1–7.

Michener, W. K., & Jones, M. B. (2012). Ecoinformatics: supporting ecology as a data-intensive science. *Trends in Ecology & Evolution*, *27*(2), 85–93.

Miller, J., Franklin, J., & ASPINALL, R. (2007). Incorporating spatial dependence in predictive vegetation models. *Ecological Modelling*, *202*(3-4), 225–242.

Mosco, V. (2014). To the Cloud: Big Data in a Turbulent World. *Popular Communication*, *12*(4), 271–269.

Moss, R. H., Edmonds, J. A., Hibbard, K. A., Manning, M. R., Rose, S. K., Vuuren, D. P. van, Carter, T. R., et al. (2010). The next generation of scenarios for climate change research and assessment. *Nature*, *463*(7282), 747–756.

NASAS PROGRESS IN ADOPTING CLOUD-COMPUTING TECHNOLOGIES. (2013)., 1–38.

Natekin, A. (2013). Gradient boosting machines, a tutorial. *Frontiers in Neurorobotics*, *7*, 1–21.

National Science Board (U.S.). (2016, January). Science & engineering indicators.

Nieto-Lugilde, D., Maguire, K. C., Blois, J. L., Williams, J. W., & Fitzpatrick, M. C. (2015). Close agreement between pollen-based and forest inventory-based models of vegetation turnover. *Global Ecology and Biogeography*, *24*(8), 905–916.

Nogués-Bravo, D. (2009). Predicting the past distribution of species climatic niches. *Global Ecology and Biogeography*, *18*(5), 521–531.

Nogués-Bravo, D., Rodríguez, J., Hortal, J., & Batra, P. (2008). Climate change, humans, and the extinction of the woolly mammoth. *PLoS Biology*, *6*(4), e79.

Nordhaus, W. D., & Yale University. Cowles Foundation for Research in Economics. (2001). The Progress of Computing.

Norman, D. A. (1984). Stages and levels in human-machine interaction. *International journal of man-machine studies*, *21*(4), 365–375.

Obrien, R. M. (2007). A Caution Regarding Rules of Thumb for Variance Inflation Factors. *Quality & Quantity*, *41*(5), 673–690.

O’Donnell, M. S., & Ignizio, D. A. (2012). Bioclimatic predictors for supporting ecological applications in the conterminous United States. *US Geological Survey Data Series*.

Package paleobioDB. (2016)., 1–29.

Package rgbif. (2016)., 1–79.

Pearman, P. B., Guisan, A., Broennimann, O., & Randin, C. F. (2008). Niche dynamics in space and time. *Trends in Ecology & Evolution*, *23*(3), 149–158.

Peterson, A. T. (2003). Predicting the geography of species’ invasions via ecological niche modeling. *The quarterly review of biology*, *78*(4), 419–433.

Peterson, A. T., Soberon, J., & Sánchez-Cordero, V. (1999). Conservatism of Ecological Niches in Evolutionary Time. *Science*, *285*(5431), 1265–1267.

Phillips, S. J., & Dudík, M. (2008). Modeling of species distributions with Maxent: new extensions and a comprehensive evaluation. *Ecography*, *0*(0), 1–15.

Phillips, S. J., Anderson, R. P., & Schapire, R. E. (2006). Maximum entropy modeling of species geographic distributions. *Ecological Modelling*, *190*(3-4), 231–259.

R Core Team. (2016). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.

Rezende, V. L., Oliveira-Filho, A. T., Eisenlohr, P. V., Kamino, L. H. Y., & Vibrans, A. C. (2015). Restricted geographic distribution of tree species calls for urgent conservation efforts in the Subtropical Atlantic Forest. *Biodiversity and Conservation*, 1–15.

Riahi, K., Rao, S., Krey, V., Cho, C., Chirkov, V., Fischer, G., Kindermann, G., et al. (2011). RCP 8.5A scenario of comparatively high greenhouse gas emissions. *Climatic Change*, *109*(1-2), 33–57.

Root, T. L., & MacMynowski, D. P. (2005). Human-modified temperatures induce species changes: joint attribution. In *Proceedings of the national academy of sciences* (pp. 7465–7469).

Rosson, M. B. (2002). Scenario-Based Design. In J. Jacko & A. Sears (Eds.), *The human-computer interaction handbook fundamentals, evolving technology and emerging applications* (pp. 1032–1050). CRC Press.

Roth, R. E. (2013). Interactive maps: What we know and what we need to know. *Journal of Spatial Information Science*, (6), 1–57.

Sadjadi, S. M., Shimizu, S., Figueroa, J., Rangaswami, R., Delgado, J., Duran, H. A., & Collazo-Mojica, X. J. (2008). A modeling approach for estimating execution time of long-running scientific applications. *IPDPS*, 1–8.

Salisbury, E. J. (1926). The Geographical Distribution of Plants in Relation to Climatic Factors. *The Geographical Journal*, *67*(4), 312.

Schaeffer, M., Pierre, S. S., Twigger, S., White, O., & Rhee, S. Y. (2008). *The future of biocuration* (Vol. 455). Nature.

Schatz, M. C., Langmead, B., & Salzberg, S. L. (2010). Cloud computing and the DNA data race. *Nature Biotechnology*, *28*(7), 691–693.

Schimel, D., Keller, M., Duffy, P., Alves, L., & Aulenbach, S. (2009). *The NEON strategy: Enabling continental scale ecological forecasting*. NEON Inc.

Schnase, J. L. (2015). **CLIMATE ANALYTICS AS A SERVICE**, 1–4.

Schnase, J. L., Duffy, D. Q., Tamkin, G. S., Nadeau, D., Thompson, J. H., Grieg, C. M., McInerney, M. A., et al. (2014). MERRA Analytic Services: Meeting the Big Data challenges of climate science through cloud-enabled Climate Analytics-as-a-Service. *Computers, Environment and Urban Systems*, 1–14.

Smith, S. E., Mendoza, M. G., Zúñiga, G., Halbrook, K., Hayes, J. L., & Byrne, D. N. (2013). Predicting the distribution of a novel bark beetle and its pine hosts under future climate conditions. *Agricultural and Forest Entomology*, *15*(2), 212–226.

Snijders, C., Matzat, U., & Reips, U.-D. (2012). *“Big Data” : Big Gaps of Knowledge in the Field of Internet Science* (Vol. 7).

Soberon, J., & Peterson, T. (2004). Biodiversity informatics: managing and applying primary biodiversity data. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *359*(1444), 689–698.

Soberón, J., & Peterson, A. T. (2005). Interpretation of Models of Fundamental Ecological Niches and Species Distributional Areas. *Biodiversity Informatics*, *2*(0).

Soberón, J., Arriaga, L., & Lara, L. (2002). Issues of quality control in large, mixed-origin entomological databases. *Towards a global biological information infrastructure*, *70*, 15–22.

Souza Muñoz, M. E. de, De Giovanni, R., Siqueira, M. F. de, Sutton, T., Brewer, P., Pereira, R. S., Canhos, D. A. L., et al. (2009). openModeller: a generic approach to species potential distribution modelling. *GeoInformatica*, *15*(1), 111–135.

Stein, A. F., Isakov, V., Godowitch, J., & Draxler, R. R. (2007). A hybrid modeling approach to resolve pollutant concentrations in an urban area. *Atmospheric Environment*, *41*(40), 9410–9426.

Stein, L. D. (2010). The case for cloud computing in genome informatics. *Genome biology*, *11*(5), 1–7.

Sun, K., & Li, Y. (2013). Effort Estimation in Cloud Migration Process. In *2013 ieee 7th international symposium on service oriented system engineering (sose 2013)* (pp. 84–91). IEEE.

Svenning, J.-C., Fløjgaard, C., Marske, K. A., Nogués-Bravo, D., & Normand, S. (2011). Applications of species distribution modeling to paleobiology. *Quaternary Science Reviews*, *30*(21-22), 2930–2947.

Svenning, J.-C., Normand, S., & Skov, F. (2008). Postglacial dispersal limitation of widespread forest plant species in nemoral Europe. *Ecography*, *31*(3), 1–11.

Swets, J. A. (1988). Measuring the accuracy of diagnostic systems. *Science*, *240*(4857), 1285–1293.

Thomas, C. D. (2010). Climate, climate change and range boundaries. *Diversity and Distributions*, *16*(3), 488–495.

Thuiller, W. (2007). Biodiversity: Climate change and the ecologist. *Nature*, *448*(7153), 550–552.

Thuiller, W., Albert, C., Araújo, M. B., Berry, P. M., Cabeza, M., Guisan, A., Hickler, T., et al. (2008a). Predicting global change impacts on plant species distributions: Future challenges. *Perspectives in Plant Ecology, Evolution and Systematics*, *9*(3-4), 137–152.

Thuiller, W., Albert, C., Araújo, M. B., Berry, P. M., Cabeza, M., Guisan, A., Hickler, T., et al. (2008b). Predicting global change impacts on plant species distributions: Future challenges. *Perspectives in Plant Ecology, Evolution and Systematics*, *9*(3-4), 137–152.

Thuiller, W., Lavorel, S., & Araújo, M. B. (2005). Climate change threats to plant diversity in Europe. In *Proceedings of the national academy of sciences* (pp. 8245–8250).

Tyree, S., Weinberger, K. Q., Agrawal, K., & Paykin, J. (2011). *Parallel boosted regression trees for web search ranking*. New York, New York, USA: ACM.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., & Lindner, M. (2008). A break in the clouds. *ACM SIGCOMM Computer Communication Review*, *39*(1), 50–56.

Václavík, T., & Meentemeyer, R. K. (2009). Invasive species distribution modeling (iSDM): Are absence data and dispersal constraints needed to predict actual distributions? *Ecological Modelling*, *220*(23), 3248–3258.

Veloz, S. D., Williams, J. W., Blois, J. L., He, F., Otto-Bliesner, B., & Liu, Z. (2012). No-analog climates and shifting realized niches during the late quaternary: implications for 21st-century predictions by species distribution models. *Global Change Biology*, *18*(5), 1698–1713.

Vieilledent, G., Latimer, A. M., Gelfand, A. E., & Merow, C. (2012). *hSDM: hierarchical Bayesian species distribution models*. R package version.

Vincent, P. J., & Haworth, J. M. (1983). Poisson Regression Models of Species Abundance. *Journal of Biogeography*, *10*(2), 153–160.

Waltari, E., Hijmans, R. J., Peterson, A. T., Nyári, Á. S., Perkins, S. L., & Guralnick, R. P. (2007). Locating Pleistocene Refugia: Comparing Phylogeographic and Ecological Niche Model Predictions. *PLoS ONE*, *2*(7), e563–11.

Williams, J. W., & Jackson, S. T. (2007). Novel climates, no-analog communities, and ecological surprises. *Frontiers in Ecology and the Environment*, *5*(9), 475–482.

Wing, M. G., Eklund, A., & Kellogg, L. D. (2005). Consumer-grade global positioning system (GPS) accuracy and reliability. *Journal of forestry*.

Woodward, F. I. (1987). Climate and plant distribution.

Wu, Q., & Datla, V. V. (2011). On Performance Modeling and Prediction in Support of Scientific Workflow Optimization. In *2011 ieee world congress on services (services)* (pp. 161–168). IEEE.

Yang, C., & Huang, Q. (2013). *Spatial Cloud Computing*. A practical approach. CRC Press.

Yang, C., Goodchild, M., Huang, Q., Nebert, D., Raskin, R., Xu, Y., Bambacus, M., et al. (2011). Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? *International Journal of Digital Earth*, *4*(4), 305–329.

Yang, C., Wu, H., Huang, Q., Li, Z., & Li, J. (2011). Using spatial principles to optimize distributed computing for enabling the physical science discoveries. *Proceedings of the National Academy of Sciences*, *108*(14), 5498–5503.

Yee, T. W., & Mitchell, N. D. (1991). Generalized additive models in plant ecology. *Journal of vegetation science*, *2*(5), 587–602.

1. https://aws.amazon.com/government-education/research-and-technical-computing/. Accessed 18 September, 2016. [↑](#footnote-ref-30)
2. http://www.nsf.gov/pubs/2013/nsf13602/nsf13602.htm [↑](#footnote-ref-31)