

University of Montana

## ScholarWorks at University of Montana

---

Graduate Student Theses, Dissertations, &  
Professional Papers

Graduate School

---

2019

# ESTIMATES OF FOREST CHARACTERISTICS DERIVED FROM REMOTELY SENSED IMAGERY AND FIELD SAMPLES: APPLICABLE SCALES, APPROPRIATE STUDY DESIGN, AND RELEVANCE TO FOREST MANAGEMENT

John S. Hogland

Follow this and additional works at: <https://scholarworks.umt.edu/etd>

**Let us know how access to this document benefits you.**

---

### Recommended Citation

Hogland, John S., "ESTIMATES OF FOREST CHARACTERISTICS DERIVED FROM REMOTELY SENSED IMAGERY AND FIELD SAMPLES: APPLICABLE SCALES, APPROPRIATE STUDY DESIGN, AND RELEVANCE TO FOREST MANAGEMENT" (2019). *Graduate Student Theses, Dissertations, & Professional Papers*. 11505.

<https://scholarworks.umt.edu/etd/11505>

This Dissertation is brought to you for free and open access by the Graduate School at ScholarWorks at University of Montana. It has been accepted for inclusion in Graduate Student Theses, Dissertations, & Professional Papers by an authorized administrator of ScholarWorks at University of Montana. For more information, please contact [scholarworks@mso.umt.edu](mailto:scholarworks@mso.umt.edu).

ESTIMATES OF FOREST CHARACTERISTICS DERIVED FROM REMOTELY  
SENSED IMAGERY AND FIELD SAMPLES: APPLICABLE SCALES,  
APPROPRIATE STUDY DESIGN, AND RELEVANCE TO FOREST  
MANAGEMENT

By

John S. Hogland

BS, Auburn University, Auburn, AL, USA, 2001  
MS, Auburn University, Auburn, AL, USA, 2005  
Dissertation

presented in partial fulfillment of the requirements  
for the degree of

Doctorate  
in Forest & Conservation Sciences

The University of Montana  
Missoula, MT

December 2019

Approved by:

Scott Whittenburg,  
Graduate School Dean

David L.R Affleck  
W.A. Franke College of Forestry & Conservation

Solomon Dobrowski  
W.A. Franke College of Forestry & Conservation

Carl Seielstad  
W.A. Franke College of Forestry & Conservation

Jon Graham  
Department of Mathematical Sciences

Robert Smith  
Department of Computer Science

Nathaniel M. Anderson  
USDA Forest Service, Rocky Mountain Research Station

© COPYRIGHT

by

John S. Hogland

2019

All Rights Reserved

Hogland, John, Doctorate, December 2019  
*Forest and Conservation Sciences*

ESTIMATES OF FOREST CHARACTERISTICS DERIVED FROM REMOTELY  
SENSED IMAGERY AND FIELD SAMPLES: APPLICABLE SCALES,  
APPROPRIATE STUDY DESIGN, AND RELEVANCE TO FOREST  
MANAGEMENT

Chairperson: David L.R. Affleck

**Abstract:** Information and knowledge about a given forested landscape drives forest management decisions. Within forest management though, information that adequately describes various characteristics of the forested environment in the spatial detail desired to make fully informed management decisions is often limited. Key metrics such as species composition, tree basal area, and tree density are typically too expensive to collect using ground-based inventory methods alone across broad extents for forest level planning (thousands of ha) at fine spatial detail that permit use at tactical spatial scales (tens of ha). However, quantifying these metrics accurately, in spatial detail, across broad landscapes is important to inform the management process. While relating remotely sensed data to classical ground-based survey data through modeling has shown promise for describing landscapes at the spatial detail need to inform planning and tactical scale projects, questions remain related to integrating both sources of data, sample design, and linking plots to remotely sensed data. This dissertation addresses critical aspects of these questions by: quantifying and mitigating the impact of co-registration errors; comparing various sample designs and estimation techniques using simulated ground-based information, remotely sensed data, and a variety of modeling techniques; developing enhanced image normalization routines; and creating an ensemble approach to estimating various forest characteristics that describe species composition, basal area, and tree density. This dissertation address knowledge gaps in the fields of forestry, remote sensing, data science, and decision science that can be used to efficiently and effectively inform the natural resource management decision-making process at fine spatial resolutions across broad extents.

## *Acknowledgments*

*I would like to thank David Affleck, Nathaniel Anderson, Carl Seielstad, Solomon Debrowski, Jon Graham, and Robert Smith for serving on my committee and providing mentorship throughout my Ph.D. I would also like to thank Jason Drake and Paul Medley for their encouragement and for their interest and dedication to longleaf pine conservation and management throughout the southeastern United States. Additionally, I would like to thank Melissa Reynolds-Hogland for her thoughts, comments, and suggestions which helped to significantly improve this dissertation and resulting articles. Finally, I would like to thank Melissa, Rose, Frank, Butch, Judy, and Dan for their never ending love, support, and personal sacrifice which gave me the encouragement and provided me the time to obtain a doctoral degree.*

## Table of Contents

Chapter 1 .....	1
Estimates of forest characteristics derived from remotely sensed imagery and field samples: applicable scales, appropriate study design, and relevance to forest management	
Chapter 2 .....	9
Mitigating the Impact of Field and Image Registration Errors through Spatial Aggregation	
Chapter 3 .....	31
Improving estimates of natural resources using model-based estimators: impacts of sample design, estimation technique, and strengths of association	
Chapter 4 .....	49
Estimating forest characteristics for longleaf pine restoration using normalized remotely sensed imagery in Florida USA	
Chapter 5 .....	70
Transforming data into information for natural resource decision making: Improving the utility of remote sensing products at tactical and planning scales	
Coding Libraries .....	78
Simulations & Statistical Analyses Libraries (R) .....	78
General Functions .....	79
Chapter 2 .....	99
Chapter 3 .....	102
Chapter 4 .....	121
Centering Plot Locations (Python) .....	146
Enhanced Aggregate No-Change Regression Library (C#) .....	150
ArcPad Library (Mobile Data Collection) .....	165
Supporting Material .....	209
Field Plot Protocols	

## Chapter 1

# Estimates of forest characteristics derived from remotely sensed imagery and field samples: applicable scales, appropriate study design, and relevance to forest management

**Abstract:** Accurate information is critical for effective management. Within forestry, key information related to forest characteristics used to inform management include stand metrics such as species composition, tree basal area ( m<sup>2</sup> per ha, BAH), and tree density (trees per ha, TPH). Quantifying those metrics accurately, in spatial detail, across broad landscapes is important to inform the management process. However, the acquisition of such information at fine spatial resolutions across large extents is cost prohibitive when only ground-based survey methods are utilized. In this dissertation, I describe and implement an alternative methodology to quantify forest metrics such as BAH and TPH at fine to medium spatial resolutions across large extents using remotely sensed data. From a theoretical perspective, I address issues of spatial scale, co-registration errors, ideal field sampling unit configurations, sample intensity and allocation, and use of derived BAH and TPH estimates. From an applied perspective, I focus on quantifying patterns of BAH and TPH across broad extents by relating field measurement to fine-grained remotely sensed data in the portion of northwest Florida, USA, known as the Florida Panhandle. The primary objectives of this dissertation are to address knowledge gaps in the fields of forestry, remote sensing, data science, and decision science which, once addressed, can be used efficiently and effectively to inform the natural resource management decision-making process at fine spatial resolutions across broad extents.

**Keywords:** basal area, trees density, co-registration, sample design, longleaf, forest characteristics

---

## 1. Introduction

Forest management is a complex integrated process that combines multiple objectives to accomplish a predefined set of goals as they relate to forested lands [1]. Since the United States National Forest Management Act of 1976, the federal definition of forest management has expanded well beyond timber management to include tenets of economic and social goals as components of management choices, the consideration of larger socially defined multiple use management problems, and the need to quantitatively justify forest management plans and decisions. This expansion in scope fundamentally changes not only what we manage for, but how we justify our forest management decisions; emphasizing the action and need for planning in a broad context in both spatial extent and contextual scope.

Across varying forests of differing ownership, complexity, size, and extent, forest plans guide management activities and steer silvicultural prescriptions to meet private, public, and more generally social objectives and goals. Effective planning and implementation of those plans requires knowledge of the biotic and abiotic condition of a forest as well as understanding of their interactions within the context of the objectives and goals defined for a given forest [2, 3]. To gain understanding of the existing structure and composition of forests, practitioners implement well established mensuration techniques [4]. Generally, these techniques can be described as aggregating a sample of field plots for a given geographic area to determine mean and variance estimates of forest characteristics within that geographic area. While these techniques are well described, they can be extremely expensive and problematic to implement at fine spatial resolutions across broad extents. Simultaneously, as the human population increases and more people rely on and move into forested areas, social questions related to the impacts of management activities on forest ecosystems, connectivity, sustainability,



water quality, esthetics, carbon, air quality, climate, and timber products markets become more important at finer spatial detail. Due to the cost associated with quantifying basic information used to describe many of these forest characteristics, often limited information is available to inform management decisions at the spatial scale of implementation.

For example, well-known inventory endeavors such as the Forest Inventory Analysis Program of the U.S. Forest Service [5] provide a wealth of data related to our nation's forests. However, the inferences that can be drawn using those data are applicable at regional spatial resolutions at best and provide little utility at the spatial resolution of a national or state forest. That is not to say these data are useless at these scales but instead to identify a mismatch between the intent and scope for which those data are collected and the needs of society to address spatially explicit questions pertaining to forest management. Owing to this discrepancy, forest managers must implement a more intensive sampling scheme for projects such as a timber cruise or sale. However, these endeavors tend to be inconsistent, vary in intensity and scope, and generally pertain to only small geographic areas (e.g., less than 1000 hectares), making the data collected incongruent with other inventory efforts and impractical to implement at broad extents.

To illustrate the financial limitations of intensive field plot inventories, it is helpful to look at the per plot costs of endeavors such as the FIA. The cost of collecting basic forest information using the FIA protocol has been estimated to be \$600 to \$1,240 per plot [6]. On the other end of the cost spectrum, timber cruises conducted primarily to estimate timber volume have plot costs as low as \$50 per plot [7]. Using this range of plot costs (\$50 to \$1,240), a 10% cruise for a forest of 100,000 hectares would require 100,000 plots, each with a radius of 11.3 m, costing between \$5 million (at \$60 / plot) and \$124 million (at \$1,240 / plot). Assuming a 10% cruise is sufficient to accurately represent the complexity of a given forest for planning and project implementation purposes, the cost for such an endeavor across 100,000 hectares is prohibitive. Due to this expense, forest practitioners often cannot describe the forest condition at fine spatial resolution across broad extents, but settle for coarse depictions that describe forest characteristics generally as totals or averages for defined areas. This decision further impacts the forest planning process by forcing managers to make general forest plans with high levels of uncertainty about the existing condition of the forest at fine spatial resolutions.

Forest traits such as species composition, spatial arrangement, basal area ( $\text{m}^2 \text{ha}^{-1}$ , BAH), and tree densities (trees  $\text{ha}^{-1}$ , TPH) as described within a classical inventory framework [4] are not by themselves expensive to collect at the spatial resolution of the plot. The expense associated with the classical inventory framework stems from the number of plots required to quantify stand characteristics based on the geographic boundary of a stand or strata. Specifically, the classical inventory approach splits a forest into many stands of similar composition, stocking, tree size, and age class, and then summarizes sample units (plots) within each stand to estimate a mean and variance of species BAH and TPH. BAH and TPH estimates are then used at the stand level to inform the forest planning process [1].

While this procedure can be applied in almost every situation, requires only plot data, and has been embraced within the forestry community, the method only produces estimates for the stand as a whole, typically requires a large sample size, and does not directly allow for additional sources of information. In instances where additional information is known about the forest, the classical approach has been expanded to include that information by grouping stands into like strata [4, 8]. Stratification aims to reduce sampling variation within like groups (i.e., the stratum), in turn reducing sampling intensity and cost to achieve a predefined level of accuracy. Within each stratum, plot data are summarized and mean and variance terms for a given variable are attributed to stands and pooled or combined in a weighted fashion to estimate an overall mean and variance for the forest as a whole.

In instances where supplementary information (e.g., remotely sensed data) about the population (e.g., BAH) is known and is correlated with the population variable of interest, regression can be employed to further increase the precision and efficiency of a given sample [4, 9]. Within this estimation framework [10], supplemental information can be categorical or continuous, tested for relevance with regard to minimizing variation, and used to estimate the strength of the relationship

between the response variable (e.g., BAH) and predictor variables (e.g. spectral values from imagery). While regression has been used by biometricians to develop many allometric equations [11], this technique has only recently been used on a limited basis to estimate key stand metrics such as species composition, BAH, and TPH for a forest. Historically, this may have been due to the availability, scale, and quality of supplemental information with regard to plots and stands within a forest. Today, however, there is a wealth of digital and remotely sensed data (e.g., [12-15]) that can be used to increase the precision of estimates of key stand metrics used to inform forest management, while simultaneously reducing sampling cost.

For many years, remotely sensed data have been used to explore our surroundings [16] and stratify the terrestrial environment in useful ways [17, 18]. With recent advancements in technology, mathematics, statistics, machine learning, and computer science, remotely sensed relationships between reflected portions of the electromagnetic spectrum and the earth's terrestrial surface have been documented and exploited to build a wide range of data products depicting terrestrial characteristics such as topography [19], land use and cover [20], vegetative indices [21], vegetation communities [22, 23], fire severity [24], land cover change [25], and temperature [26]. Spatially defining these terrestrial characteristics has elevated the importance of fields such as landscape ecology [27] in understanding the impacts of patterns within a forest as they relate to the landscape-scale functions and services they provide. Within the context of forest management, these concepts underlie the necessity of accurately quantifying not only general amounts or resources and the condition of the forest as a whole, but spatially depicting spatial variations in forest characteristics such as species BAH and TPH with a high degree of fidelity.

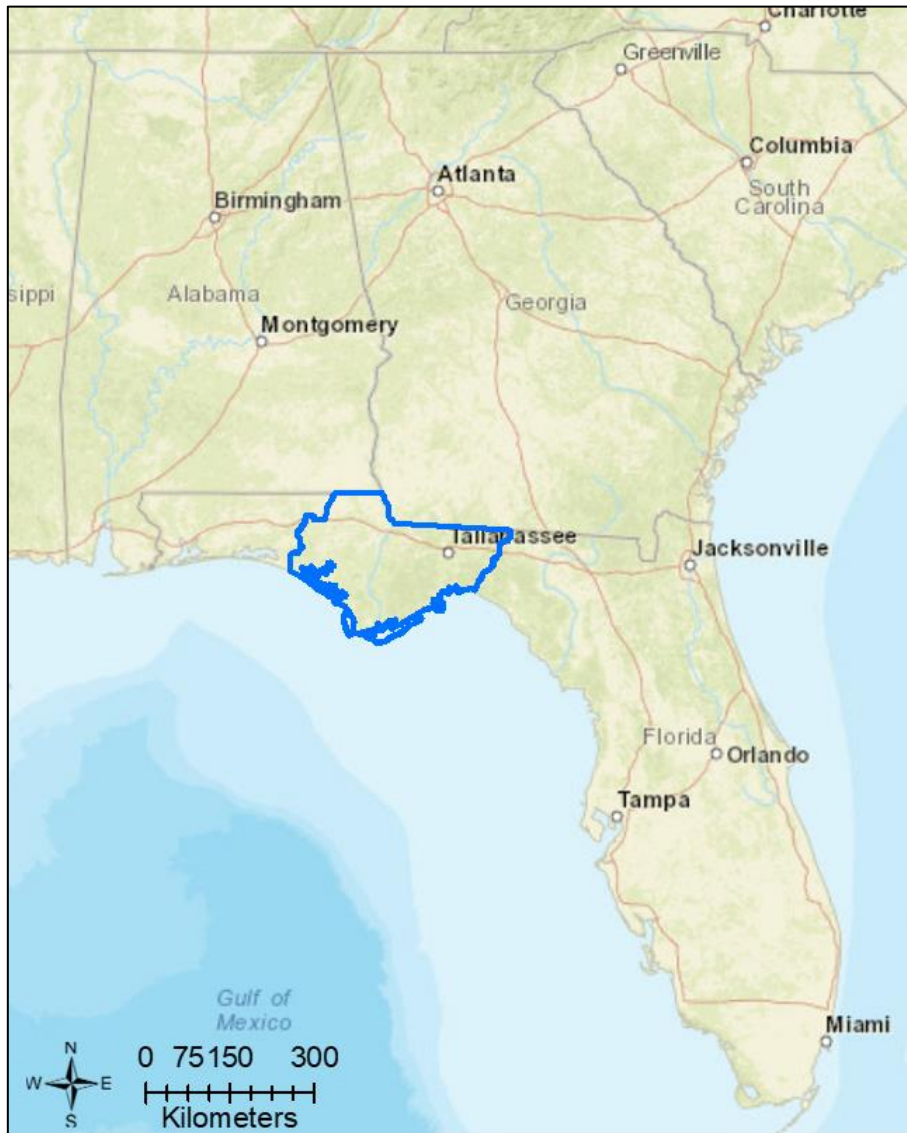
Given the benefits in sampling efficiency, precision of adopting regression techniques to estimate forest characteristics, and the wealth of supplemental remotely sensed data that are now available, it is surprising that regression has not been fundamentally adopted to quantify metrics such as BAH and TPH at the spatial scale of the plot, stand, and forest. Some reasons for this lack of adoption stem from practical limitations related to: 1) a lack of familiarity with remotely sensed data, 2) historically coarse spatial resolution of remotely sensed data, 3) technical challenges associated with modeling and processing data, 4) additional cost associated with the acquisition of remotely sensed data, and 5) lack of appropriate statistical techniques and associated strong statistical relationships among coarse remotely sensed data and traditional forestry metrics.

Despite some of these obstacles, field measured BAH and TPH have been successfully related to fine grained remotely sensed data, predictive models have been used to create surfaces that predict BAH and TPH continuously across forests at the spatial resolution of a plot, and those cell estimates have been successfully aggregated to stands and forests [28-35]. Moreover, using the spatially explicit outputs of this work, in collaboration with others, I have developed techniques to optimize a sustained yield across a 202,000 hectare forested landscape [36] and to estimate delivered costs and feedstock supply for more than 8 million hectares [37]. These examples demonstrate that the forest characteristics derived from linking field plots to remotely sensed data provide the baseline characterization of both stands and the forest needed to perform various fine resolution analyses in a spatially explicit manner.

## **2. Summary of the Chapter Contributions**

While many of the practical issues associated with using remotely sensed data are currently being addressed through education and outreach (e.g., [38-40]), the development of new fine-grained sensors (e.g., Sentinel II), and new processing techniques and software (e.g., [31, 41, 42]), unanswered questions remain related to scale, sample design, modeling approaches, and the utility of derived outputs for forest planning and management. In this dissertation, I address aspects of these issues from theoretical and applied perspectives using tenets of data and decision science. From a theoretical perspective in chapters 2 and 3, I quantify the impact of co-registration errors and describe how to minimizing their impacts through spatial aggregation and outline the benefits of sample designs that spread and balance sample observations across predictor variable space for various estimation

techniques. From an applied perspective, I describe, develop, and quantify the improvements in forest composition, BAH, and TPH estimates given a new enhanced image normalization methodology, ensemble general additive modeling approach, and various sources of remotely sensed data across broad extents for a case study in an area of northwest Florida known as the Apalachicola Significant Geographic Area (Figure 1). This area is the focus of an intensive landscape scale, cross-ownership hydrologic assessment and watershed management plan that includes restoration of longleaf pine (*Pinus palustris*) forests.



**Figure 1.** Location of Apalachicola Significant Geographic Area (blue polygon).

One of the primary objectives of this endeavor is to efficiently and effectively inform the natural resource management decision-making process by providing fine-grained descriptions of BAH and TPH patterns at less expense than traditional inventory approaches. From a research perspective, this dissertation helps move the disciplines of remote sensing, geography, forestry, landscape ecology, and data science forward by addressing knowledge gaps and solving theoretical and applied problems associated with: 1) scale as it relates to grain size of remotely sensed data and the field plot, 2) co-registration errors between GPS field plot locations and remotely sensed data, 3) plot size, layout, sample intensity, and plot allocation across the landscape to meet a defined level of precision,

4) the importance of image normalization, and 5) how sources of modeling error can be expressed and integrated into spatially explicit outputs and used within the decision-making process.

This dissertation consists of five chapters: this introductory chapter, three chapters in journal article format that are the bulk of the technical work, and a concluding communications chapter that synthesizes my findings and expresses how these results can be applied within decision science and forest management. Chapters two through four address the following topics: Chapter 2) Mitigating the impact of field and image registration errors through spatial aggregation, Chapter 3) Improving estimates of natural resources using model-based estimators: impacts of sample design, estimation technique, and strengths of association, and Chapter 4) Estimating forest characteristics for longleaf pine restoration using normalized fine and medium resolution remotely sensed imagery in Florida USA. In addition, this work contributes significantly to coding libraries that improve data collection, spatial analysis, and image processing that can be used to collect and store field data and efficiently quantify meaningful patterns in remotely sensed data. These libraries are included in supplemental materials for each article and can be used by data scientists to improve natural resource estimates and process larger amounts of data quickly and efficiently, providing the detailed information need to inform natural resource decision making.

## References

1. Davis, L.; Johnson, N. *Forest management, third edition*. McGraw Hill, New York, New York, **1987**, pp 790.
2. Kimmins J. *Forest ecology: a foundation for sustainable management*, Prentice Hall, Upper Saddle River, New Jersey, **1997**, pp 596.
3. Smith, D.; Larson, B.; Kelty, M; Ashton, M. *The practice of silviculture: applied forest ecology, ninth edition*, John Wiley & Sons, New York, New York, **1997**, pp 537
4. Avery T.; Burkhart H. *Forest measurements, fourth edition*. McGraw Hill, Boston, Massachusetts, **1994**, pp 408.
5. Birdsey, R.; Schreuder, H. An overview of forest inventory and analysis estimation procedures in the eastern United States -- With an emphasis on the components of change, Rep. No. RM-214. **1992** U.S.D.A., Forest Service, Rocky Mountain Forest and Range Experiment Station, Fort Collins, CO.
6. Grotefendt, R.; Schreuder, H. A new FIA-Type strategic inventory (NFI). In: Aguirre-Bravo, C.; Pellicane, Patrick J.; Burns, Denver P.; Draggan, S.E. Monitoring Science and Technology Symposium: Unifying Knowledge for Sustainability in the Western Hemisphere Proceedings RMRS-P-42CD. Fort Collins, CO: U.S. Department of Agriculture, Forest Service, Rocky Mountain Research Station. **2006** p. 790-798
7. Long, B. April 25. Interviewed by J. Hogland, 4/24/2018, University of Montana, Missoula, MT.
8. Gregoire, T.; Valentine, H. *Sampling Strategies for Natural Resources and the Environment*; Taylor & Francis Group, Boca Raton London New York, **2008**, pp 474
9. Neter, J.; Kutner, M.; Nachtsheim C.; Wasserman, W. *Applied linear statistical models fourth edition*, McGraw Hill, Boston, Massachusetts, **1996** pp 1408.
10. Rao, J; Molina, I. *Small area estimation, 2<sup>nd</sup> edition*. John Wiley & Sons, Inc., Hoboken, New Jersey, 2015, pp 480
11. Jenkins, J; Chojnacky, D; Heath, L; Birdsey, R. National-scale biomass estimation for United States tree species, *Forest Science*, **2003**, 49(1), 12-35.
12. Landsat. Landsat project description, **2018**, Available online: <https://landsat.usgs.gov/landsat-project-description>, (Accessed 4/27/2018).
13. National Agriculture Imagery Program [NAIP]. National Agriculture Imagery Program (NAIP) Information Sheet, **2012**, Available online: [http://www.fsa.usda.gov/Internet/FSA\\_File/naip\\_info\\_sheet\\_2013.pdf](http://www.fsa.usda.gov/Internet/FSA_File/naip_info_sheet_2013.pdf) (Accessed 5/14/2014).
14. Sentinel. ESA Sentinel online, **2018**, Available online: <https://sentinels.copernicus.eu/web/sentinel/home>, (Accessed 4/27/2018).
15. 3D Elevation Program [3DEP]. 3D Elevation Program, **2018**, Available online: <https://nationalmap.gov/3DEP/>, (Accessed 4/27/2018).
16. Jensen, J. *Remote sensing of the environment: an earth resource perspective*, Prentice Hall, Upper Saddle River, New Jersey, **2000**, pp 544.

17. Forest Inventory and Analysis Program [FIA]. Forest Inventory and Analysis National Core Field Guide: Field Data Collection Procedures for Phase 2 Plots. Version 6.0. Vol. 1; Internal Report, 2012; U.S. Department of Agriculture Forest: Washington, DC, USA, 2012, Available online: [http://www.fia.fs.fed.us/library/field-guides-methods-proc/docs/2013/Core%20FIA%20P2%20field%20guide\\_6-0\\_6\\_27\\_2013.pdf](http://www.fia.fs.fed.us/library/field-guides-methods-proc/docs/2013/Core%20FIA%20P2%20field%20guide_6-0_6_27_2013.pdf), (Accessed 6/5/2014).
18. Omernik, J.; Griffith, G. Ecoregions of the conterminous United States: Evolution of a hierarchical spatial framework. *Environ. Manag.* **2014**, *54*, 1249–1266.
19. Gesch, D.; Oimoen, M.; Greenlee, S.; Nelson, C.; Steuck, M.; Tyler, D. The National Elevation Dataset. *Photogr. Eng. Remote Sens.* **2002**, *68*, 5–11.
20. Homer, C.; Dewitz, J.; Yang, L.; Jin, S.; Danielson, P.; Xian, G.; Coulston, J.; Herold, N.; Wickham, J.; Megown, K. Completion of the 2011 National Land Cover Database for the conterminous United States—Representing a decade of land cover change information. *Photogr. Eng. Remote Sens.* **2015**, *81*, 345–354.
21. Gandhi, G.; Parthiban, S.; Thummalu, N.; Christy, A. 2015. Ndvi: Vegetation Change Detection Using Remote Sensing and Gis—A Case Study of Vellori District, *Procedia Comput. Sci.* **2015**, *57*, 1199–1210.
22. LANDFIRE. Existing Vegetation Type Layer, LANDFIRE 1.1.0, U.S. Department of the Interior, Geological Survey, **2008**, Available online: <http://landfire.cr.usgs.gov/viewer/> (Accessed on 4/28/2018).
23. Lowry, J.; Ramsey, R.; Boykin, K.; Bradford, D.; Comer, P.; Falzarano, S.; Kepner, W.; Kirby, J.; Langs, L.; Prior-Magee, J. Southwest Regional Gap Analysis Project: Final Report on Land Cover Mapping Methods; RS/GIS Laboratory, **2005**, Utah State University: Logan, UT, USA.
24. Escuin, S.; Navarro, R.; Fernandez, P. Fire severity assessment buy using NBR (Normalized Burn Ratio) and NDVI (Normalized Difference Vegetation Index) derived from LANDSAT TM/ETM images. *Int. J. Remote Sens.* **2008**, *29*, 1053–1073.
25. Davids, C.; Doulgeris, A. Unsupervised change detection of multitemporal Landsat imagery to identify changes in land cover following the Chernobyl accident. In Proceedings of the Geoscience and Remote Sensing Symposium, Barcelona, Spain; **2008**, pp. 3486–3489.
26. Weng, Q.; Fu, P.; Gao, F. Generating daily land surface temperature at Landsat resolution by fusing Landsat and MODIS data. *Remote Sens. Environ.* **2014**, *145*, 55–67.
27. Turner, M.; Gardner, G.; O'Neil, R. *Landscape Ecology in Theory and Practice: Pattern and Process*. Springer, New York. **2001**, pp 406.
28. Hogland, J.; Anderson, N.; Chung, W.; Wells, L. Estimating forest characteristics using NAIP imagery and ArcObjects. Proceedings of the 2014 ESRI Users Conference; July 14-18, 2014, San Diego, CA. Environmental Systems Research Institute: Redlands, CA, USA; **2014**, pp. 1-22.
29. Hogland, J.; Anderson, N.M. Improved analyses using function datasets and statistical modeling. Proceedings of the 2014 ESRI Users Conference; July 14-18, **2014**, San Diego, CA. Environmental Systems Research Institute: Redlands, CA, USA; **2014**, pp. 1-14.
30. Hogland, J.; Anderson, N.M. Estimating FIA plot characteristics using NAIP imagery, function modeling, and the RMRS raster utility coding library; U.S. Department of Agriculture, Forest Service, Pacific Northwest Research Station: Portland, OR, USA; **2015**, pp. 340-344.
31. Hogland, J.; Anderson, N. Function Modeling Improves the Efficiency of Spatial Modeling Using Big Data from Remote Sensing, *Big data and cognitive computing*, **2017**, 1:3.
32. Hogland, J.; Anderson, N.; St. Peters, J.; Drake, J.; Medley, P. Mapping Forest Characteristics at Fine Resolution across Large Landscapes of the Southeastern United States Using NAIP Imagery and FIA Field Plot Data. *ISPRS International Journal of Geo-Information*, **2018**, *7*(4), 140; doi:10.3390/ijgi7040140.
33. Wells, L.A.; Chung, W.; Anderson, N.M.; Hogland, J.S. 2016. Spatial and temporal quantification of forest residue volumes and delivered costs. *Canadian Journal of Forest Research*, **2016**, *46*: 832-843.
34. Shang, C.; Treitz, P.; Caspersen, J.; Jones, T. Estimation of forest structural and compositional variables using ALS data and multi-seasonal satellite imagery. *Int. J. Appl. Earth Obs. Geoinf.* **2019**, *78*, 360–371.
35. Ahl, R.; Hogland, J.; Brown, S. A comparison of standard modeling techniques using digital aerial imagery with National Elevation Datasets and airborne LiDAR to predict size and density forest metrics in the Sapphire Mountains MT, USA. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 24.
36. Noordermeer, L.; Martin, B.; Ørka, H.O.; Naesset, E.; Gobakken, T. Comparing the accuracies of forest attributes predicted from airborne laser scanning and digital aerial photogrammetry in operational forest inventories. *Remote Sens. Environ.* **2019**, *226*, 26–37.

37. Hogland, J.; Anderson, N. Chung, W. New Geospatial Approaches for Efficiently Mapping Forest Biomass Logistics at High Resolution over Large Areas. *ISPRS International Journal of Geo-Information*, **2018**, 7(4), 156. doi:10.3390/ijgi7040156.
38. Brunner, R.; Kim, E.J. Teaching Data Science, *Procedia Computer Science*, **2016**, 80, 1947-1956. doi: 10.1016/j.procs.2016.05.513
39. Hogland, J; Ahl, R. Image Processing and Classification, **2018**, Available online: <http://www.umt.edu/sell/cps/magip/workshopdescriptions/imageprocessing.php>, (Accessed 4/27/2018).
40. Hogland, J. Automating your workflow using Python: Focus on reusable code, **2018**, Available online: <http://www.umt.edu/sell/cps/magip/workshopdescriptions/python.php>, (Accessed 4/27/2018).
41. Microsoft©. ML.NET, **2018**. Available online: <https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet>, (Accessed 10/29/2019).
42. SciKit-Learn, SciKit-Learn Machine Learning in Python **2018**. Available online: <https://scikit-learn.org/stable/index.html>, (Accessed 10/29/2019).

## Chapter 2

# Mitigating the Impact of Field and Image Registration Errors through Spatial Aggregation

**Abstract:** Remotely sensed data are commonly used as predictor variables in spatially explicit models depicting landscape characteristics of interest (response) across broad extents, at relatively fine resolution. To create these models, variables are spatially registered to a known coordinate system and used to link responses with predictor variable values. Inherently, this linking process introduces measurement error into the response and predictors, which in the latter case causes attenuation bias. Through simulations, our findings indicate that the spatial correlation of response and predictor variables and their corresponding spatial registration (co-registration) errors can have a substantial impact on the bias and accuracy of linear models. Additionally, in this study we evaluate spatial aggregation as a mechanism to minimize the impact of co-registration errors, assess the impact of subsampling within the extent of sample units, and provide a technique that can be used to both determine the extent of an observational unit needed to minimize the impact of co-registration and quantify the amount of error potentially introduced into predictive models.

**Keywords:** attenuation; registration; aggregation; spatial correlation; co-registration

---

## 1. Introduction

Remotely sensed data play an ever-increasing role in characterizing and quantifying landscapes. These types of data have been used to study our surroundings [1], stratify the terrestrial environment [2, 3], and build a wide range of data products depicting terrestrial characteristics, such as topography [4], land use and cover [5], vegetative indices [6], vegetation communities [7, 8], fire severity [9], land cover change [10], and temperature [11]. Due to the success and relatively low cost of using remotely sensed data to depict landscape patterns and changes in those patterns, fields like landscape ecology [12] and concepts like spatial connectivity and the relationships between patterns and processes are now at the forefront of many land management and planning endeavors [13–16].

Ideas such as spatial contiguity, patch size, and patch juxtaposition, and their relationships to processes and concepts such as forest management, land use planning, and sustainable forestry have in part fueled the desire to precisely and accurately define existing patterns at fine spatial detail, across broad extents [17–19]. Coupled with the availability of fine-grained remotely sensed data ( $\leq 5$  m) and advancements in computer hardware and software [20], a fine-scaled depiction of the landscape can now be produced across broad extents relatively quickly, at a low cost [21–23]. At the same time, the fine-grain nature of these types of data provide unique opportunities to relate characteristics of the landscapes measured for small spatial extents (response variables) to remotely sensed data (predictor variables) collected across vast areas.

Many have capitalized on this point to develop mathematical, statistical, and spatial models that can be used to create surfaces depicting landscape variables of interest using geo-rectified field and remotely sensed data [18, 22–24]. Generally, this process can be described as: (1) registering both field and remotely sensed data to a known coordinate system, (2) using the spatial coordinates of the field and remotely sensed data to link measured values in the field to remotely sensed data, (3) building a model for the linked variable as a function of variables derived from the remotely sensed data, and (4) applying the model to remotely sensed surfaces to create a continuous surface of estimated characteristics. While straightforward, the linking process is subject to error (co-registration error) owing to the imperfectly identified spatial coordinates of the response and predictors, and this can have a negative impact on the accuracy of the model estimates (i.e., increased bias and imprecision). With regression models, predictor variables ( $X_i$ ) are assumed to be measured without error. Response variables ( $Y_i$ ) can be measured with error, and this is accounted for within the modeling process,



often by specifying an additive random discrepancy, typically denoted as  $\varepsilon_i$  [25]. Take for example a simple linear model equation:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i, \quad (1)$$

where  $\beta_0$  and  $\beta_1$  correspond to the intercept and slope, and  $\varepsilon_i$  corresponds to model error which includes any potential error associated with measuring the response variable. When co-registration errors occur, this amounts to the introduction of error into the ability to measure  $X_i$  (e.g., spectral values) coincident with  $Y_i$  (e.g., basal area per hectare). Measurement error in  $X_i$  is not typically accounted for in regression models and will cause attenuation bias [25, 26], which manifests in estimates trending towards the global mean of the response variable.

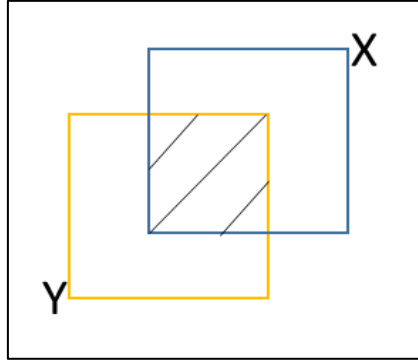
To circumvent the impacts of co-registration errors, analysts have employed a wide variety of solutions, ranging from rectifying images in a relative manner [27] to ignoring these errors and assuming them to be of little importance in predictions [28]. Regardless of the precision of measuring the true or relative surface location, spatial error will always be part of the rectification process and will have an impact on the underlying predictive model.

Within remote sensing literature, the impact of co-registration error has been recognized, especially for Light Detection and Ranging (lidar) data [29–32], but typically is not directly quantified. Often studies cite co-registration as an additional source of error that should be minimized, but fall short in describing the effects of those errors or providing suggestions to minimize the influence of those errors on predicted values. In this study, we address this knowledge gap by developing techniques to quantify this source of error and mitigate co-registration errors in applied work. Through simulation using Landsat 8 and National Agriculture Imaging Program (NAIP) imagery and images created with specific spatial correlation, based on Landsat 8 and NAIP images, we investigate co-registration errors and their impacts on the modeling process, and test the hypothesis that co-registration errors can be mitigated through spatial aggregation. Additionally, given estimates of global spatial continuity and co-registration errors, we provide recommendations on the size and layout of field observations with respect to the grain size of remotely-sensed data that will help to minimize the impact of co-registration errors.

## 2. Materials and Methods

### 2.1. Theoretical Background

The impact of co-registration errors on predictive models should be related to four primary factors: (1) the horizontal misalignment between response and predictor variables, (2) the spatial extent of the sample unit, (3) the spatial correlation of predictor and response variables, and (4) the strength and form of the relationship between response and predictor variables. Prior to performing a study, researchers typically do not know the spatial correlation of response variables, nor the strength or form of the relationship between response and predictor variables. To address this lack of information in our study, remove issues of measurement error, and focus our study solely on the impacts of co-registration error, we constrain our predictor surfaces to have a one-to-one relationship with our response variables. In this scenario, the  $Y$  and  $X$  surfaces, in the absence of co-registration errors, should exhibit a perfect linear relationship (i.e., an intercept of 0, a slope of 1, and a coefficient of determination of 1). Also, where the relationship between  $X$  and  $Y$  is linear, aggregated values (i.e., averages over multiple adjacent pixels) will also exhibit the same one-to-one relationship as non-aggregated values. Given this design, deviations from a one-to-one relationship can be solely attributed to co-registration errors.



**Figure 1.** Graphical depiction of co-registration error. The location of a sample unit determined by a global positioning system (GPS) ( $Y$ ) and its corresponding location found within an image ( $X$ ) represent the same extents located on surface of the earth, but due to co-registration errors,  $X$  and  $Y$  only share a portion of the same area in projected space (diagonal black lines).

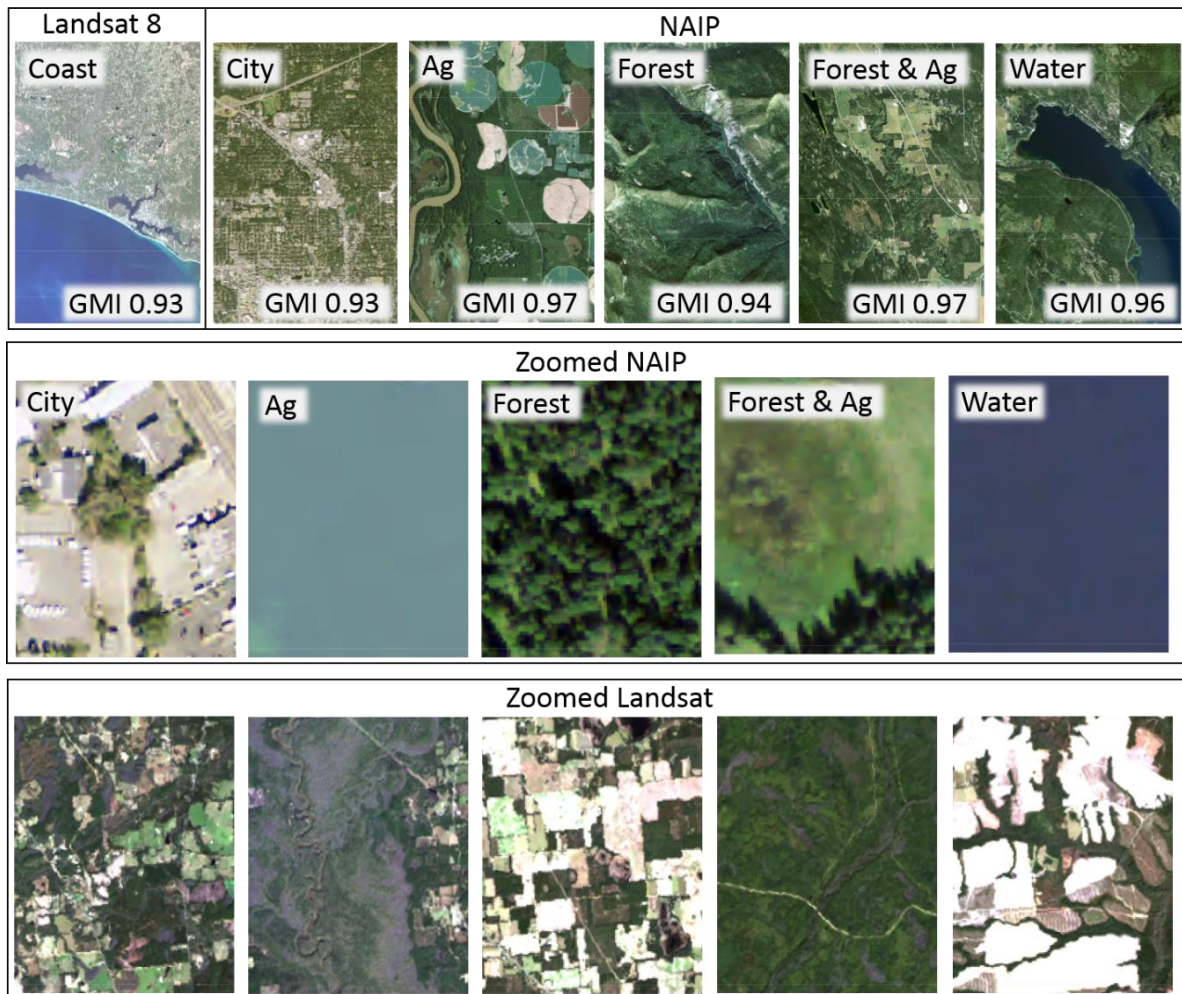
Additionally, assuming that co-registration errors manifest as random noise within the regression models, we anticipate that the proportion of variation in  $Y$  explained by  $X$  ( $R^2$ ) should follow the squared geometric relationship between the sample unit size ( $A_s$ ) and the area of overlap ( $A_o$ ) between the  $X$  and  $Y$  units, when the  $X$  values are distributed independently at random over space (Figure 1, Appendix A). This can be expressed as follows:

$$R^2 = \left(\frac{A_o}{A_s}\right)^2 \quad (2)$$

In concept, each sample unit's  $Y$  values are related to a combination of the corresponding  $X$  values now attached to an area only partially overlapping with the sample unit (cross-hatched area in Figure 1), as well as to  $X$  values attached to distinct spatial areas that have been falsely aligned with the sample unit. The latter occurs only because co-registration errors incorrectly identify a spatial match. If the  $X$  values are distributed independently at random over space, then on average the proportion of information on  $Y$  that can be explained by  $X$  should correspond to the average amount of area shared between response and predictor sample units, given the registration errors. Given this assumption, deviation from this condition in our simulations can be attributed to the spatial correlation within a landscape, and provide a rationale for using measures like global Moran's index (GMI) [33] as predictors, to estimate the proportion of modeling error contributed by co-registration errors.

## 2.2. Overview

All analyses within this study were performed using R [34]. Images created with specified amounts of spatial correlation (virtual images) were built using the raster [35] and gstat [36, 37] packages. Our simulations use one Landsat 8 [38] and five NAIP [39] images as baseline datasets taken from varying landscapes (Figure 2, Table 1), to produce nine virtual Landsat images and ten virtual NAIP images, respectively. To determine the amount of spatial correlation associated with the Landsat and NAIP baseline images, a uniform random selection of 20 locations was used to extract raster cell values within a 200 by 200 cell window, and to calculate empirical omnidirectional covariogram statistics [40].



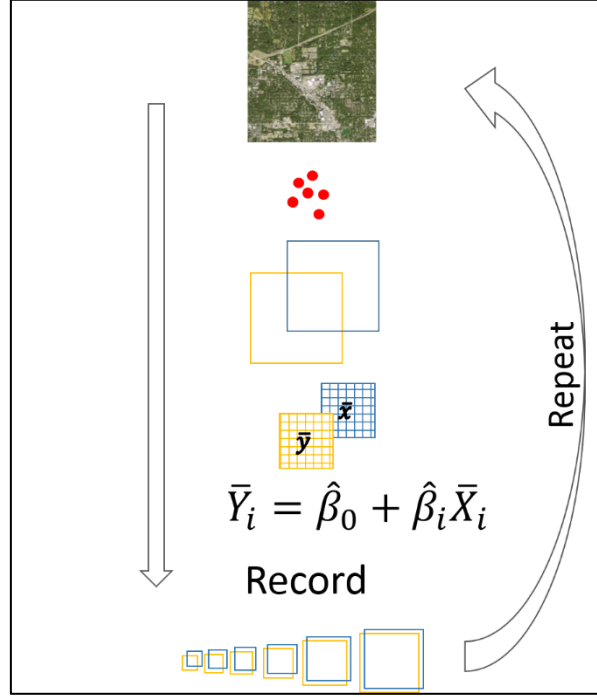
**Figure 2.** Base images used in simulations. Zoomed-in areas illustrate the extent for which images were subset and summarized to estimate mean digital number, sill, nugget, and range values.

**Table 1.** Average digital number (MDN), sill, nugget, and global Moran's index (GMI) and maximum range (in number of cells) values for Landsat and National Agriculture Imaging Program (NAIP) imagery. Averages and maximum values were based on all bands within an image.

Name	Label	MDN	Sill	Range	Nugget	GMI
Landsat 8 Coast	Coast	7,478.1	552,790.7	40.5	180,321.3	0.93
NAIP City	City	140.2	1,630.3	30.1	348.7	0.93
NAIP Agriculture	Ag	115.7	449.3	46.0	161.0	0.97
NAIP Forest	Forest	86.3	525.0	31.2	187.4	0.94
NAIP Forest & Agriculture	Forest & Ag	119.0	593.9	41.9	78.7	0.97
NAIP Forest & Water	Water	88.9	548.3	33.8	113.9	0.96

Cell values within each 200 by 200 cell window were summarized to estimate a mean digital number (DN) value, as well as sill, nugget, and range values for empirical omnidirectional covariograms. Mean DN, sill, and nugget statistics from each band were then averaged across image sources and used as inputs for creating Landsat- and NAIP-based virtual surfaces. To mimic different degrees of spatial correlation, range values were allowed to vary from 0.5 cells (completely random image) to the maximum range found among bands within each image source. Together, mean DN, sill, nugget, and ranges with a spherical spatial model were used to create virtual NAIP and Landsat surfaces (36, 37). A complete listing of the code used to estimate spectral and spatial statistics and create virtual Landsat and NAIP images can be found in Appendix B (general libraries).

After creating the single band virtual image, two simulated sampling experiments were conducted, using actual and virtual Landsat 8 and NAIP images to evaluate the impacts of co-registration errors, spatial aggregation, sampling intensity, and spatial correlation on model prediction. The first set of simulations (stage I) were used to quantify the impacts of spatial aggregation of individual cells into multi-cell sampling units with regards to model prediction given co-registration error and defined spatial correlations (Figure 3). To account for potential logistical constraints of sampling large areas in the field, a second simulation was performed (stage II) that explored the impacts of alternative subsampling configurations corresponding to varying levels of measurement intensity and sample unit extent.



**Figure 3.** Visualization of Stage I simulations. A total of 200 sample locations (red points) were used to extract and calculate mean values from an image for different spatial extents around a point before and after a spatial shift was introduced (yellow and blue squares). Values were then regressed against one another to determine the impact of co-registration errors. This process was performed for each image used in the study.

Due to computational limitations associated with calculating range values for the extent of Landsat and NAIP imagery, we explored using GMI as a surrogate for range. GMI, while different than range, quantifies spatial correlation as an index value bounded between  $-1$  (negative correlation) and  $1$  (positive correlation), with a value of zero corresponding to no spatial correlation (completely random image). For each band within each image of our simulations, GMI was calculated as follows:

$$\text{GMI} = \frac{N \sum_i \sum_j w_{ij} (x_i - \bar{x})(x_j - \bar{x})}{W \sum_i (x_i - \bar{x})^2}, \quad (3)$$

with  $x$  equal to values within a raster surface indexed by  $i$  and  $j$  rows and columns,  $w_{ij}$  representing a weighted spatial matrix (rook's case),  $N$  being the number of cells, and  $W$  being the sum of all weights. The remainder of this section describes in detail the design and implementation of each simulation stage within our study and model fitting used to estimate the impact of co-registration errors.



#### 2.4. Stage II Simulations

Preferably, when relating remotely sensed data to field samples, the entire area within a sample unit would be measured on the ground. However, due to practical limitations related to collecting field data for sample units with large spatial extents, this is often not economically feasible. This situation can lead to instances when the only practical way to estimate a mean for a spatial extent is to use subsampling. To quantify the impact of six common subsampling (subplots) layouts and various subsampling intensities (area measured) within a given sample unit size (plot), we investigated multiple plot/subplot layouts. A depiction of plot extents, subplot layouts, and subsampling intensities are illustrated in Figure 4.

Identical to Stage I simulations, Stage II simulations mimic registration errors for 200 observations and extract cell values surrounding each plot location for each sample unit size. For the response variable, the mean values for each sample unit size are calculated based on the spatial extent of one of six subplot layouts, and subsampling intensities ranging from 0.05 to 0.95 of the plot extent, by increments of 0.05. Subplot layouts include one subplot located in the center of the plot (One), four subplots located systematically in the corners of the plot (Sys 4), four randomly located subplots within the plot (Rnd4), four subplots oriented in a similar fashion as the U.S. Forest Service Forest Inventory Analysis (FIA) program plot protocol (FIA 4)[43], five subplots systematically placed within the plot extent (Sys 5), and nine plots systematically placed within the subplot (Sys 9). For predictor variables, mean values were calculated using all cell values within the extent of the plot ( $P_{all}$ ), and for only the areas within the subplots ( $P_{sub}$ ). Regression fit statistics and coefficients were then compared with results from 100% of the sample unit size measured in stage I.

#### 2.5. Modelling the Impacts of Co-Registration Error

After performing each simulation and recording error and fit statistics for each image, we developed a suite of models to relate those statistics to predictors measuring spatial correlation in the images (GMI) and the magnitude of spatial co-registration errors (expected proportion of area overlapped between field plots and corresponding image locations). While the overlap between two rectangles can be calculated if both the distance and direction of co-registration errors are known, the direction of co-registration errors is seldom calculated or reported. Therefore, within our iterations we estimated the expected proportion of overlap (PO) for each sample unit size, given the offsets used to simulate co-registration errors.

For virtual images with no spatial correlation, we hypothesized a one-to-one relationship between  $PO^2$  and the proportion of variation explained ( $R^2$ ). However, as spatial correlation increases within images, we anticipate that the ratio of  $R^2$  to  $PO^2$  will be greater than one and will interact with spatial correlation metrics. Additionally, we recognize that  $PO^2$  would be difficult to calculate in practice given commonly reported horizontal rectifications. Therefore, when modeling the impact of co-registration errors on  $R^2$  in the presence of spatial correlation, we used only sample unit size and GMI as predictors and beta regression with a logit link [44]. Similar in concept to logistic regression, beta regression was developed to work with observations between zero and one, and is typically used to characterize natural rates or proportions on a continuous scale. Using a logit link, our proposed model takes the following form:

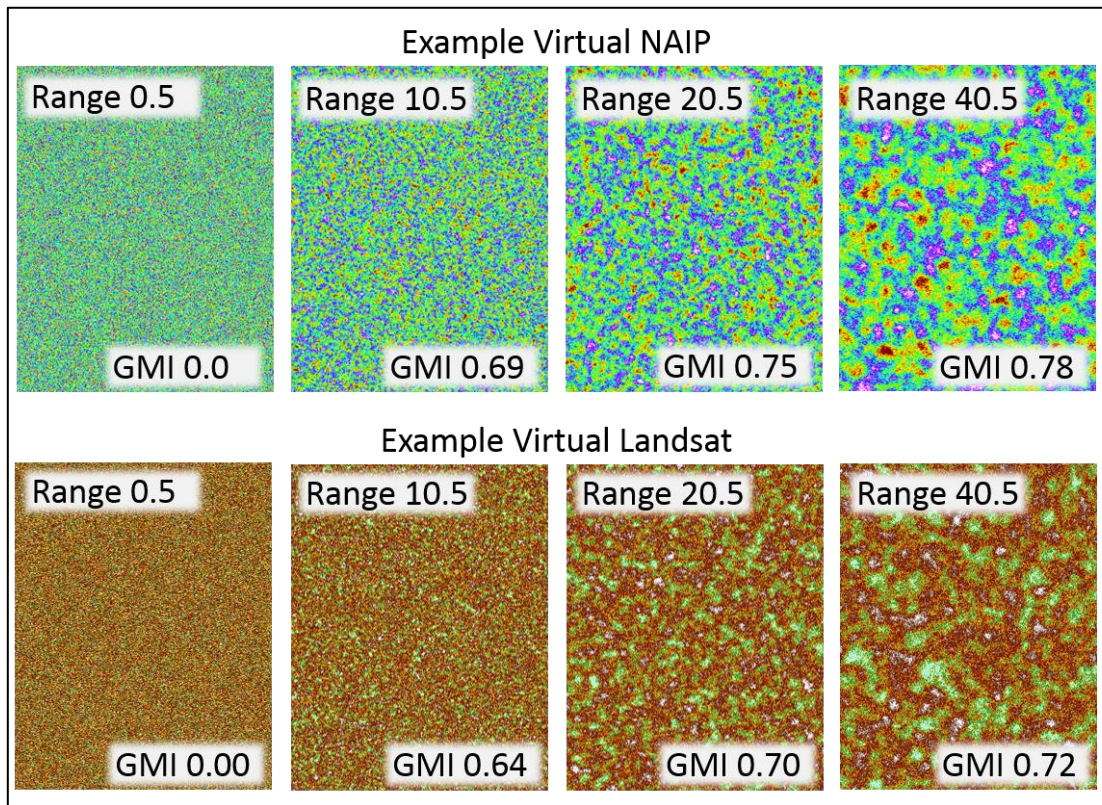
$$\ln\left(\frac{R^2}{1-R^2}\right) = \beta_0 + \beta_1 f(|samlpe\ unit|) + B_2 g(GMI) + \beta_3 f(|sampleunit|) * g(GMI) \quad (4)$$

where  $f()$  and  $g()$  are known transformations of the sample unit size and image GMI, respectively, and the  $\beta_k$  are parameters estimated from the data. Transformations of predictor variables were determined based on graphical analyses. While we anticipated needing sample unit size, GMI, and their interaction to estimate  $R^2$ , we also evaluated nested models using only sample unit size and GMI. All beta regression models were compared using Akaike's information criterion (AIC)[45, 46].

### 3. Results

#### 3.1. Datasets

Estimated mean DN, sill, and nugget and maximum range values varied by image (Table 1). While most of these characteristics varied substantially by data source due to pixel depth (Landsat 16-bit pixel depth versus NAIP 8-bit pixel depth), range values, measured in cells, were quite similar. Using the average DN, sill, and nugget and maximum range values of each data source, we created nine virtual Landsat images and ten virtual NAIP images of varying spatial correlation (Figure 5). It should be noted that virtual image GMI values were less than actual image GMI values, suggesting that there was less positive spatial correlation in the virtual images than in the actual images. However, the range of simulated autocorrelations produced virtual images with a variety of spatial structures and aggregated patterns that closely resembled patterns found within homogenous patches of actual images (Figure 2 zoomed-in examples and Figure 5). Generally, the boundaries between patches representing different DN values within the virtual images were not as sharp when compared to the base images. However, the patterns created in the virtual images provide an objective way to evaluate varying levels of spatial correlation.

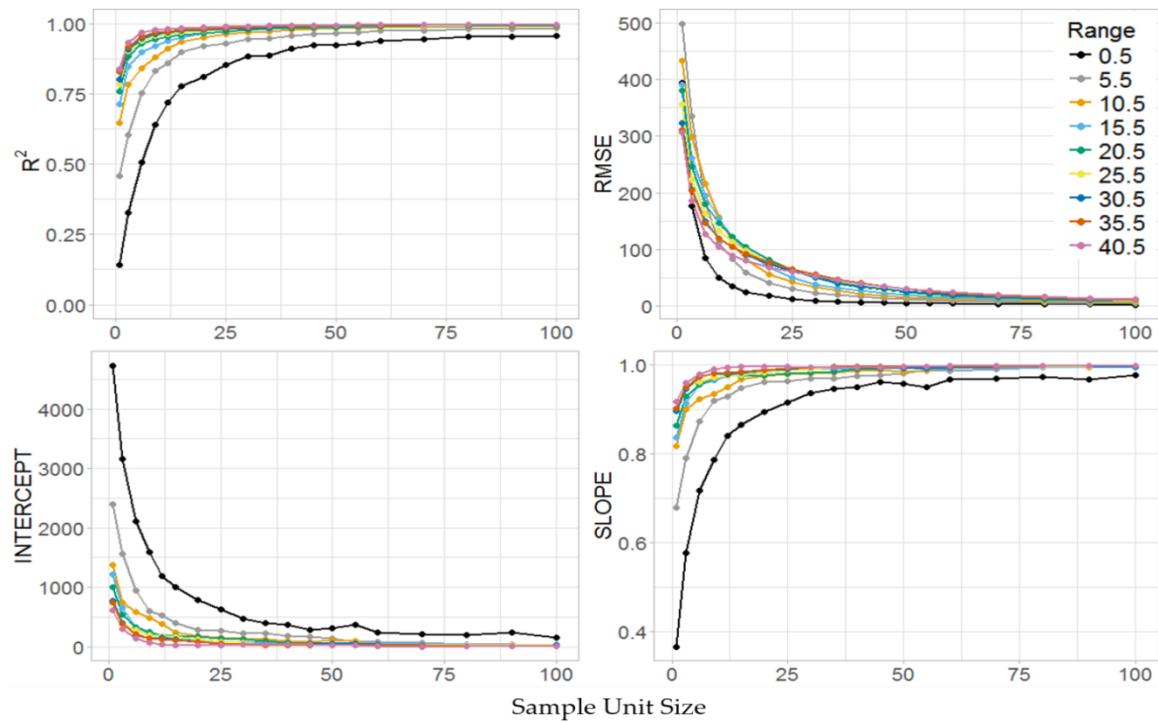


**Figure 5.** Subset of images created from average digital number (DN), sill, and nugget and maximum range values derived from NAIP and Landsat 8 imagery.

#### 3.2. Stage I

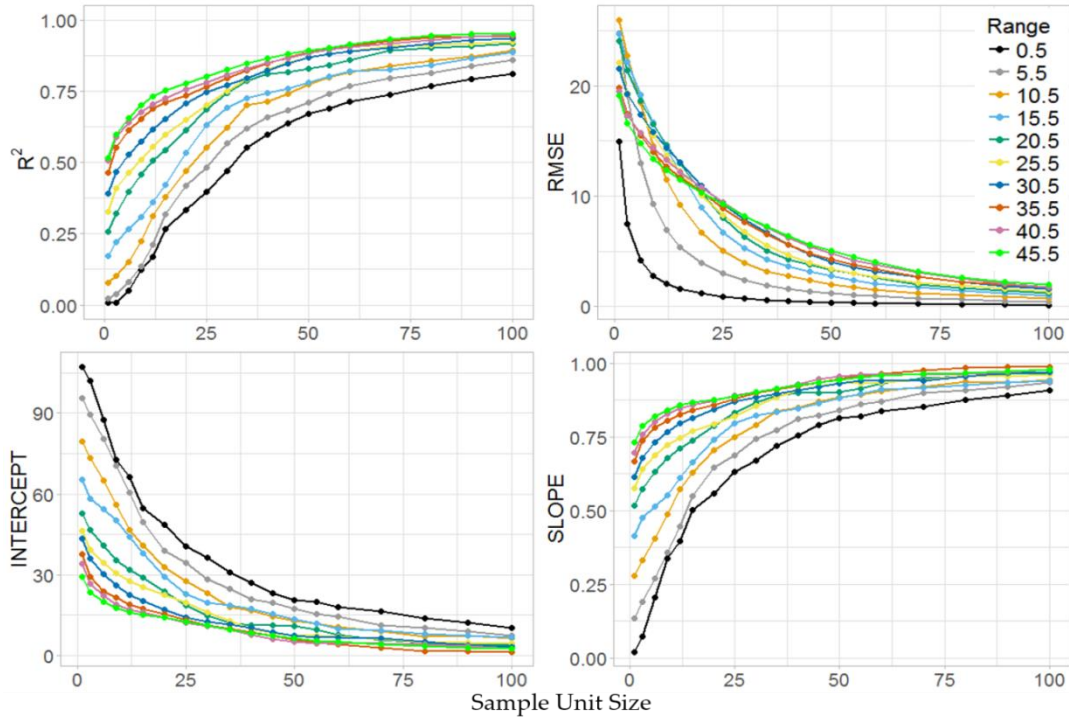
Comparisons in stage I indicate that increasing the spatial footprint of a sample unit can mitigate the effects of co-registration errors on predictive models. On average, horizontal shifts between  $L_1$  and  $L_2$  locations were 1.6 and 7.8 cells for Landsat 8- and NAIP-based images, respectively. For all images and bands analyzed, the extent of the sample unit was strongly related to the magnitude of deviation from the anticipated one-to-one regression relationship (intercept, slope, and  $R^2$  equal to 0, 1, and 1, respectively). Linear models derived from raster datasets with large spatial correlation, in terms of range or GMI, produced slope and intercept estimates closer to 1 and 0, respectively (less

attenuation), than raster datasets, with less spatial correlation for both Landsat- and NAIP-based datasets (Figure 6 and 7). While larger sample unit sizes reduced attenuation bias, for spatial correlation ranges above 30 cells, sample unit sizes greater than approximately 9 and 40 cells for Landsat- and NAIP-based images, respectively, appear to produce only marginal reductions in parameter bias or improvements in  $R^2$ . For NAIP-based imagery, this suggests that a field plot with an extent as large as 40 m by 40 m might be required to mitigate the effects of co-registration errors between NAIP imagery and GPS locations. Similarly, for Landsat-based images a field plot with an extent as large as 270 m by 270 m may be required to mitigate model error introduced by co-registration error.



**Figure 6.** Stage I Virtual Landsat image regression statistics for varying sample unit sizes and spatial correlation (Range), given an average image registration error of 1.6 cells and an average GPS navigational unit error of 0.23 cells. Actual Landsat image regression statistics are shown in Appendix C (Figure A1).

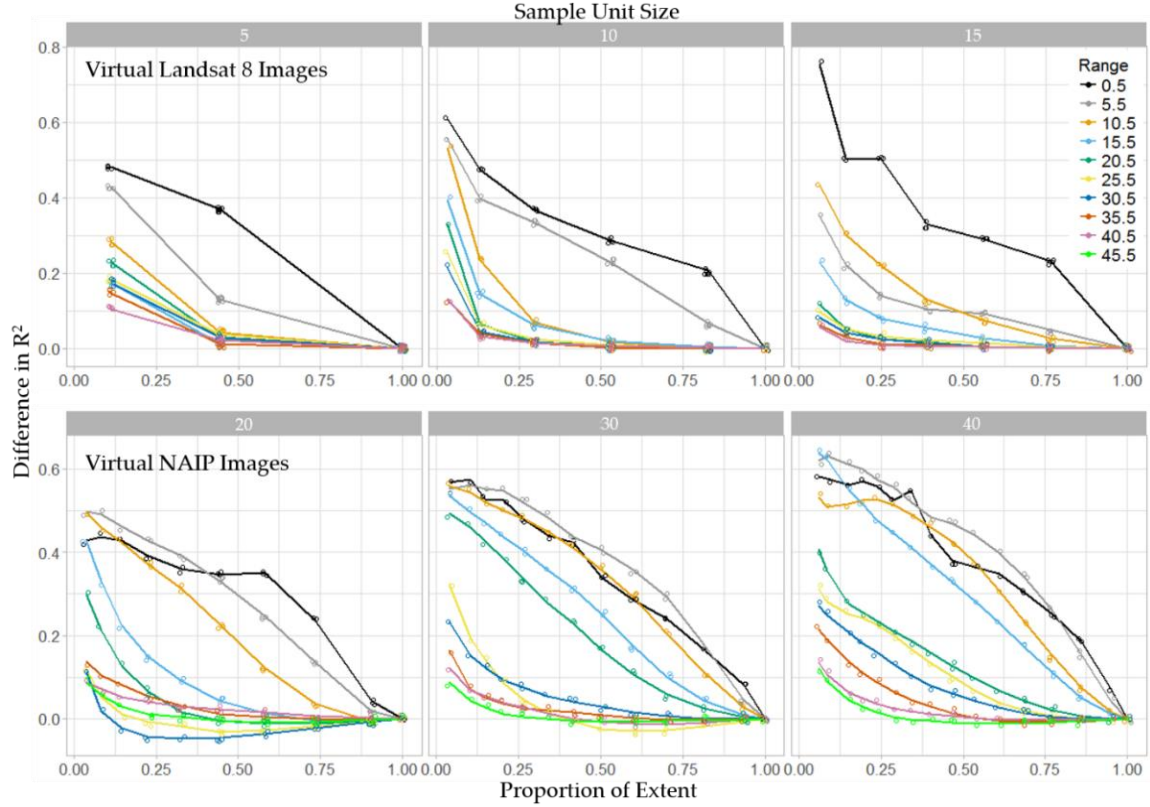




**Figure 7.** Stage I virtual NAIP image regression statistics for varying sample unit sizes and spatial correlation (Range), given average raster and GPS registration errors of 6 and 7 cells respectively. Actual NAIP image regression statistics are shown in Appendix C (Figure A2).

### 3.3. Stage II

Comparisons in stage II had similar trends as found in stage I, and verify that subsampling intensity and layout also impacted the amount of variation explained by models in the  $P_{all}$  subsampling scenario. For sample unit sizes between 20 and 50 cells and 5 and 20 cells for NAIP- and Landsat-based imagery, respectively, larger proportions of the area subsampled within a sample unit consistently explained more variation within the data, and produced smaller RMSE across all levels of spatial correlation and data sources. After the proportion of area subsampled reached approximately 80% of the plot extent ( $P_{sub}$ ),  $R^2$  appeared to differ only marginally relative to the  $R^2$  associated with  $P_{all}$  (Figure 8). This was also the case for RMSE. Across all subsampling intensities and sample unit sizes, the worst-performing subplot layouts were Rnd 4, FIA 4, and Sys 5. Subplot layouts One, Sys 4, and Sys 9 produced similar results, especially when the proportion of area measured within the plot extent was greater than 75%. As expected,  $P_{sub}$  generally produced better results than  $P_{all}$ , given that the response and predictor variables shared the same spatial configurations. However, there was little difference between  $P_{sub}$  and  $P_{all}$  subsampling techniques when greater than 80% of the plot extent was measured. As one might expect, smaller subsampling intensities (< 20% of the sample unit extent) substantially reduce  $R^2$  in our linear models. In some cases, when subsampling intensity and spatial correlation was small, the reduction in  $R^2$ , compared to measuring all the area within a plot extent, was greater than 60%. However, for actual Landsat 8 and NAIP images, which have relatively high levels of spatial correlation, the reduction in variation ranged from approximately 0.4% to 30%, depending on the data source, subsampling intensity, spatial correlation, and co-registration errors (Appendix C, Figures A3, A4). Similar to stage I simulations, increased amounts of spatial correlation generally dampened the negative effects of co-registration errors in stage II simulations. Additionally, this same dampening effect carried over to subsampling intensities when estimating means for all cells within a sample unit extent of a predictor variable.



**Figure 8.** Reduction in the proportion of variation explained ( $R^2$ ) for Landsat 8 and NAIP virtual images by subsample intensity (Proportion of Extent), sample unit size (in cells), and spatial correlation (Range) using SYS 4 and  $P_{all}$  subsampling layout. Figures A3 and A4 in Appendix C show actual Landsat 8 and NAIP image reductions.

**Table 2.** Linear regression statistics for 19 independently random images, given the average proportion of overlap (PO) determined by sample unit size and simulated co-registration errors.

Model	Equation	Slope	$R^2$	RSE	F-stat	P-value
Landsat 8	$R^2 = 0 + PO^2$	1.008	0.9984	0.03456	11440	< 0.001
NAIP	$R^2 = 0 + PO^2$	1.035*	0.9983	0.02364	10330	< 0.001

\* Statistically different than one at  $\alpha = 0.01$ .

### 3.4. Model Fitting

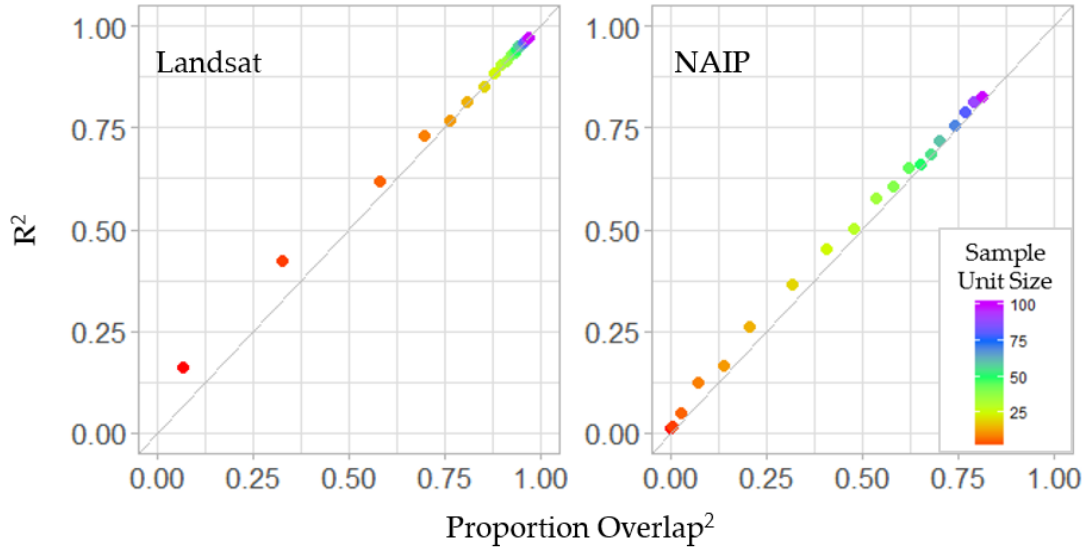
Regressed mean DN values for  $L_1$  and  $L_2$  locations in both simulations indicate that co-registration errors can have substantial impacts on model fit, and can bias DN estimates. Globally, across the extent of each image, estimates of mean DN were necessarily unbiased. However, local estimates tended to over- or under-estimate DN values that were respectively smaller or larger than the mean (attenuation). The degree of attenuation in our models, identified by deviations from theoretical intercept and slope, was strongly related to both the spatial extent of an observation (sample unit size) and the spatial correlation of predictor variables (Figure 6 and 7).

For completely independent virtual images, the amounts of variation explained in our linear models were closely related to  $PO^2$  (Table 2, Figure 9). For both Landsat 8- and NAIP-based imagery with average co-registration errors of 1.6 and 7.8 cells, respectively,  $R^2$  and  $PO^2$  closely followed a one-to-one ratio. For images with spatial correlation, exploratory analysis revealed that sample unit size and GMI did not appear to be linearly related to the logit of  $R^2$ . However, the natural log of sample unit size (LSS) and the exponentiation of GMI (EGMI) did appear to be linearly related to  $R^2$ . Therefore, we included LSS and EGMI in our suite of models for comparison (Table 3). Our top fitting models were statistically significant ( $p$ -value < 0.001), and included LSS, GMI, and the interaction

between LSS and GMI for both Landsat 8- and NAIP-based images (Table 4). RMSE values for top-fitting Landsat 8 and NAIP models were 0.019 and 0.089, respectively (expressed on the scale of  $R^2$ ). Regression diagnostics of our top-fitting models are shown in Appendix A, Figure A5. Untransformed, observed versus predicted  $R^2$  followed a one-to-one relationship for both Landsat and NAIP-based imagery (Figure 10), and the latter was constrained to fall between 0 and 1, with more variation occurring within the middle portion of the observed domain, as expected.

**Table 3.** Suite of potential models and their associated AIC and  $\Delta$ AIC values. Interaction term denoted by \* specifies a full interaction model.

Model	Rank	Source	Predictors	AIC	$\Delta$ AIC
1	6	Landsat 8	<i>sample unit size</i>	-1723.823	-651.018
2	4	Landsat 8	<i>sample unit size + GMI</i>	-1920.781	-454.06
3	3	Landsat 8	<i>sample unit size * GMI</i>	-1938.580	-436.261
4	5	Landsat 8	$\ln(\text{sample unit size})$	-1870.267	-504.574
5	2	Landsat 8	$\ln(\text{sample unit size}) + e^{GMI}$	-2368.824	-6.017
6	1	Landsat 8	$\ln(\text{sample unit size}) * e^{GMI}$	-2374.841	0
1	6	NAIP	<i>sample unit size</i>	-1086.606	-681.595
2	5	NAIP	<i>sample unit size + GMI</i>	-1469.469	-298.732
3	3	NAIP	<i>sample unit size * GMI</i>	-1494.508	-273.693
4	4	NAIP	$\ln(\text{sample unit size})$	-1193.989	-574.212
5	2	NAIP	$\ln(\text{sample unit size}) + e^{GMI}$	-1728.674	-39.527
6	1	NAIP	$\ln(\text{sample unit size}) * e^{GMI}$	-1768.201	0

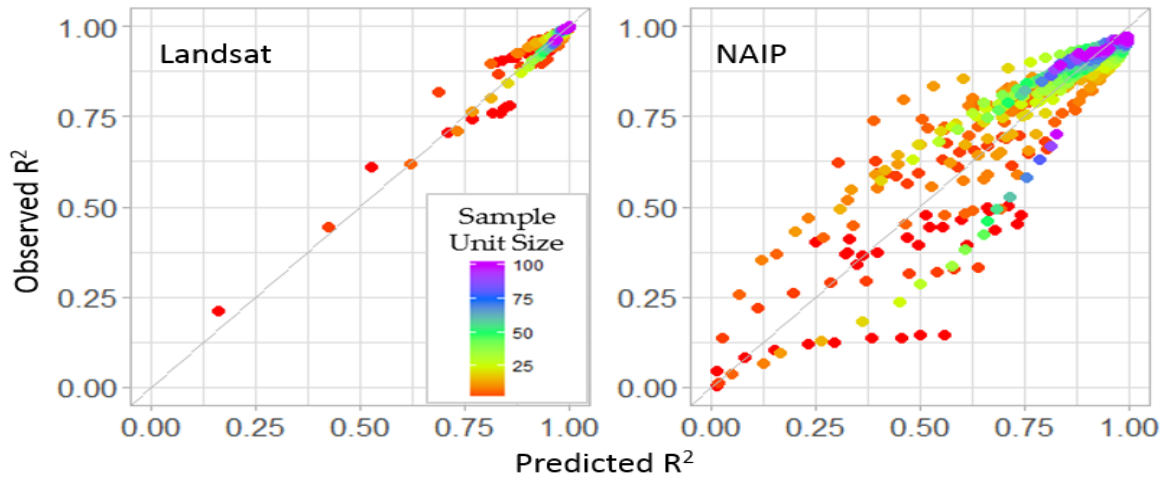


**Figure 9.** Scatter plot of proportion of variation explained ( $R^2$ ) versus the squared proportion of overlap between  $L_1$  and  $L_2$  locations, given various sample unit sizes, independent random surfaces, and Landsat 8 and NAIP horizontal registration errors. The gray diagonal line is a one-to-one line, for the purpose of comparison.

**Table 4.** Beta regression coefficients and statistics for top fitting Landsat 8 and NAIP based images given natural log sample unit size (LSS), exponent of global Moran’s index (EGMI), interaction between LSS and EGMI, and simulated average co-registration errors.

Model	N	Intercept <sup>+</sup>	LSS <sup>+</sup>	EGIM <sup>+</sup>	EGMI * LSS <sup>+</sup>	Pseudo DR <sup>2</sup> [45]	P-value
Landsat 8	304	-3.743	1.089	2.423	-0.085	0.918	< 0.001
NAIP	570	-9.364	1.883	3.481	-0.419	0.8255	< 0.001

<sup>+</sup> Statistically different from zero at  $\alpha = 0.01$ .



**Figure 10.** Observed versus predicted proportion of variance explained ( $R^2$ ) for co-registration errors associated with Landsat 8 and NAIP imagery and virtual imagery, given various sample unit sizes, measures of spatial correlation, and top-fitting beta regression models. The gray diagonal line is a one-to-one reference line.

#### 4. Discussion

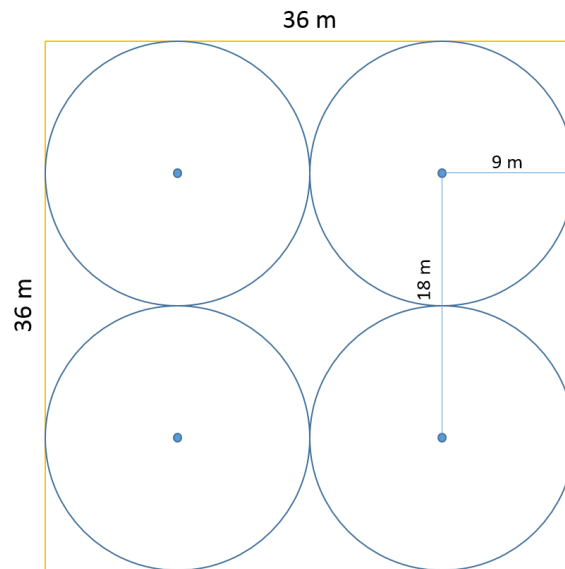
Through our simulations, we have documented that spatial co-registration errors produce attenuation bias in linear models (Figures 6 and 7). For studies that relate field data located using GPS to geo- or ortho-rectified remotely sensed data, this bias will manifest in regression coefficients biased toward 0 and regression estimates trending towards the sampled mean value of the variable of interest (Appendix C, Table A1). While every attempt to minimize the amount of co-registration error should be taken, technical and financial limitations often make it impractical to completely remove this source of error. Due to these limitations, we explored the impacts of spatial aggregation of observational units on model performance when predictor variables have spatial co-registration errors.

Our findings demonstrate that increasing the spatial extent of sample units can help to reduce the impacts of imperfect co-registration. This result further verifies that larger field plots can mitigate the effects of co-registration error found by others [29, 30, 47, 48]. However, when choosing the extent of a field sample unit, one must take into consideration practical issues associated with the costs of implementation and measurement (i.e., large plots cost more to measure), as well as the fact that large field sampling units can have a smoothing effect on spatial variability [29]. Moreover, subsampling within the extent of a field plot, regardless of the subplot layout, introduces additional variability into the predictive models, and should be used sparingly when spatially relating field measurements to remotely sensed information.

Effectively linking remotely sensed data to field plot data is a powerful tool for landscape modeling and requires thoughtful design to minimize the negative impacts of co-registration errors. Given the sample unit sizes, co-registration errors, and spatial correlation we investigated, we recommend selecting a field plot extent large enough to substantially reduce bias in linear regression, while also keeping the extent of the field plot as small as possible to retain spatial detail. In the case of NAIP imagery, this recommendation would correspond to a field plot with an area between 400 m<sup>2</sup> and 1,600 m<sup>2</sup>. For Landsat 8 imagery, this recommendation corresponds to a field plot with an area between 8,100 m<sup>2</sup> and 72,900 m<sup>2</sup>. Fortunately, most NAIP and Landsat 8 images have a large degree of spatial correlation, suggesting that the lower end of these recommendations may suffice in mitigating the impacts of co-registration errors. For other sources of remotely sensed information that have different co-registration errors, simulations similar to those presented in this study should be completed to help determine suitable field plot extents and sampling intensities.

If subplots are used to estimate mean values within the extent of a sample unit, it is important that the subplot layout covers as much of the area within the extent of the sample unit as possible. For NAIP imagery, we recommend measuring 75% or more of the sample unit area to minimize the negative effects of subsampling. When it is too costly to measure 75% of the area within a sample unit, a tradeoff between cost and precision must be made. In this situation, collecting more sample units with less than 75% of the subsample area measured can help to offset the losses in precision associated with subsampling (Figure 9). Additionally, when subsampling is used, layouts should be chosen such that there is no overlap among subplots, such as layout Sys 4 from this study (e.g., Figure 11).

The actual extent of a sampling unit should depend on the amount of co-registration error, the spatial correlation within the imagery, and the amount of model error one is willing to accept, and the resources available for measuring plots in the field. For readily available Landsat and NAIP imagery, their reported horizontal accuracies, and their estimated spatial correlations, we can estimate the co-registration error-induced reduction in variation explained by linear regression for various sample unit sizes (Equation 4 and Table 4). From these estimates, one can select a sample unit extent that both reduces estimation bias and quantifies error in predictor variables due to co-registration. For example, if a project was to use NAIP imagery with a sample unit size of 20 cells (field plot extent of 400 m<sup>2</sup>) and an estimated GMI of 0.92, then one would expect the logit of  $R^2$  to be approximately 1.744, and the loss in predictive ability associated with co-registration errors to be  $1-R^2 = 0.149$ .



**Figure 11.** Example of a recommended field plot size and layout for NAIP imagery.

While Equation 4 and the coefficients from Table 4 can be used to help guide the size of a field plot needed to mitigate the negative impacts of co-registration (Appendix C, Table A2), they should be interpreted as a best-case scenario. Specifically, our simulations were developed under the premise that there was a perfect one-to-one relationship between response and predictor variables. In many applications this will not be the case, and co-registration errors will be coupled with model error. To decouple co-registration errors from model errors, model coefficients can be dis-attenuated [26, 49]. Within that context, simulations similar to the ones performed in our study, which use a random sample of the predictor variables and regress those values against shifted locations, can be used to estimate a ratio adjustment factor for model coefficients, as described by Forest and Thompson [49]. Appendix B (general libraries) provides examples of  $R$  coding that can be used to simulate co-registration errors and determine ratio adjustment factors.

Fortunately, most remotely-sensed images have relatively high levels of spatial correlation, which in turn dampens the impacts of co-registration errors. In our study, we evaluated the effects of co-registration on model error for levels of spatial correlation that spanned independent random landscapes, to those commonly found in terrestrial environments. For all actual landscapes used in our study, the minimum spatial correlation found had a GMI value of 0.93. Interestingly, virtual images with ranges comparable to actual image ranges had corresponding GMI values that were substantially less than those found in the actual images. This is likely due to the dramatic transitions found between land use and cover types that can occur within actual landscapes (e.g., a forest adjacent to agricultural lands). This further suggests that natural landscapes have more localized spatial correlation than our virtual landscapes, and that edges between land use and cover types constitute a substantial amount of the overall area within an image. Because these edge areas can make up a substantial component of the landscape, it is important that they are included in future investigations that use simulated landscapes, and more importantly, model training. Mapping endeavors that omit these transition areas from training sets do so at the cost of extrapolating model results to potentially large portions of an image.

## 5. Conclusions

In this study, we looked at the impacts of co-registration errors on model prediction. We found that increasing field plot size helps to mitigate the negative impacts of co-registration errors by reducing attenuation bias. Additionally, we identified that increased positive spatial correlation within imagery reduces the negative impacts of co-registration for a given sample unit size. Finally, we presented a simulation methodology that can be easily applied to remotely sensed data that both quantifies the impact of co-registration on model prediction and can be used to estimate measurement error in predictor variables. Using our plot size recommendation and components of the simulation techniques described, estimation bias can be mitigated, which in turn should help analysts and managers to precisely define the complex spatial relationships needed to promote more effective, spatially informed decision making.

### Appendix A

*The relationship between the proportion of overlapping area between two square sampling units offset by a specified direction and distance (PO) and Pearson's correlation.*

Let  $X(p)$  be the DN value of pixel  $p$  in a random raster and  $X(p) = \mu + e(p)$  where  $e(p)$  are offsets from the mean DN  $\mu$  with expected value 0 and variance  $\sigma_e^2$ . Then let  $X(b)$  be the mean DN value of a block  $b$  of pixels. Denote the size (in pixels) of  $b$  by  $|b|$ . Then

$$X(b) = |b|^{-1} \sum_{p \in b} X(p) = \mu + |b|^{-1} \sum_{p \in b} e(p) \quad (5)$$

For blocks selected uniformly at random, the expected value of  $X(b)$  is  $\mu$  and its variance is  $|b|^{-1} \sigma_e^2$  provided that  $e(p)$  are uncorrelated (but  $\text{var}[X(b)] > |b|^{-1} \sigma_e^2$  if there is positive spatial autocorrelation among the  $e(p)$ ). The covariance between any  $X(b)$  and  $X(b')$  is

$$\text{cov}[X(b), X(b')] = E[e(b)e(b')] \quad (6)$$

$$= \frac{1}{|b|^2} E \left[ \left( \sum_{p \in b} e(p) \right) \left( \sum_{p \in b'} e(p) \right) \right] \quad (7)$$

$$\geq \frac{1}{|b|^2} E \left[ \left( \sum_{p \in (b \cap b')} e(p)^2 \right) \right] = \frac{|b \cap b'|}{|b|^2} \sigma_e^2 \quad (8)$$

with equality holding only if the  $e(p)$  are spatially uncorrelated.

Co-registration error can be simulated by shifting the original raster  $X$  by some random amount, resulting in a shifted raster  $Y$  where  $Y(p) = X(p')$  and thus  $Y(b) = X(b')$ . If the  $e(p)$  are spatially uncorrelated, then

$$\text{cov}[X(b), X(b')] = \text{cov}[X(b'), X(b)] = \frac{|b \cap b'|}{|b|^2} \sigma_e^2. \quad (9)$$

Furthermore,

$$\text{corr}[Y(b), X(b)] = \frac{\text{cov}[X(b'), X(b)]}{\sqrt{\text{var}[X(b')] \text{var}[X(b)]}} = \frac{|b \cap b'| |b|^{-2} \sigma_e^2}{|b|^{-1} \sigma_e^2} = \frac{|b \cap b'|}{|b|} \quad (10)$$

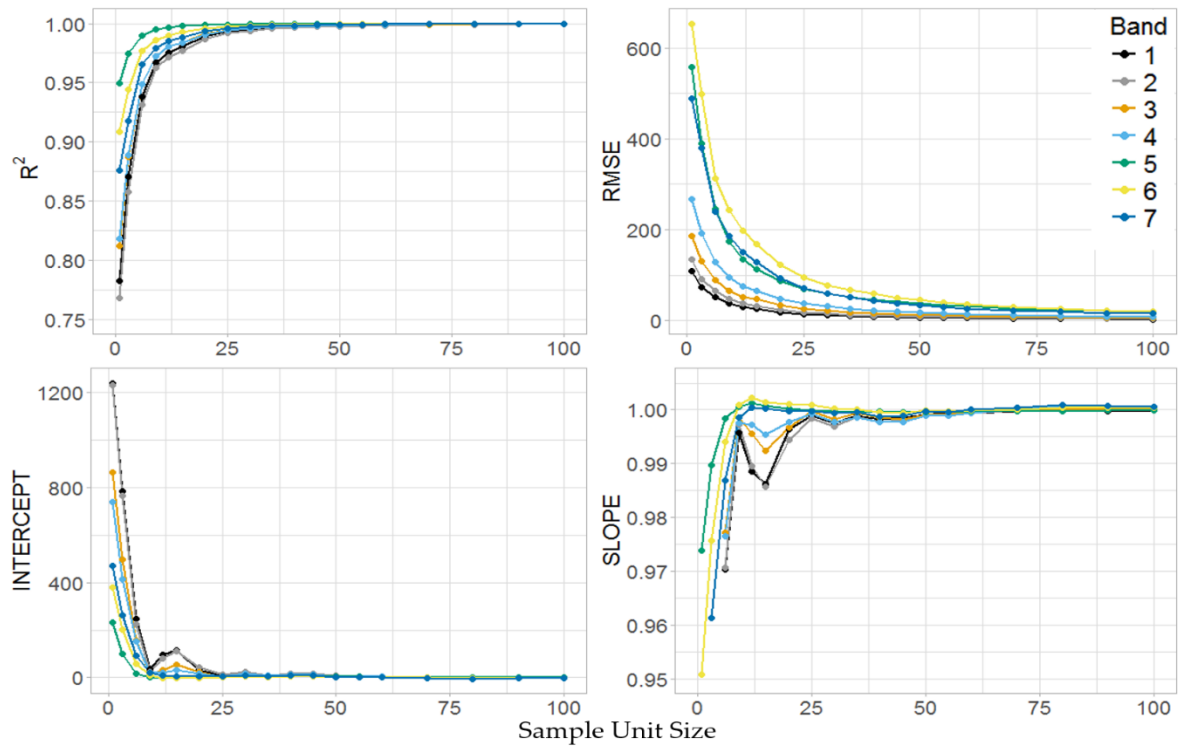
Where the last quantity is the proportion of the original block  $b$  that is overlapped by the shifted block  $b'$ . As a result, the coefficient of determination ( $R^2$ ) obtained by regressing  $Y(b)$  on  $X(b)$  will be directly related to the proportion of block overlap:

$$R^2 = (\text{corr}[Y(b), X(b)])^2 = \left( \frac{|b \cap b'|}{|b|} \right)^2 \quad (11)$$

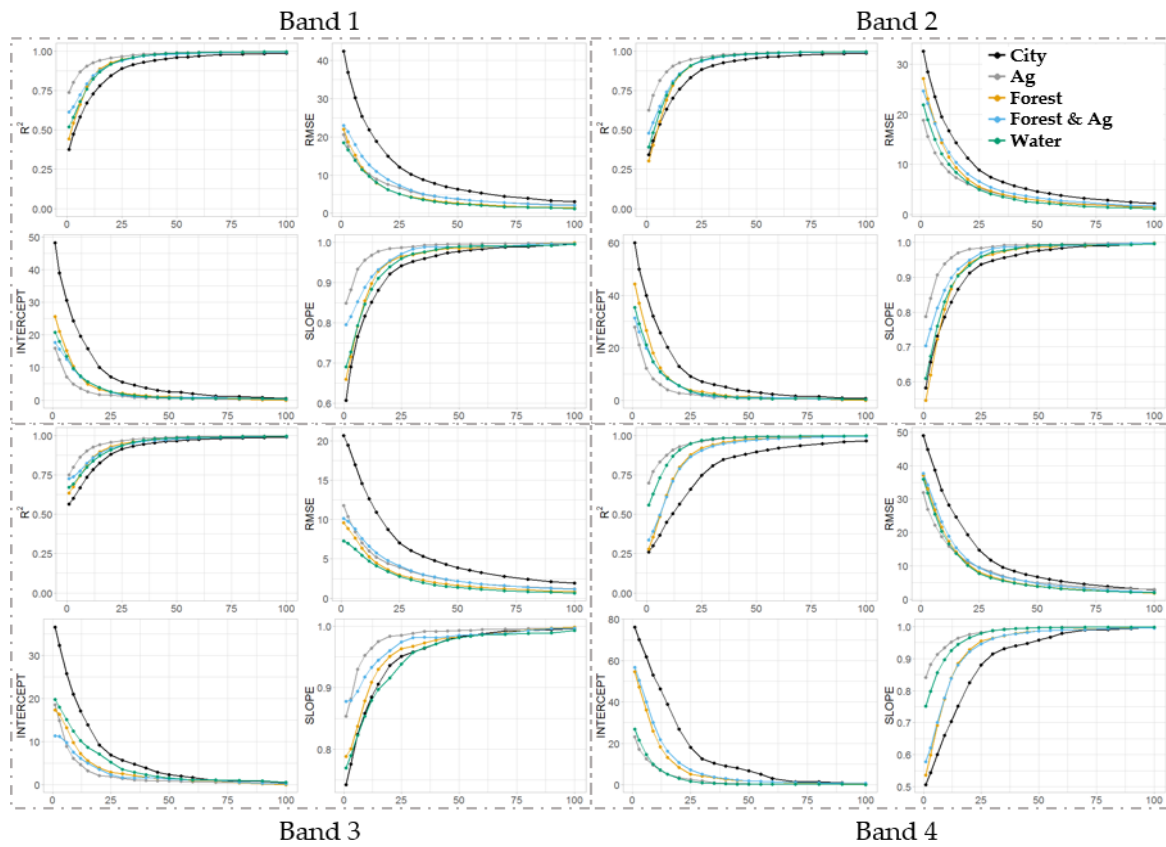
### Appendix B

See R Libraries coding section (pp. 76-144).

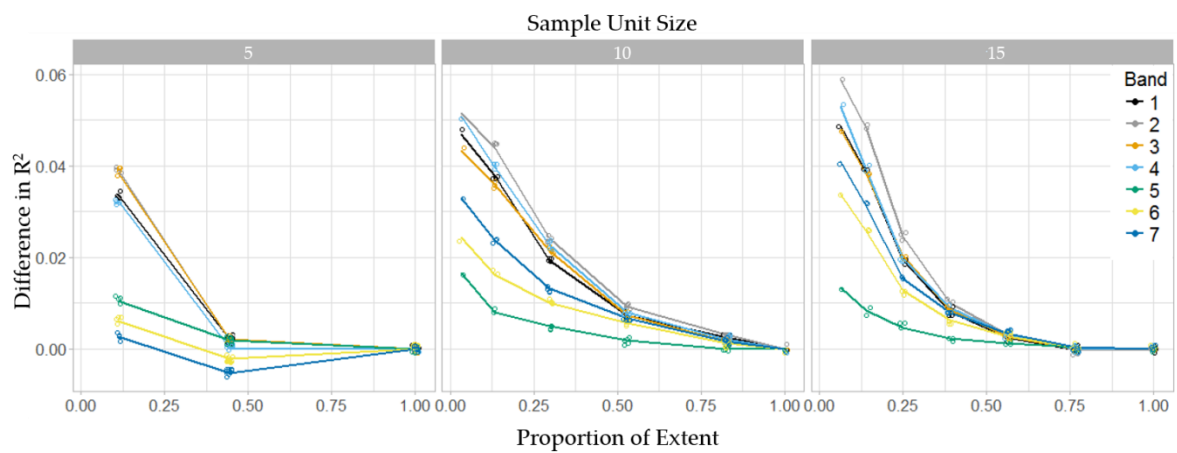
### Appendix C



**Figure A1.** Stage I Landsat image regression statistics for varying bands and sample unit size lengths given an average image registration error of 1.6 cells and an average GPS navigational unit error of 0.23 cells.

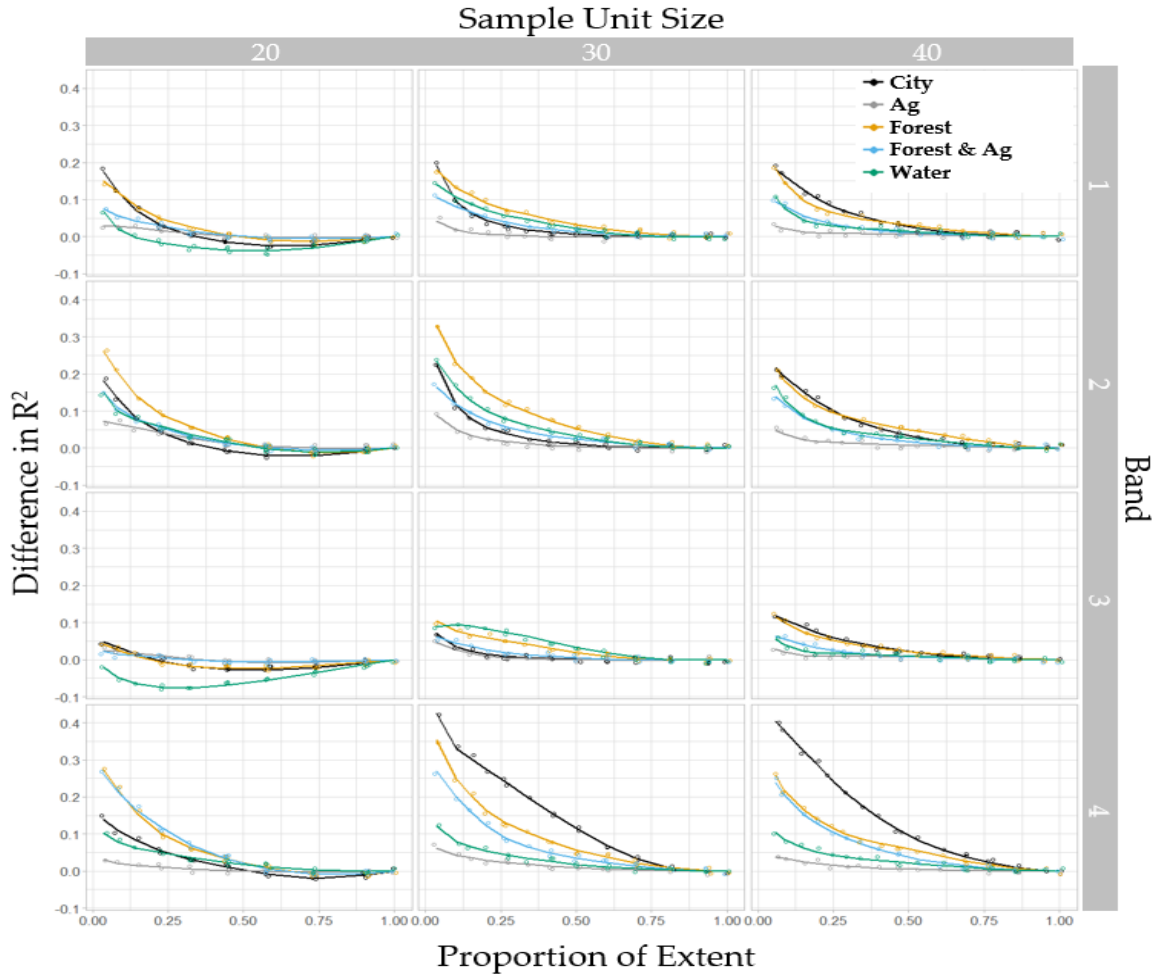


**Figure A2.** Stage I NAIP image regression statistics for varying sample unit sizes and spectral bands given average raster and GPS registration errors of 6 and 7 cells respectively.

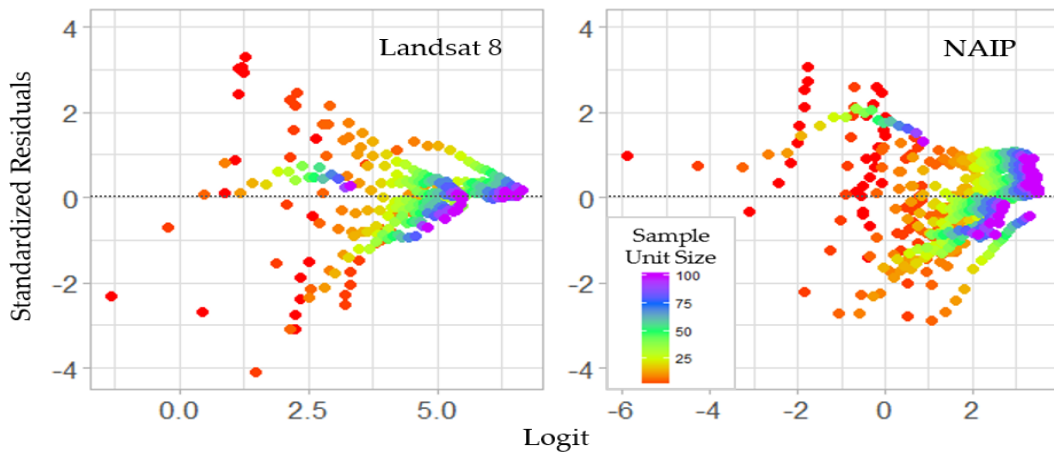


**Figure A3.** Reduction in the proportion of variation explained ( $R^2$ ) for Landsat 8 image by subsample intensity (Proportion of Extent), sample unit size, and spectral band using SYS 4 and  $P_{all}$  subsampling layout.





**Figure A4.** Reduction in the proportion of variation explained ( $R^2$ ) for NAIP images by subsample intensity (Proportion of Extent), sample unit size and spectral band using SYS 4 and  $P_{all}$  subsampling layout.



**Figure A5.** Scatter plot of standardized Residuals vs linear predictors (Logit) given block size for beta regression Landsat 8 and NAIP based models.

**Table A1.** Examples of remote sensing applications impacted by co-registration errors and the impact on model fit and estimates.

Application	Data Source	Impact on Model Fit and Estimates
Mapping forest characteristics using models derived from field data and remotely sensed data.	Landsat imagery	Attenuated estimates and reduction in model fit. The amount depends on the spatial correlation within the imagery, the co-registration error between imagery and field data, and the spatial extent of the field data observations (Table A2).
	NAIP imagery	Attenuated estimates and reduction in model fit. The amount depends on the spatial correlation within the imagery, the co-registration error between imagery and field data, and the spatial extent of the field data observations (Table A2).
	Other remotely sensed data.	Attenuated estimates and reduction in model fit. The amount depends on the spatial correlation within the imagery, the co-registration error between imagery and field data, and the spatial extent of the field data observations.
Change detection derived from multiple images of a given area.	Satellite and aerial based imagery	Attenuated estimates and reduction in model fit.
Image radiometric normalization	Satellite and aerial based imagery	Attenuated estimates and reduction in model fit.
Image segmentation	Attenuated outputs	Less variation in estimated values potentially reducing the accuracy of the segmentation process.
Practitioner use of attenuated spatial data products derived from field plots and remotely sensed imagery.	Attenuated outputs	Mean estimates derived from the entire surface will not be bias. Subsets of the derived surface will be biased and will either over estimate (values < mean) or under estimate (values > mean) the true values.

**Table A2.** Estimated reduction in  $R^2$  ( $\Delta R^2$ ) for Landsat 8 and NAIP imagery given equation 4, sample unit size, GMI value, and published horizontal image and GPS errors.

Source	Sample Unit Size (Cells Wide)	GMI	$\Delta R^2$
Landsat 8	3	0.8	0.067
Landsat 8	5	0.8	0.043
Landsat 8	9	0.8	0.026
Landsat 8	3	0.9	0.040
Landsat 8	5	0.9	0.026
Landsat 8	9	0.9	0.016
Landsat 8	3	0.95	0.030
Landsat 8	5	0.95	0.019
Landsat 8	9	0.95	0.012
NAIP	20	0.8	0.226
NAIP	30	0.8	0.166
NAIP	40	0.8	0.131
NAIP	20	0.9	0.148
NAIP	30	0.9	0.109
NAIP	40	0.9	0.088
NAIP	20	0.95	0.116
NAIP	30	0.95	0.087
NAIP	40	0.95	0.070

## References

1. Jensen, J. *Remote Sensing of the Environment: An Earth Resource Perspective*; Prentice Hall: Upper Saddle River, NJ, USA, **2000**; p. 544.
2. Bechtold, W.A.; Patterson, P.L.; [Editors]. The enhanced forest inventory and analysis program - national sampling design and estimation procedures. Gen. Tech. Rep. SRS-80. Asheville, NC: U.S. Department of Agriculture, Forest Service, Southern Research Station. **2005**; p. 85.
3. Omernik, J.; Griffith, G. Ecoregions of the conterminous United States: Evolution of a hierarchical spatial framework. *Environ. Manag.* **2014**, *54*, 1249–1266.
4. Gesch, D.; Oimoen, M.; Greenlee, S.; Nelson, C.; Steuck, M.; Tyler, D. The National Elevation Dataset. *Photogr. Eng. Remote Sens.* **2002**, *68*, 5–11.
5. Homer, C.; Dewitz, J.; Yang, L.; Jin, S.; Danielson, P.; Xian, G.; Coulston, J.; Herold, N.; Wickham, J.; Megown, K. Completion of the 2011 National Land Cover Database for the conterminous United States—Representing a decade of land cover change information. *Photogr. Eng. Remote Sens.* **2015**, *81*, 345–354.
6. Gandhi, G.; Parthiban, S.; Thummalu, N.; Christy, A. Ndvi: Vegetation Change Detection Using Remote Sensing and Gis—A Case Study of Vellori District. *Procedia Comput. Sci.* **2015**, *57*, 1199–1210.
7. LANDFIRE. Existing Vegetation Type Layer, LANDFIRE 1.1.0, U.S. Department of the Interior, Geological Survey. **2008**. Available online: <http://landfire.cr.usgs.gov/viewer/> (accessed on 28 April 2018).
8. Lowry, J.; Ramsey, R.; Boykin, K.; Bradford, D.; Comer, P.; Falzarano, S.; Kepner, W.; Kirby, J.; Langa, L.; Prior-Magee, J. *Southwest Regional Gap Analysis Project: Final Report on Land Cover Mapping Methods*; RS/GIS Laboratory, Utah State University: Logan, UT, USA, **2005**.
9. Escuin, S.; Navarro, R.; Fernandez, P. Fire severity assessment buy using NBR (Normalized Burn Ratio) and NDVI (Normalized Difference Vegetation Index) derived from LANDSAT TM/ETM images. *Int. J. Remote Sens.* **2008**, *29*, 1053–1073.
10. Davids, C.; Doulgeris, A. Unsupervised change detection of multitemporal Landsat imagery to identify changes in land cover following the Chernobyl accident. In Proceedings of the Geoscience and Remote Sensing Symposium, Barcelona, Spain, 8–11 July **2008**; pp. 3486–3489.
11. Weng, Q.; Fu, P.; Gao, F. Generating daily land surface temperature at Landsat resolution by fusing Landsat and MODIS data. *Remote Sens. Environ.* **2014**, *145*, 55–67.
12. Turner, M.; Gardner, G.; O'Neil, R. *Landscape Ecology in Theory and Practice: Pattern and Process*; Springer: New York, NY, USA, **2001**; p. 406.
13. Pietsch, M. Contribution of connectivity metrics to the assessment of biodiversity—Some methodological considerations to improve landscape planning. *Ecol. Indic.* **2018**, *94*, 116–127.
14. Stancioiu, P.; Nita, M.; Lazar, G. Forestland connectivity in Romania—Implications for policy and management. *Land Use Policy* **2018**, *76*, 487–499.
15. Lechner, M.; Harris, R.; Doerr, V.; Doerr, E.; Drielsma, M.; Lefroy, E. From static connectivity modelling to scenario-based planning at local and regional scales. *J. Nat. Conserv.* **2015**, *28*, 78–88.
16. Shafer, G. 2015. Land use planning: A potential force for retaining habitat connectivity in the Greater Yellowstone Ecosystem an dBeyon. *Glob. Ecol. Conserv.* **2015**, *3*, 256–278.
17. Ersoy, E.; Jorgensen, A.; Warren, P. Identifying multispecies connectivity corridors and the spatial pattern of the landscape. *Urban For. Urban Green.* **2018**, doi:10.1016/j.ufug.2018.08.001.
18. Hogland, J.; Anderson, N. Function Modeling Improves the Efficiency of Spatial Modeling Using Big Data from Remote Sensing. *Big Data Cogn. Comput.* **2017**, *1*, 3.
19. Hogland, J.; Anderson, N.; St. Peters, J.; Drake, J.; Medley, P. Mapping Forest Characteristics at Fine Resolution across Large Landscapes of the Southeastern United States Using NAIP Imagery and FIA Field Plot Data. *Int. J. Geo-Inf.* **2018**, *7*, 140, doi:10.3390/ijgi7040140.
20. Masek, J.; Hayes, D.; Hughes, M.; Healey, S.; Turner, D. The role of remote sensing in process-scaling studies of managed forest ecosystems. *For. Ecol. Manag.* **2015**, *355*, 109–123.
21. Hogland, J.; Anderson, N.; Chung, W. New Geospatial Approaches for Efficiently Mapping Forest Biomass Logistics at High Resolution over Large Areas. *Int. J. Geo-Inf.* **2018**, *7*, 156, doi:10.3390/ijgi7040156.
22. St. Peters, J.; Hogland, J.; Anderson, N.; Drake, J.; Medley, P. Fine resolution probabilistic land cover classification of landscapes in the southeastern United States. *Int. J. Geo-Inf.* **2018**, *7*, 107.
23. Graves, S.; Caughlin, T.; Asner, G.; Bohlman, S. A tree-based approach to biomass estimation from remote sensing data in a tropical agricultural landscape. *Remote Sens. Environ.* **2018**, *218*, 32–43.

24. Faga, M.; Morton, D.; Cook, B.; Masek, J.; Zhao, F.; Nelson, R.; Huang, C. Mapping pine plantation sin the southeastern U.S. using structural, spectral, and temporal remote sensing data. *Remote Sens. Environ.* **2018**, *216*, 415–426.
25. Cheshner, A. The effect of measurement error. *Biometrika* **1991**, *78*, 451–462.
26. Fuller, W. *Measurement Error Models*; Wiley: New York, NY, USA, **1987**; p. 500.
27. Greenberg, J.; Dobrowski, S.; Ustin, S. Shadow allometry: Estimating tree structural parameters using hyperspatial image analysis. *Remote Sens. Environ.* **2005**, *97*, 15–25.
28. Sheridan, R.; Popescu, S.; Gatzliolis, D.; Morgan, C.; Ku, W. Modeling Forest Above ground Biomass and Volume Using Airborne LiDAR metrics and Forest Inventory and Analysis Data in the Pacific Northwest. *Remote Sens.* **2015**, *7*, 229–255, doi:10.3390/rs70100229.
29. Frazer, G.; Magnussen, S.; Wulder, M.; Niemann, K. Simulated impact of sample plot size and co-registration error on the accuracy of and uncertainty of LiDAR-derived estimates of forest stand biomass. *Remote Sens. Environment* **2011**, *115*, 636–649.
30. Bobakken, T.; Naesset, E. Assessing effects of positioning errors and sample plot size on biophysical stand properties derived from airborne laser scanner data. *Can. J. For. Res.* **2009**, *39*, 1036–1052.
31. Saarela, S.; Schnell, S.; Tuominen, S.; Balazs, A.; Hyypä, J.; Grafstrom, A.; Stahl, G. Effects of positional errors in model-assisted and model-based estimation of growing stock volume. *Remote Sens. Environ.* **2016**, *172*, 101–108.
32. Van Niel, T.; McVicar, T.; Li, L.; Gallant, J.; Yang, Q. The impact of misregistration on SRTM and DEM image differences. *Remote Sens. Environ.* **2008**, *112*, 2430–2442.
33. Moran, P. Notes on Continuous Stochastic Phenomena *Biometrika* **1950**, *37*, 17–23, doi:10.2307/2332142.
34. Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2014. Available online: <http://www.R-project.org/> (accessed on 28 April 2018).
35. Hijmans, R.; Etten, J. Raster: Geographic Analysis and Modeling with Raster Data. R Package Version 2.0-12, 2012. Available online: <http://CRAN.R-project.org/package=raster> (accessed on 24 September 2018).
36. Pebesma, E. Multivariable geostatistics in S: The gstat package. *Comput. Geosci.* **2004**, *30*, 683–691.
37. Gräler, B.; Pebesma, E.; Heuvelink, G. Spatio-Temporal Interpolation using gstat. *R J.* **2016**, *8*, 204–218.
38. Landsat. Landsat Project Description, 2018. Available online: <https://landsat.usgs.gov/landsat-project-description> (accessed on 27 April 2018).
39. National Agriculture Imagery Program [NAIP]. National Agriculture Imagery Program (NAIP) Information Sheet. Available online: [http://www.fsa.usda.gov/Internet/FSA\\_File/naip\\_info\\_sheet\\_2013.pdf](http://www.fsa.usda.gov/Internet/FSA_File/naip_info_sheet_2013.pdf) (accessed on 14 May 2014).
40. Cressie, N. *Statistics for Spatial Data Revised Edition*; Published by Wiley Classics Library, John Wiley, **2015**; p. 928.
41. Loveland, T.; Irons, J. Landsat 8: The plans, the reality, and the legacy. *Remote Sens. Environ.* **2016**, *185*, 1–6.
42. Beyerhelm, C. *Head-to-Head Comparison of Four SiRF-Based GPS Receivers*; **2009** Report; Available online: <https://www.fs.fed.us/database/gps/documents/SiRFComp.pdf> (accessed on 24 September 2018).
43. Forest Inventory and Analysis Program [FIA]. *Forest Inventory and Analysis National Core Field Guide: Field Data Collection Procedures for Phase 2 Plots. Version 6.0. Vol. 1*; Internal Report; U.S. Department of Agriculture Forest: Washington, DC, USA, **2012**. Available online: [http://www.fia.fs.fed.us/library/field-guides-methods-proc/docs/2013/Core%20FIA%20P2%20field%20guide\\_6-0\\_6\\_27\\_2013.pdf](http://www.fia.fs.fed.us/library/field-guides-methods-proc/docs/2013/Core%20FIA%20P2%20field%20guide_6-0_6_27_2013.pdf) (accessed on 5 June 2014).
44. Cribari-Neto, F.; Zeileis, A. Beta Regression in R. *J. Stat. Softw.* **2010**, *34*, doi:10.18637/jss.v034.i02
45. Akaike, H. Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*; Petrov, B.N., Csake, F., Eds.; Springer: New York, NY, USA, **1973**; pp. 267–281.
46. Akaike, H. A new look at the statistical model identification. *IEEE Trans. Autom. Control* **1974**, *19*, 716–723.
47. Patterson, P.L.; Williams, M.S. Effects of registration error between remotely sensed and ground data on estimators of forest area. *For. Sci.* **2003**, *49*, 110–118.
48. Zhang, M.; Lin, H.; Zeng, S.; Li, J.; Shi, J.; Wang, G. Impacts of plot location errors on accuracy of mapping and scaling up aboveground forest carbon using sample plot and landsat tm data. *IEEE Geosci. Remote Sens. Lett.* **2013**, *10*, 1483–1487.
49. Frost, C.; Thompson, S. Correcting for regression dilution bias: Comparison of methods for a single predictor. *J. R. Statist. Soc. A* **2000**, *163*, 173–189.

## Chapter 3

# Improving estimates of natural resources using model-based estimators: impacts of sample design, estimation technique, and strengths of association

**Abstract:** Natural resource managers need accurate depictions of existing resources to make informed decisions. The classical approach to describing resources for a given area in a quantitative manner uses probabilistic sampling and design based-inference to estimate population parameters. While probabilistic designs are accepted as being necessary for design based-inference, many recent studies have adopted non-probabilistic designs that do not include elements of random selection or balance and have relied on models to justify inferences. While common, model-based inference alone assumes that a given model accurately depicts the relationship between response and predictors across all populations. Within complex systems, this assumption can be difficult to justify. Alternatively, models can be trained to a given population by adopting design-based principles such as balance and spread. Through simulation, we compare estimates of population totals and pixel-level values using linear and nonlinear model-based estimators for multiple sample designs that balance and spread sample units. Findings indicate that model-based estimators derived from samples spread and balanced across predictor variable space reduce the variability of population and unit-level estimators. Moreover, if samples achieve approximate balance over feature space, then model-based estimates of population totals approached simple expansion-based estimates of totals. Finally, in all comparisons made, improvements in estimation were achieved using model-based estimation over design-based estimation alone.

Keywords: sample design; model-based estimation; spread; balance

## 1. Introduction

Managers of natural resources need accurate information describing the resources they manage to make informed decisions. Owing to high acquisition costs, managers typically employ sampling and estimation techniques to describe various aspects of a given population. Additionally, to increase estimation accuracy, ancillary data, such as remotely sensed data, can be used to improve population estimates through specification and application of models that characterize how the resources of interest vary as a function of the ancillary data (e.g., [1-3]). Whether or not such models are employed or if simple expansion based techniques are used to estimate population parameters, sample design plays an important role in estimation accuracy [4, 5].

Within the context of estimating forest characteristics from remotely sensed data, less emphasis is generally placed on rigorous sample design to train models than when validating models [6]. In large part this discrepancy stems from the conditions required for design versus model-based inference [1, 2]. As Gregoire [5] describes, “in the design-based framework, the population is regarded as fixed whereas the sample is regarded as a realization of a stochastic process”. Conversely the model-based framework assumes, “that the values  $y_1, \dots, y_n$  are regarded as realizations of random variables  $Y_1, \dots, Y_n$  and hence the population is a realization of a random process”. The primary difference being, “in the model-based approach [inference] stems from the model, not from the sampling design”. While some rely on the availability of models as justification for deviating from probabilistic designs when drawing inferences, it is critical to remember that with natural systems we seldom understand or can collect all the information needed to build models that completely describe the complexities of those systems. This can lead to model misspecification, localized deviations in relationships, and more generally to models that do not describe a specific population well. Therefore, sample designs used to calibrate models that include randomness in the selection process are critical to developing models that can be used to estimate population and subpopulation parameters. Moreover, probabilistic samples that also capture characteristics such as balance and

spread across feature space can reduce the variability of population estimates while more consistently including sample units across the entirety of feature space.

However, many mapping projects employ non-probabilistic sample designs when calibrating predictive models [6] or use designs that may not fully capture the spread of predictor variables and that may be unbalanced with regard to the population. Furthermore, some authors have argued that sample units should only contain pure homogenous examples of a given class [7-12] and have developed sampling protocols to ensure that outcome. Compared to probabilistic sampling designs, samples of homogeneous units can be expected to estimate lower error rates when calibrating and validating models. However, such metrics can only be attributed to the class of units targeted for sampling and typically cannot be generalized to the rest of the population [13, 14].

Stehman [13, 15, 16] describes multiple probabilistic sampling designs and outlines their associated strengths and weaknesses as they relate to map accuracy. Tille and Wilhelm [17], Grafstrom et al. [3], and Steven and Olsen [18] further describe the importance of probabilistic designs while highlighting concepts of balance and spread as they relate to the precision of estimators. Balance and spread in this case refers to characteristic of a sample with regard to auxiliary variables. Specifically, a balanced sample is defined as the condition when sample means or totals of auxiliary variables equal or approximately equal population means or totals [17]. Whereas a sample that is spread in auxiliary space refers to how well sample units are dispersed across the auxiliary values of that population [18]. For population estimates, balance of a sample draws on the strength of the relationship between response and auxiliary variables and recognizes that when a sample captures the true mean or total of an auxiliary variable, it will also capture the true mean or total of the response variable. Likewise, a sample that is well spread across auxiliary variables has the advantage of being balanced [19] while also capturing the variability in auxiliary variables. While often described for expansion-based estimators, these arguments are equally important as they relate to samples collected to develop models to support estimation [6, 17, 20-22]. Nevertheless, many researchers disregard these warnings and build models derived from unbalanced, non-probabilistic sampling schemes to estimate characteristics of a given population.

Reasons for deviating from a probabilistic study design typically stem from logistical constraints and cost of implementation. However, using predicted values derived from models developed from non-probabilistic designs to describe complex natural systems such as a forested landscape can be highly questionable. Questions pertaining to how sample units should be spread across multiple dimensions of feature and geographical space and subsequent impacts on estimator accuracy and inference are at the very forefront of understanding how resource assessments and maps can be used to inform decision making. In this study, we investigate these questions through a series of simulations that compare and contrast estimates of population totals and pixel values obtained under varying sampling designs. We also evaluate the relative impact of sampling design on estimators derived without use of ancillary data and from various linear, parametric, non-linear, and non-parametric models. Specifically, in this study we 1) evaluate the relative accuracy of alternative model-based estimators of unit values and population totals across populations where the form and/or strength of associations vary, 2) evaluate the impact of probabilistic designs that spread and balance samples over geographic or feature space on the accuracy of those estimators, 3) demonstrate the potential impact of using a probabilistic sampling design taken from a subset of geographic space on the performance of those estimators and 4) assess the impact of misspecified models on the accuracy of estimation.

## **2. Theoretical Background**

In this study we assess expansion and model-based estimation procedures applied to populations constructed according to distinct functional relationships between response and predictor variables but tempered by normally distributed errors of varying magnitudes. While fundamentally different, expansion and model-based estimation techniques can be used to estimate population parameters [23]. For expansion-based methods, parameter estimators are derived from

elements of a probabilistic sample design. In this approach, the population is regarded as fixed and variations in parameter estimates are due to randomness within the sample design [4]. In contrast, model-based estimators of population parameters are derived from a model of how the response variables change across the population. In this approach, the emphasis is on estimating model parameters which can then be indirectly used to estimate either parameters of a given realization of the model, or the model-expectations thereof [23]. Contrary to the expansion-based methodology, the model-based approach is not necessarily tied to a specific population but instead assumes that a given population is a realization of some random process, meaning that the emphasis of the model-based approach is on the relationship between a variable of interest (response) and some other variable(s) (predictors). Given that relationship, model error, and the predictor variable values within a specific population, one can then estimate population parameters.

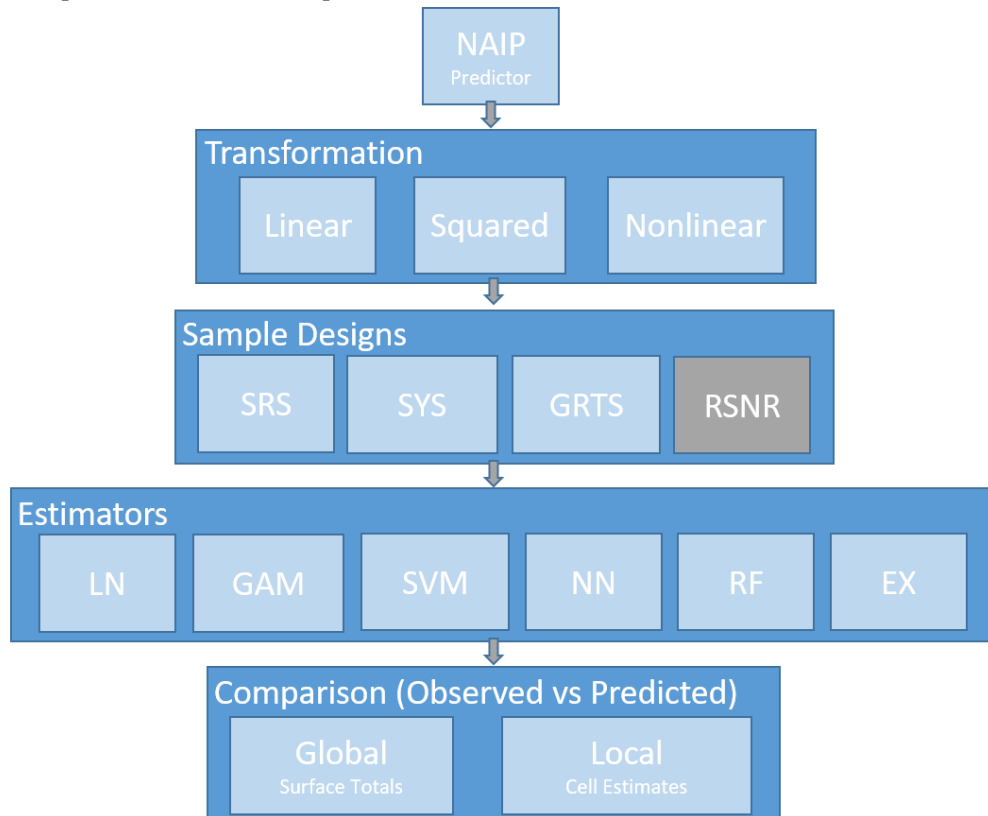
While models do not necessarily need to be tied to a given population, it is the case that within natural systems the underlying shapes and forms of the relationships between response and predictor variables vary and may be unknown and noisy. Therefore, model development is often based on a specific population (e.g., [24, 25]), making models uniquely calibrated to a given time and place. Though uniquely calibrated, different modeling techniques make different assumption with regard to the relationships between response and predictor variables. For many natural systems these assumption may be difficult to meet and can lead to a model that is misspecified. To assess the impact of potential model misspecification on estimating population and subpopulation parameters, we use simulated landscapes with known functional relationships and introduced error (Figure 1). The functional shape of relationships evaluated in our simulations include linear, quadratic, and nonlinear distributions with normally distributed errors and various amounts of introduced noise (section 3.1). Additionally, estimation techniques are evaluated under various sample designs to assess the impact of spread and balance on parameter estimation (section 3.2). Expansion and model-based estimators were chosen based on their popularity, underlying assumptions, and flexibility in capturing various relationships among response and predictor variables. Modeling techniques evaluated include linear regression (LN), neural networks (NNs), support vector machines (SVMs), random forests (RF), and generalized additive models (GAMs).

In terms of complexity, expansion estimators are the simplest, followed by LN, NNs, SVMs, RF, and GAMs. Expansion estimators are limited to a single estimate such as mean or total for a population of interest [4]. Conversely, linear regression models can estimate the mean response value of a class of pixels for given values of predictor variables, but are obviously limited in their ability to describe nonlinear relationships and can produce estimates outside the range of values observed in a sample [26]. NNs, a machine learning technique, can represent nonlinear associations by introducing activation functions and weighting schemes of linear coefficients [27]. SVMs use kernel functions to identify subsets of the data within multidimensional feature space that describe the relationship between response and predictor variables [27]. RFs allow for discontinuities in the response function across feature space by implementing classification and regression trees (CART) and incorporating bagging and randomness into model training. RF ultimately rely on ensembles of CART models to capture relationships between response and predictors, and can yield only estimates limited to the range of observed response values [28]. GAMs are a flexible modeling technique that assumes additivity between response and predictor variables, and that response values change smoothly over feature space. GAMs allow for nonlinear, nonparametric relationships between response and predictor variables [29] and can explicitly account for non-Gaussian error distributions.

Of particular interest with regard to expansion and model-based estimators is that an expansion estimator can be thought of as a model-based estimator with only an intercept parameter if an equal probability design is used and if the presumed model incorporates an independently identically distributed error structure. That is, if the same sample is used by an expansion estimator to calibrate an intercept only model, the population estimates from the expansion and model-based estimators will be the same. Furthermore, when models are calibrated with predictors that are unrelated to a given response, expansion and model-based estimates will also be the same. Owing to this, within



our simulation we predict that as the amount of model error (noise) introduced into the relationship between response and predictors is increased, model and expansion-based estimates will converge. Conversely, as the amount of noise introduced between response and predictor decreases, the estimated values from the model-based estimators will be less variable than the expansion-based estimates. Similarly, within our simulations we anticipate that when underlying model relationships are misspecified or calibrated in the absence of important correlated data, model-based estimates will converge with expansion-based estimates. Moreover, we predict that samples drawn from probabilistic designs that are spread and balanced across predictor variable space produce better estimates of the population and subpopulations than samples drawn in a manner that may not be balanced or spread within feature space.



**Figure 1.** Diagram of simulations and workflow to determine the impact of sample designs on model and design-based estimates. The gray box in sample design denotes a probabilistic sample design for a subset of geographic space. NAIP = National Agriculture Imagery Program imagery; SRS = simple random sample, SYS = systematic random sample, GRTS = generalized random tessellation stratified, RSNR = simple random sample near roads, LN = linear model; GAM = generalized additive models, SVM = support vector machines, NN = neural networks, RF = random forests, EX = Horvitz Thompson expansion estimator.

### 3. Materials and Methods

#### 3.1 Simulated raster surfaces

A spatial subset extracted from a National Agricultural Imaging Program (NAIP) [30] image located in the panhandle of Florida, USA serves as our base dataset for all derived raster surfaces used within our simulations. This subset covers approximately 6 km<sup>2</sup> of a mixed forested/urban landscape (Figure 2). The four spectral bands of the NAIP image ( $x_i$ ;  $i = 1, 2, 3, 4$ ) were transformed to produce three distinct continuous surfaces (SC1, Figure 2). The first SC1 surface was obtained by averaging the band values for each pixel:

$$\mu_{1j} = 0.25 \sum_{i=1}^4 x_{ij} \quad (1)$$

The second SC1 surface was obtained as the square of the first,

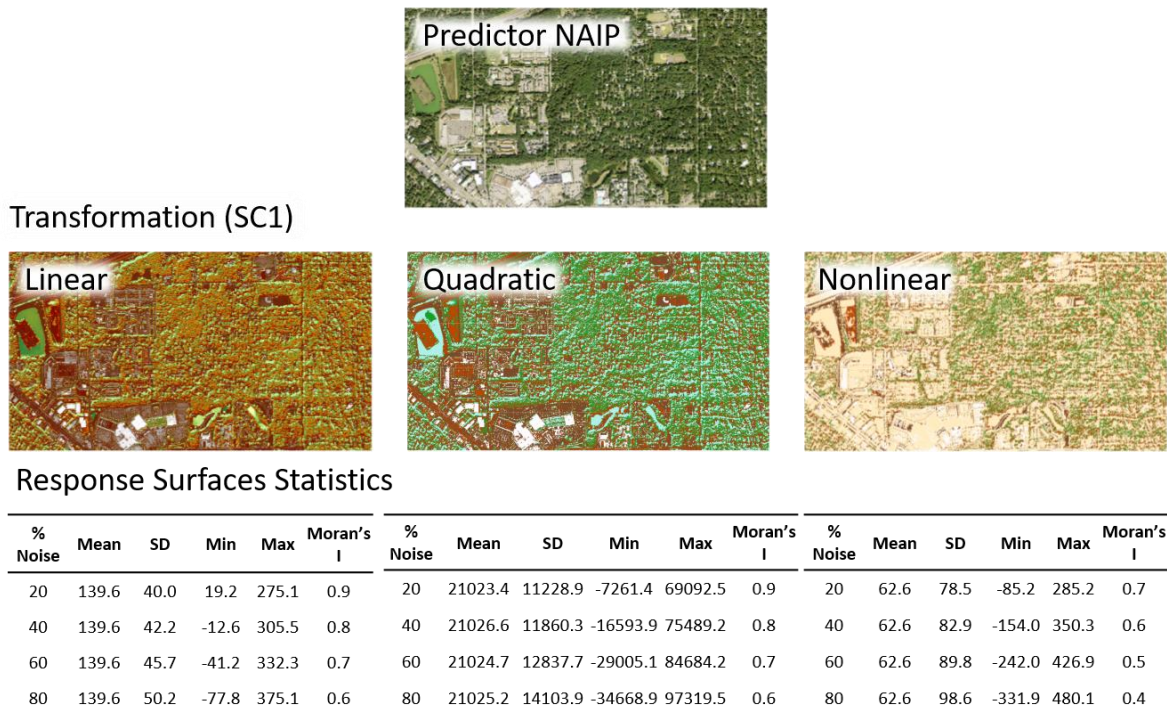
$$\mu_{2j} = \mu_{1j}^2 \quad (2)$$

and thus is a quadratic function of the individual band values and their cross-products. Finally, a discontinuous, nonlinear SC1 surface was created by adding an additional logical query on the NAIP band 4 cell values:

$$\mu_{3j} = \lambda_j \mu_{1j} \quad (3)$$

where  $\lambda_j$  is an indicator variable taking the value 1 if the value of band 4 for pixel  $j$  falls between 170 and 210, and is 0 otherwise.

For each SC1 surface (Linear, Quadratic, and Nonlinear), normally distributed errors were added to create response surfaces  $y_{kj} = \mu_{kj} + \epsilon_j$  (Figure 2), where the  $\epsilon_j$  are independent, identically distributed errors. Normally distributed errors were generated using a standard deviation proportional to the standard deviation of pixel values within a given SC1 and a mean of zero. The constants of proportionality modifying the normal standard deviation parameters were set to 20%, 40%, 60%, and 80% to provide an equal distant range of noise added to each relationship and to evaluate the impact of noise on model estimation. The realized errors produced a homoscedastic surface with error values centered at zero before being added to SC1 surfaces. In total, 12 unique continuous response surfaces were created, with various shapes and amounts of error (% noise).



**Figure 2.** Depiction of the NAIP, Linear, Quadratic, and Nonlinear transformations (SC1). Normally distributed error terms are additive (% Noise) and based on SC1 surface standard deviation statistics. Mean, standard deviation, minimum, maximum, and global Moran's I [31] (rook neighborhood) statistics are presented for each response surface by transformation type and noise level.

### 3.2 Sample designs

To evaluate the impact of sample designs on estimation, we compared population and raster cell estimates for each of our response surfaces using four different sampling designs. Sample designs included simple random sampling (SRS), systematic random sampling (SYS), modified Generalized Random Tessellation Stratified sampling (GRTS; [18, 32]), and randomly selecting sample units next

to roads (RSNR). In our simulation, SRS selects locations across the spatial domain of the image, independently and uniformly at random. SYS spreads sample units evenly across the spatial domain of the image by distributing locations over a square grid of fixed orientation. The dimensions of each square grid cell was calculated as the square root of the total image area divided by a sample size of 50. A fixed sample size for SYS was achieved by selecting a random start within the bottom east square grid cell and systematically selecting locations based on the grid cell dimensions. Owing to the systematic nature of the design and the dimension of each grid cell, some SYS samples had fewer or more than 50 sample units. For SYS samples with more than 50 location selected, location were randomly removed until the sample size reached 50. For SYS samples with less than 50 locations selected, random locations across the spatial domain of the image were added to the sample until the sample size reached 50. GRTS spreads and balances samples in auxiliary space based on the NAIP spectral values (bands 1-4). To achieve this, we selected an initial SRS of 100,000 locations within the NAIP image, extracted cell values from each band at those locations, and performed a principal components analysis (PCA) using the bands' correlation matrix. Using the loadings of the first two components of the PCA, we then transformed the 100,000 sampled NAIP cell values to bivariate component scores. These were then input into the GRTS algorithm [32] to select a sample spread and balanced within a two-dimensional PCA score space. RSNR uses Tiger Road files [33] and a 50 m buffer around roads to subset the population and then randomly select sample unit locations within those subsets. Sample sizes for model calibration and parameter estimation were held constant at 50 for each comparison.

### 3.3 Model calibration and estimation

LN, GAM, SVM, NN, and RF models were trained separately for each sample drawn from the 12 response surfaces and 4 sample designs. For LN models, ordinary least squares regression was used to relate response and predictor variables [34]. For GAMs, the Gaussian family with an identity link and additive thin plate penalized regression splines were used to relate response and predictor variables [35]. SVM models used Epsilon Regression and a Radial Basis kernel (Gaussian) [36]. NNs fit a single layer hidden linear model (least squares) with a size parameter equal to half the sample size and a decay parameter equal to 1/10 the size parameter [37]. RF models averaged the results of 20 regression trees that each randomly selected one predictor variable at each training node, for 66% of the data [38]. All model calibrations assumed independent, homoscedastic errors.

Model predictor variables initially included all four NAIP band cell values. That is, all information used to generate the response surfaces were made available in the modeling process as predictor variables. Below, we refer to these as fully-specified models. A second class of models (partially specified models) utilized only the first three NAIP bands as predictors. This represents the common situation in which the available predictor variables do not fully describe the response function.

Once trained, model and NAIP cell values were used to create prediction surfaces of estimated raster cell values. Estimates of the population total ( $\hat{t}_y$ ) were calculated by aggregation

$$\hat{t}_y = \sum_{i=1}^N \hat{y}_i \quad , \quad (4)$$

where  $\hat{y}_i$  is an estimated pixel value obtained from a calibrated model of one of the above types and N is the total number of pixels in the population.

Additionally, from the response values alone, each sample provided an expansion estimate of the population total obtained simply from the per-pixel sample mean expanded by the population size:

$$\hat{t}_y = \frac{N}{n} \sum_{i=1}^n y_i \quad (5)$$

where n is the sample size (50).

### 3.4 Evaluation

For each sample design ( $d$ ) and response surface ( $k$ ), 100 samples (iterations) were drawn and used to calibrate models and estimate surface characteristics. The spread of each sample ( $B$ ) was measured using the approach described in [19, 39]. Briefly, Mahalanobis distance [40] was used to identify Voronoi polytopes ( $p_i$ ) around each selected pixel of a particular sample in multidimensional predictor space or in geographic space. Inclusion probabilities ( $n/N$ ) were then summed within each polytope and the variance of these sums were then calculated:

$$B = \frac{1}{n} \sum_{i \in s} (v_i - 1)^2, \quad (6)$$

where  $v_i$  is the sum of inclusion probabilities in  $p_i$ .

For each response surface, sample design, and estimation technique, estimates were compared against the corresponding response surface total or against individual cell values. For a global perspective, estimates of the response surface totals were recorded and compared against actual totals. For a local perspective, estimated cell values were recorded and the RMSE was calculated across cells as follows:

$$RMSE = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2. \quad (7)$$

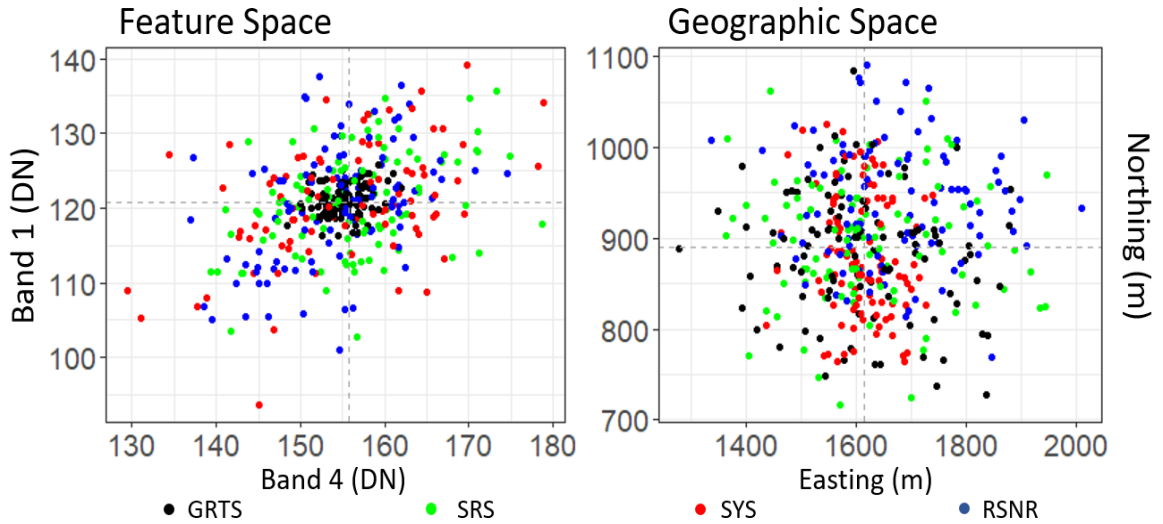
RMSE was also calculated for the expansion estimator reflecting the fact that the expansion estimator (scaled by  $N$ ) is the only available estimate of cell values as follows:

$$RMSE = \frac{1}{N} \sum_{i=1}^N (\bar{y} - y_i)^2, \quad (8)$$

where  $\bar{y}$  is the simple sample mean. While similar to the variance of the expansion estimator, this  $RMSE$  expression is evaluated over the entire population and is a measure of within response-surface variability.

## 4. Results

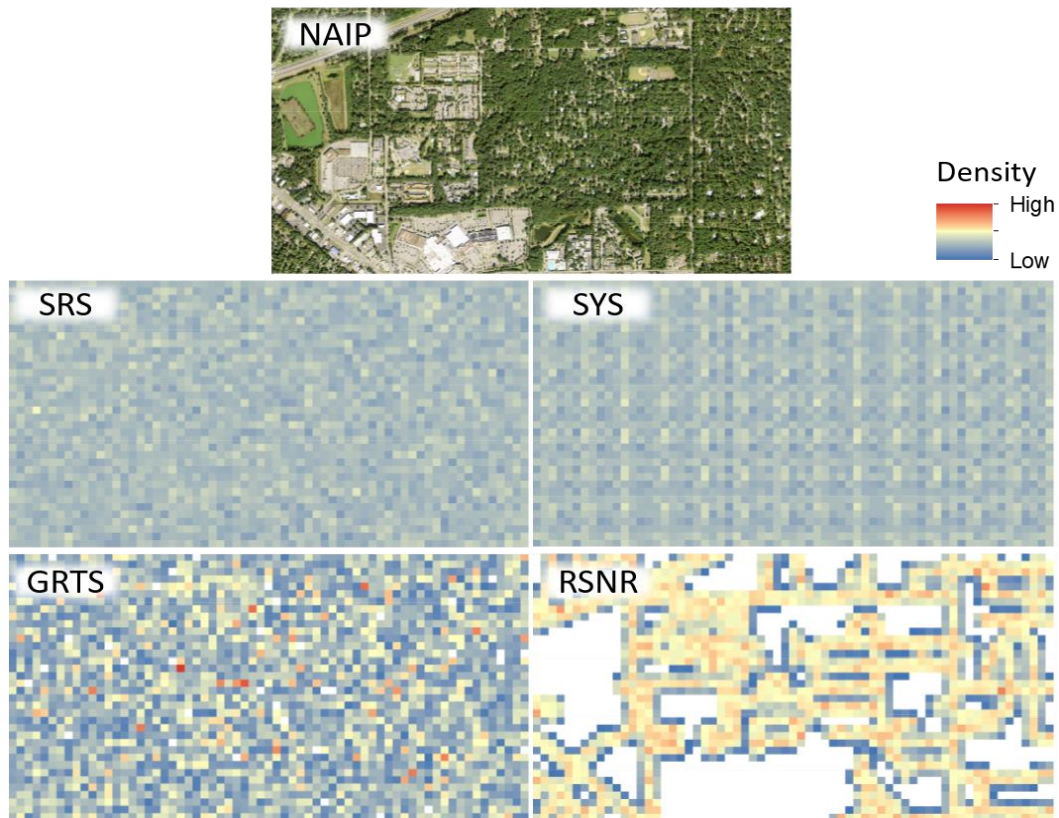
### 4.1 Allocation of sample units



**Figure 3.** Distribution of sample mean digital number (DN) values of bands 1 and 4 (left panel; Feature Space) and sample mean northing and easting values (right panel; Geographic Space) for 100 samples obtained from generalized random tessellation stratified (GRTS, black), simple random sampling (SRS, green), systematic random sampling (SYS, red), and random sampling near roads (RSNR, blue) designs. Gray dashed lines identify population mean values. A sample size of 50 was used for each of the 100 samples.

Of all the sample designs considered, only GRTS utilized information from feature space. Although it used only the first two principal components of the 4 bands, these accounted for approximately 97% of the total variation in the NAIP image. Within feature space, GRTS sample means were approximately balanced, clustering near the population mean values for each band (Figure 3). By contrast, samples drawn by SRS, SYS and RSNR designs were unbalanced, with individual sample means varying substantially from band averages (Figure 3). In geographic space, GRTS, SRS, and RSNR had similarly dispersed distributions of sample means while the distribution of sample means for the SYS design was tighter (Figure 3). Imbalance of SYS samples in geographic space was higher than anticipated, possibly as a result of the sample adjustments made to ensure a constant sample size of 50 units.

While, the RSNR design only selected sample units within 50 m of roads inside the study area and appeared to under represent impervious surfaces such as industrial areas and buildings (Figure 4), mean values of NAIP bands and northing and easting closely matched population parameters (Figure 3). Moreover, even though no sample units were selected further than 50 m from a road in the RSNR design, sample unit locations appeared to be generally spread across the study area in both feature and geographic space (Figure 3). While the areas within 50 m of a road represent a subset of the geographic domain of the study area, the spatial extent of that subset still spanned a broad distribution of feature space (Figure 3).

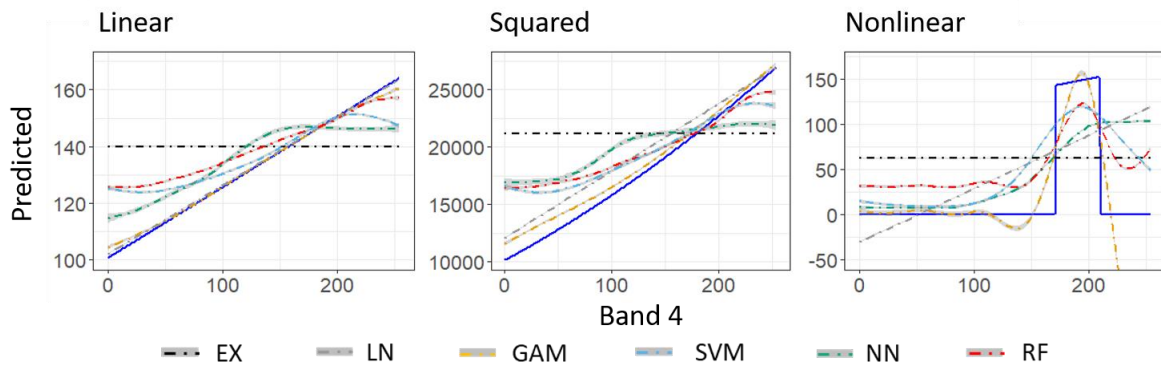


**Figure 4.** Illustration of spatial distribution (density) of sample unit locations for each sample design across the study area. The NAIP image is used for reference; the density raster surfaces depict the frequency of selected cell locations for samples using the SRS, SYS, GRTS, and RSNR designs. To enhance the display of density surfaces, each surface was created by drawing 2000 random samples (sample size 50) for each design and counting the number of sample units falling within each density raster cell (50 m grain size). White cells within the density raster surfaces identify cells where no sample units were selected. Note, the spatial clustering in sample density depicted by high density values (orange to red areas) for both GRTS and RSNR designs. SRS = simple random sample; SYS = systematic random sample; GRTS generalized random tessellation stratified; RSNR = random sample near roads.

#### 4.2 Model calibration and functional relationships

While trends in population estimates were generally consistent across response surfaces, the most interesting results occurred when estimators were applied to the nonlinear SC1 response surfaces. For these surfaces, the RF estimator would presumably have an advantage over expansion, LN, SVM, NN, and GAM estimators given that the latter techniques presume smooth response functions over feature space and are limited in their ability to capture the discontinuities in these response surfaces. Figure 5 depicts the averaged fitted estimates over the 100 SRS iterations for a slice of feature space (40% introduced normally distributed errors). Relatively speaking, fully-specified GAM, RF, and SVM estimators were able to reproduce the rise and drop of the response surface cell values. While NN estimators partially capture the rise in values, they were not able to capture the subsequent fall in those values in nonlinear relationships (Figure 5). As expected, the linear estimators only allowed for constant rates of change. They could therefore describe patterns in the linear SC1 surfaces but failed utterly to describe characteristics of the nonlinear SC1 surfaces.

These aspects of model performance were consistent across sample designs and levels of introduced noise. As anticipated, when more error was introduced into the response surfaces, variability in estimates increased. When comparing estimation techniques across these slices of SC1 response surface values, fully-specified GAMs, RFs, and SVMs consistently outperformed other estimation approaches (Figure 5).

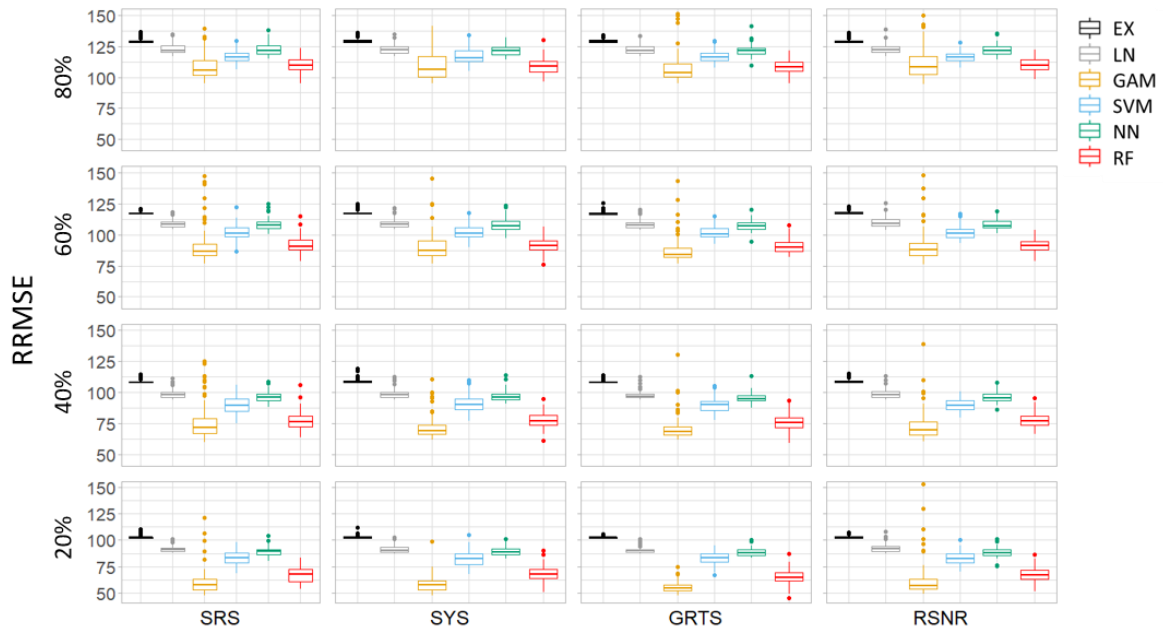


**Figure 5.** Depiction of functional relationships of fully-specified estimators and SC1 surface values (blue line and y-axis) for a slice of predictor variable space where NAIP cell values were held constant at the mean for bands 1-3 while band 4 cell values (x-axis) were allowed to vary. The amount of random noise introduced into the relationships between predictor and SC1 surfaces was 40% of the SC1 surface standard deviation. Sample design used to generate models was simple random sampling. The blue solid line corresponds to the actual SC1 values within this slice of predictor variable space while the dashed colored lines correspond to averaged expansion and model based estimates. The grey shaded regions around each estimation technique correspond to the 99% confidence interval given the 100 iterations performed. Sample size for expansion estimates and model calibration was 50. Ex = expansion; LN = linear; GAM = general additive model; SVM = support vector machine; NN = neural networks; RF = random forests.

#### 4.3 Estimates using NAIP bands as predictors

As expected, sample design had an impact on the accuracy of estimated pixel values and population totals (Figures 6 and 7). While we anticipated that the RSNR design would produce biased estimates, bias was minor in estimated values (Figure 7). Moreover, across all response surfaces, similar patterns in RMSE and variability in the estimated population totals were observed. Generally, as the proportion of noise increased in the response surfaces, RMSE and the variability in population estimates increased (Figures 6 and 7). Because results and trends were similar for linear, squared, and nonlinear response surfaces, we primarily present results for the nonlinear (most complex) response surfaces within this section.

For the expansion estimators, RMSE varied little across iterations within nonlinear response surfaces but consistently exceeded the RMSEs of the fully specified model-based estimators (Figure 6). While there did appear to be minor improvements in RMSE for the GRTS design, it is difficult to clearly rank sample designs with regards to RMSE in these figures. Generally, though, GRTS and SYS designs had less variation in RMSE than the other designs (Table 1).



**Figure 6.** Matrix of RMSE divided by SC1 image pixel standard deviation (RRMSE) boxplots for expansion and fully-specified model-based estimates derived from 100 iterations of nonlinear trend response surfaces. Response surfaces incorporated random errors at 20%, 40%, 60%, and 80% of the total nonlinear SC1 image standard deviation. Column and row titles identify sample designs and proportion of SC1 standard deviation used in response surfaces. Ex = expansion; LN = linear; GAM = general additive model; SVM = support vector machine; NN = neural networks; RF = random forests; SRS = simple random sample; SYS = systematic random sample; GRTS generalized random tessellation stratified; and RSNR = random sample near roads.

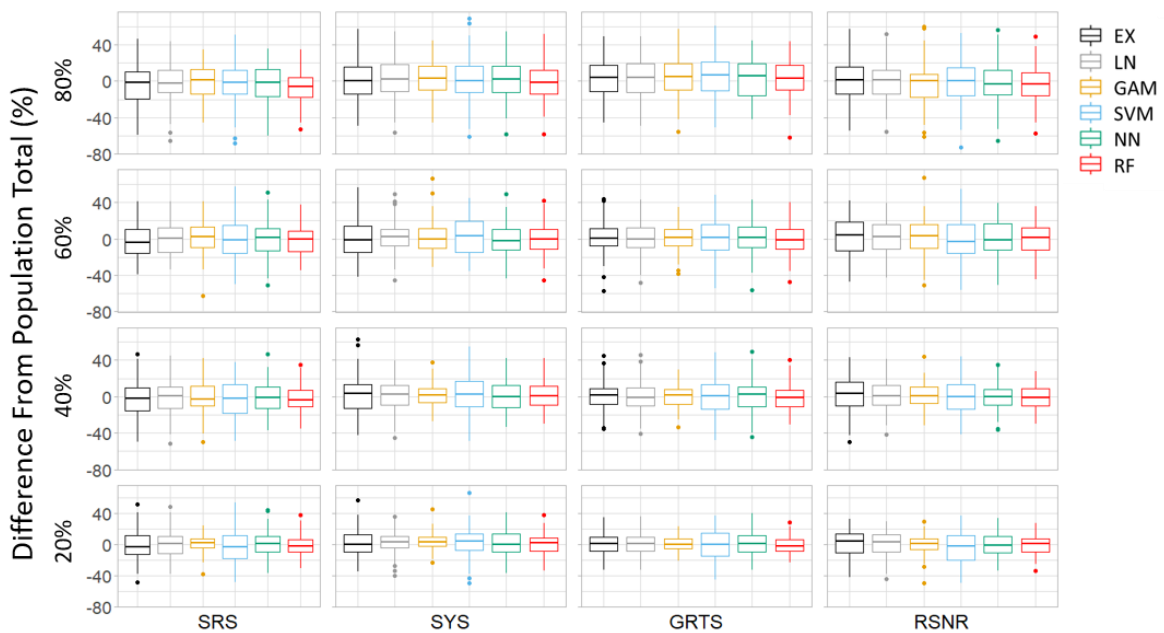
From the perspective of population totals, it is clear that the GRTS designs had the largest positive impact on expansion-based estimators (Figure 7, reduction in variability of % bias). While the impact of sample designs were less apparent for model-based estimators, generally the GRTS design had the least variation in population estimates (Figure 7) while producing the smallest RMSEs (Table 1). Across all iteration the mean difference between observed and estimated total, measured as a percentage of the response surface total, was close to zero (Figure 7), suggesting that estimator bias, if present, was minor.

The best performing estimator for all sample designs, transformations, and introduced errors was the GAM estimator followed by RF, SVM, NN, LN, and expansion estimators (smallest RMSE). Comparing results across sample designs for the GAM estimators, GRTS and SYS typically had the smallest RMSE and least variation in population estimates. While we anticipated that expansion and model-based estimates would be slightly biased for the RSNR sample design, our results did not fully support that claim. This is likely due to the relatively large proportion of area within 50 m of a road in the NAIP image subset, the identical trend and error distributions applied across the study area, and the similar distribution of spectral values found within the road buffers and the image.

**Table 1.** Design with smallest average root mean squared error (RMSE) across iterations by response surface distribution (SC1), fully-specified modeling technique, and amount of introduced noise (% Noise).

SC1	% Noise	GAM	RF	SVM	NN	LN	EX
Linear	20	SRS	GRTS***	GRTS***	GRTS*	SRS	GRTS***
	40	GRTS	GRTS	GRTS***	SRS	GRTS	GRTS**
	60	GRTS	GRTS*	GRTS***	GRTS*	RSNR	GRTS***
	80	SYS	SYS	GRTS**	GRTS	SYS	GRTS*
Squared	20	GRTS*	GRTS***	GRTS***	GRTS	GRTS***	GRTS***
	40	SYS	GRTS***	GRTS**	GRTS	GRTS*	GRTS***
	60	GRTS	GRTS*	GRTS*	SYS	GRTS	GRTS***
	80	SRS	GRTS	GRTS*	GRTS	GRTS	GRTS*
Nonlinear	20	GRTS*	GRTS*	SYS	RSNR*	GRTS	GRTS*
	40	GRTS**	GRTS	GRTS	GRTS	GRTS	GRTS
	60	GRTS	GRTS	RSNR	GRTS	GRTS	GRTS
	80	GRTS	GRTS*	RSNR	GRTS	GRTS	GRTS

\* statistically different than SRS at  $\alpha = 0.05$ ; \*\* statistically different than SRS at  $\alpha = 0.01$ ; \*\*\* statistically different than SRS at  $\alpha = 0.001$ ; GAM = general additive model; RF = random forests; SVM = support vector machine; NN = neural network; LN = linear model; EX = expansion estimator; SRS = simple random sample; SYS = systematic random sample; GRTS generalized random tessellation stratified; RSNR = random sample near roads.



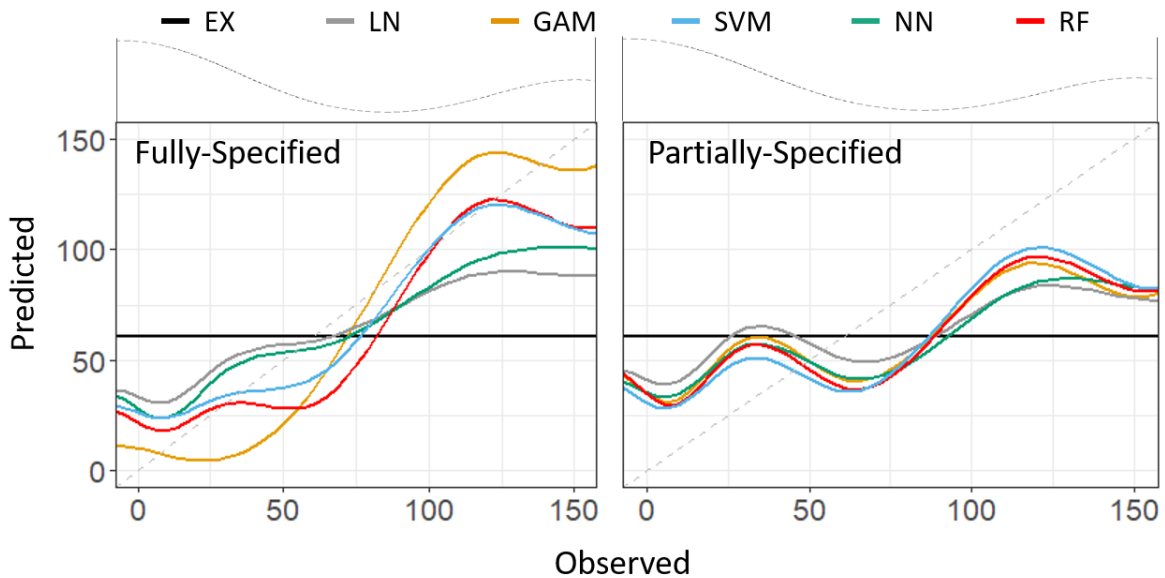
**Figure 7.** Boxplots of proportion bias matrix for differences from population totals. Difference values are expressed as a percentage of the population total for expansion and fully-specified model-based estimators derived from 100 iterations of nonlinear trend response surfaces with normally distributed errors based on 20%, 40%, 60%, and 80% of the total nonlinear SC1 image standard deviation. Column and row titles identify sample designs and proportion of SC1 standard deviation used in response surfaces. Ex = expansion; LN = linear; GAM = general additive model; SVM = support vector machine; NN = neural networks; RF = random forests; SRS = simple random sample; SYS = systematic random sample; GRTS generalized random tessellation stratified; and RSNR = random sample near roads.

As anticipated, when evaluating designs, percent error, and model estimation techniques for various ranges of response variable values using nonlinear response surface cell values, we saw similar trends as described in section 4.2 and as displayed in Figure 5. Figure 8 illustrates these



relationships by presenting predicted versus observed smoothed trends for the GRTS sample designs, expansion and model-based estimators, and 40% introduced noise into the relationship between response and predictors. Of particular note in Figure 8 is that all methods produced attenuated estimates and that the GAM, RF, and SVM procedures more closely match the one-to-one line depicted in Figure 8 within the most common areas in feature space (frequency graphs above graphics in Figure 8). These general trends also extended to partially-specified estimators in which only NAIP bands 1-3 were used as predictor variables.

Moreover, all models regardless of misspecification (model distribution or lack of predictors) tended to produced better estimates than expansion-based estimation alone. Similar to results in section 4.2, the GRTS sample design coupled with the GAM-based estimator produced the best results and more closely followed the one-to-one line when weighted by the frequency of sampled values (Figure 8). Finally, in all comparisons, as percent error increased departures from the one-to-one line also increased (not shown).

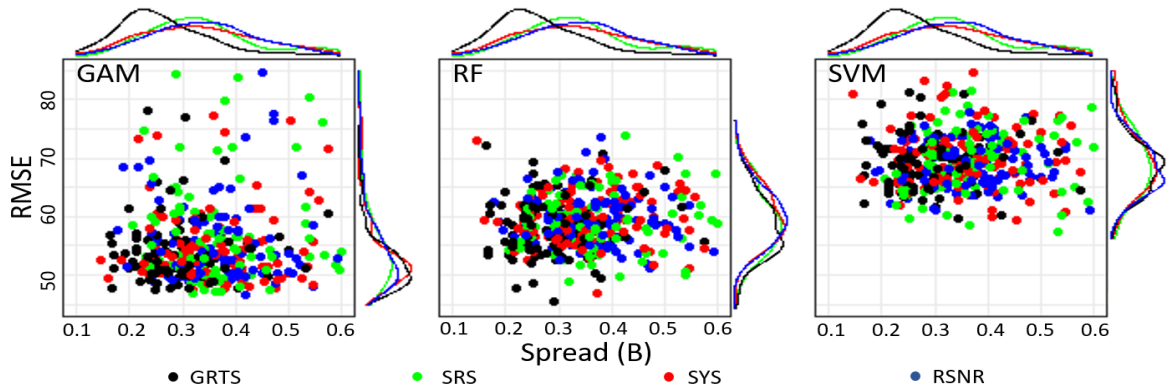


**Figure 8.** Smoothed trend of nonlinear response surface predicted values (y-axis) versus observed (x-axis) values for design and model based estimators calibrated using generalized random tessellation stratified designs (GRTS) and image bands 1-4 (Fully-Specified) and image bands 1-3 (Partially-Specified) as predictor variables. Introduced noise into the nonlinear response surface was held constant at 40% of the nonlinear SC1 surface standard deviation. Sample size was held constant at 50 for each of the 100 iterations used in the study. The gray dashed lines within Fully-Specified and Partially-Specified graphs serve as a one-to-one reference line while the gray dashed line above the graph depicts relative frequency of each observed value. Ex = expansion; LN = linear; GAM = general additive model; SVM = support vector machine; NN = neural networks; RF = random forests

#### 4.4 Impact of spreading sample units in feature space

While estimation approach had a larger effect on our results, sample design also impacted parameter estimates. Across all response surfaces, iterations, and modeling techniques the GRTS sample design had the smallest or was not statistically different than the smallest averaged RMSE (Table 1). Additionally, the variability in estimated population totals tended to be smaller for the GRTS sample design (Figure 7). Moreover, for the top three estimation techniques (GAM, RF, and SVM) and the most complex response surfaces (nonlinear), spreading samples in feature space generally produced the smallest RMSE (Figure 9, Table 1). Interestingly, estimates derived from samples spread in geographic space (SYS) often performed equally to samples spread in feature space (GRTS) even though response surfaces did not depend directly on geography (eq. 1-3). In large part

this can be attributed to spatial correlation in NAIP bands (Figure 2, Moran's I statistic). Nevertheless, across all estimation techniques, estimators derived from samples spread in feature space tended to have the smallest RMSE (Figure 9). Moreover, the sample design that typically had the smallest B statistic (well spread samples) was GRTS.



**Figure 9.** Root mean squared error (RMSE) versus spread values ( $B$ , eq. 6) for the top three estimation techniques given nonlinear response surface sample design, 40% introduced noise into the relationship between response and predictors, a sample size of 50, and 100 iterations. Density of RMSE and  $B$  values are presented to the right and above each graph. GAM = general additive model; RF = random forests; SVM = support vector machine; SRS = simple random sample; SYS = systematic random sample; GRTS generalized random tessellation stratified; and RSNR = random sample near roads.

## 5. Discussion

While sample design affected the performance of all estimators, the estimation approach had a larger impact on cell and population estimates. As expected, in all instances evaluated, regardless of the underlying relationship between response and predictor variables, model-based estimators produced better pixel-level estimates than expansion estimators (Figure 6, Table 1). Additionally, when sample units were spread and balanced in feature space model-based estimators of population totals were typically less variable than expansion estimators and had relatively low bias. This finding suggests that models derived from samples balanced and spread in feature space produce low-bias estimates with less variability than RSNR, SRS, and SYS designs.

Likewise, expansion estimators derived from samples spread and balanced across feature space (GRTS), produced similar estimates as model-based estimates of population totals with less variability than SYS, SRS, and RSNR expansion estimators; supporting the idea that spreading and balancing sample observations in feature space can substantially improve population estimates [3, 17, 39]. Within our simulations, the variability in estimated totals was on average always less for model-based estimators and expansion estimators derived from samples spread and balanced in feature space. In all instances the increased precision (reduction in RMSE) of using a model-based estimator over expansion-based estimators alone outweighed the impacts of the bias introduced by model-based estimators, even when models were misspecified.

Within our simulations, GAM and RF were the two best modeling techniques when all NAIP bands were used to calibrate models. Once calibrated, these estimators were able to better identify and track the underlying transformations of NAIP cell values better than other evaluated estimators. Additionally, when models were misspecified (e.g., a linear model applied to a nonlinear response), raster cell values approached expansion-based estimates. Furthermore, when only the first three NAIP bands were used to calibrate models, model-based cell estimates also approached expansion-based estimates. These findings suggest that using probability sampling, ancillary data, and models calibrated from probability samples improves pixel-level value estimates, even if the model is misspecified and especially if sample units are spread across the values of the ancillary data (feature space). This finding is especially relevant today with the availability of remotely sensed imagery and

the correlations between land cover (e.g., forest ecosystems) and spectral and textural image metrics (e.g., [24, 25]). While it is tempting to view our results as confirmation of the robustness of GAM and RF modeling techniques, it is important to recognize that within our simulation, randomness played a critical role in all sample designs [17] and that error and SC1 transformations were consistent across the spatial domain. Other relationships and error structures may favor different modeling techniques [8]. However, on average when samples were spread and balanced in feature space, pixel-level and population totals estimates derived from expansion and model-based procedures were as good as or better than samples that were not spread or balanced in feature space. Relatively speaking, this result is most likely because spreading and balancing sample units across feature space will more consistently find changes in complex relationship between response and predictors than samples that are not spread and balanced.

While we did not directly evaluate sample size in our comparisons, we assume that the strength of the relationship between response and predictor variables, represented in our study as added random noise, would have impacts on estimation similar to sample size. Specifically, we would expect that an increase in sample size would have a similar impact to estimated values as a decrease in noise introduced into the response surfaces. This would suggest that as sample size increases, estimates should become more precise and measures of spread, such as the B statistic [19, 39] will become smaller. Future investigations should look at quantifying tradeoffs in sample size, spread, and the strength of the relationship between response and predictor variables on estimation for modeling techniques such as LN, GAM, RF, SVM, and NN.

These simulated findings have practical relevance for natural resource managers who are interested in inventory, monitoring, and managing natural resources. Specifically, improvements in estimating characteristics of a natural resource for a given population (e.g. basal area in a forest) can be gained by incorporating models into the estimation process [1, 24, 25]. Moreover, indirect estimates of population subdomains (e.g., stands) can be made from model-based estimates [23]. In the case where observational units (e.g., pixels) cover a smaller spatial domain than the subdomain of interest (e.g., stands), those pixel estimates can be aggregated to the spatial extent of the stand. Conversely, when pixels have an extent larger than the stand of interest, model estimates can be attributed to the entire stand. In instances where pixel estimates partially cover the extent of the stand, estimates can be weighted and attributed to the stand based on the amount of overlapping area between pixels and the stand. However, models have estimation error, which should be incorporated into the standard errors of the estimates of the domain and subdomain characteristics [23, 41, 42].

In our simulation, we used iteration and sampling to quantify estimation error. Our top performing estimator (GAM) on average had the least amount of bias and the smallest RMSE across iterations. However, there were instances (iterations) within our simulations when the GAM model had, relatively speaking, large RMSE. These instances within our iterations identify the case in which a sample drawn from the population were not well balanced or spread. While the GRTS sample design minimized the occurrence of samples that were not well balanced or spread, it did not eliminate those types of samples. This means that while rare, some samples may have a disproportionately large number of extreme occurrences within feature space which could adversely affect the calibration of a given model and model estimates. In this situation bagging or boosting could be used to iteratively draw random subsets from a given sample, calibrate multiple models, average model estimates, and empirically estimate standard error to reduce the impact of extreme observations [43, 44]. Applying methods such as this should have a similar impact to the variation in RMSE values as what is displayed in Figure 9 for the RF modeling techniques, which uses bagging and model averaging [28].

For forest managers, this suggests that estimates of forest characteristics such as species composition, basal area, and tree counts can be improved for a given area by relating field measurements to remotely sensed imagery such as NAIP (e.g., [1, 24]). Moreover, with the relative abundance of free remotely sensed data and newer software designed to facilitate these types of

analyses (e.g., [45]), managers can estimate characteristics of the forests they manage with a greater level of detail and accuracy than was previously possible.

## 6. Conclusion

In this study we compared multiple estimators for a variety of response and predictor variable relationships, error distributions, amounts of noise, and sample designs. Our findings indicated that balance and spread are important aspects of a sample, estimates of pixel-level and population totals can be improved by incorporating models, and spreading samples within feature space can improve estimates. Likewise, for those same samples, when the relationship between response and predictor variables is misspecified, missing key information, or is extremely noisy, expansion and model-based estimates of the population converge, suggesting that with regards to estimation, nothing is lost by using ancillary data. Moreover, for mapping endeavors attempting to spatially depict various characteristics of a landscape, samples used to calibrate predictive models should aim to spread and balance observational units across feature space.

## References

1. McRoberts, R. Probability- and model-based approaches to inference for proportion forest using satellite imagery as ancillary data, *Remote Sensing of Environment*, **2010**, 114, 1017-1025.
2. McRoberts, R. Satellite image-based maps: Scientific inference or pretty pictures?, *Remote Sensing of Environment*, **2011**, 115, 715-724.
3. Grafström, A.; Shao, X.; Nylander, M.; Petersson, H. A new sampling strategy for forest inventories applied to the temporary clusters of the Swedish national forest inventory, *Can. J. For. Res.* **2017**, 47, 1161-1167.
4. Gregoire, T.; Valentine, H. *Sampling Strategies for Natural Resources and the Environment*, Chapman & Hall, Boca Raton London, New York, **2008**, 474 p.
5. Gregoire, T. Design-based and model-based inference in survey sampling: appreciating the difference, *Can. J. For. Res.*, **1998**, 28: 1429-1447.
6. Stehman S. & Czaplewski R. 1998. Design and analysis for thematic map accuracy assessment: fundamental principles. *Remote Sensing of Environment*, **1998**, 64, 331-344.
7. Crowson, M, Hagenseker, R.; Waske, Bjorn. Mapping Land cover change in northern Brazil with limited training data, *Int J Appl Earth Obs GeoInf.* **2019**. 78, 202-214.
8. Foody, G.; Mathur, A.; The use of small training sets containing mixed pixels for accurate hard image classification: Training on mixed spectral responses for classification by a SVM, *Remote Sensing of Environment*, **2006**, 103, 179 -1 89
9. Xie, Y.;Lark, T.; Brown, J.F.; Gibbs, H. Mapping irrigated cropland extent across the conterminous United States at 30 m resolution using a semi-automatic training approach on Google Earth Engine, *ISPRS Journal of Photogrammetry and Remote Sensing*, **2019**, 155, 136- 149,
10. Houborg, R.; McCabe M. A hybrid training approach for leaf area index estimation via Cubist and random forests. *machine-learning*, **2018**, 135, 173 – 188.
11. Kavzoglu T. Increasing the accuracy of neural network classifications using refined training data. *Environmental Modelling & Software*, **2009**, 24, 850-858.
12. Lesparre J. & Gorte B. Using mixed pixels for the training of a maximum likelihood classification. Proceeding in *International Society for Photogrammetry and Remote Sensing*, 2006, 36(7), 6.
13. Stehman S. Basic probability sampling designs for thematic map accuracy assessment. *Remote Sens.* **1999**, 20(212), 2423-2441.
14. Comber A., Fisher P., Brunson C., & Khmag A. Spatial analysis of remote sensing image classification accuracy. *Remote Sensing of Environment*, **2012**, 127, 237-246, doi: 10.1016/j.rse.2012.09.005.
15. Stehman S. Model-assisted estimation as a unifying framework for estimating the area of land cover and land-cover change from remote sensing. *Remote Sensing of Environment*, **2009**, 113, 2455-2462, doi:10.1016/j.rse.2009.07.006.
16. Stehman S. Sampling designs for accuracy assessment of land cover. *International Journal of Remote Sensing*, **2009**, 3(20), 5243-5272, doi:10.1080/01431160903131000.

17. Tille, Y.; Wilhelm, M. Probability Sampling Designs: Principles for Choice of Design and Balancing, *Statistical Science*, **2017**, 32(2), 176-189.
18. Stevens, D.L.; Olsen, A.R. Spatially-balanced sampling of natural resources, *Journal of American Statistical Association*, **2004**, 99, 262-277.
19. Grafström, A.; Lundström, N.L.P. Why well spread probability samples are balanced. *Open Journal of Statistics*. **2013**, 3, 36-41.
20. Edwards T., Cutler D., Zimmermann n., Geiser L., & Moisen G. Effects of sample survey design on the accuracy of classification tree models in species distribution models. *Ecological Modeling*, **2006**, 199, 132-141, doi:10.1016/j.ecolmodel.2006.05.016.
21. Brus, D.J. Sampling for digital soil mapping: A tutorial supported by R scripts, *Geoderma*, **2019**, 338, 464-480.
22. McRoberts R.E. & Walter B. 2012. Statistical inference for remote sensing-based estimates of net deforestation. *Remote Sensing of Environment*, **2012**, 124: 394-401, doi: 10.1016/j.rse.2012.05.011.
23. Rao, J.N.K.; Molina, I. *Small Area Estimation Second Edition*. Wiley and Sons, Hoboken, New Jersey, **2015**, 441.
24. Hogland, J.; Anderson, N. St. Peters, J; Drake, J; Medley, P. Mapping Forest Characteristics at Fine Resolution across Large Landscapes of the Southeastern United States Using NAIP Imagery and FIA Field Plot Data. *ISPRS International Journal of Geo-Information*, **2018**, 7(4): 140; doi:10.3390/ijgi7040140.
25. Hogland, J.; Anderson, N.; Affleck, D.L.R.; St. Peter, J. Using Forest Inventory Data with Landsat 8 imagery to Map Longleaf Pine Forest Characteristics in Georgia, USA. *Remote Sens.* **2019**, 11, 1803, doi: 10.3390/rs11151803
26. Neter, J.; Kutner, M.; Nachtsheim C.; Wasserman, W. *Applied linear statistical models fourth edition*; McGraw Hill, Boston, Massachusetts, **1996**, pp 1408.
27. Bishop, C.M. *Pattern Recognition and Machine Learning*. Springer Science and Business Media, LLC, Singapore KYO. **2006**, pp 738.
28. Wood, S.N.; Augustin, N.H. GAMs with integrated model selection using penalized regression splines and applications to environmental modeling. *Ecological Modeling*. **2002**, 157, 157-177.
29. Breiman, L. Random forests. *Mach. Learn.* **2001**, 45, 5–32.
30. National Agriculture Imagery Program [NAIP]. National Agriculture Imagery Program (NAIP) Information Sheet, **2012**, Available online: [http://www.fsa.usda.gov/Internet/FSA\\_File/naip\\_info\\_sheet\\_2013.pdf](http://www.fsa.usda.gov/Internet/FSA_File/naip_info_sheet_2013.pdf) (Accessed 5/14/2014).
31. Moran, P. Notes on Continuous Stochastic Phenomena. *Biometrika*. **1950**, 37, 17–23, doi:10.2307/2332142.
32. Kincaid, T. M. and Olsen, A. R. spsurvey: Spatial Survey Design and Analysis. R package version 4.0.0, **2019**.
33. TIGER 2017. TIGER/Line Shapefiles (machine readable data files) / prepared by the U.S. Census Bureau, online: <https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html> (last accessed 4/27/2019).
34. R Core Team. R: A language and environment for statistical computing. R Foundation for statistical Computing, Vienna, Austria, **2014**, online: <http://www.R-project.org/>, last accessed 4/28/2018.
35. Wood, S.N. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *Journal of the Royal Statistical Society (B)*, **2011**, 73(1), 3-36
36. Karatzoglou, A; Smola, A.; Hornik, K.; Zeileis, A. kernlab - An S4 Package for Kernel Methods, R. *Journal of Statistical Software*, **2004**, 11(9), 1-20.
37. Venables, W. N.; Ripley, B. D. *Modern Applied Statistics with S. Fourth Edition*. Springer, New York, **2002**, pp. 495.
38. Liaw, A.; Wiener, M. Classification and Regression by random forest. *R News*, 2002, 2(3), 18--22.
39. Stevens, D.L.; Olsen, A.R. Spatially Balanced Sampling of Natural Resources. *Journal of the American Statistical Association*. **2004**, 22(3), 262-278.
40. Johnson, R.; Wichern, D. *Applied Multivariate Statistical Analysis – Fifth Edition*. Prentice hall, Upper Saddle River, New Jersey, **2002**, 767
41. Opsomer, J.D.; Breidt, F.J.; Moisen, G.G.; Kauermann, G. Model-Assisted Estimation of Forest Resources With Generalized Additive Models, *Journal of the American Statistical Association*, **2007**, 102:478 400-416.

42. Breidt, F.J.; Opsomer, J.E. 2017. Model-Assisted Survey Estimation with Modern Prediction Techniques, *Statistical Sciences*, **2017**, 32(2) 190-205.
43. Opitz, D.; Maclin, R. Popular ensemble methods: an empirical study. *JAIR*. **1999**, 11, 169-198
44. Breiman, L. Stacked regressions. *Machine Learning*. **1999**, 24(1), 49-64.
45. Hogland, J.; Anderson, N. Function Modeling Improves the Efficiency of Spatial Modeling Using Big Data from Remote Sensing, *Big data and cognitive computing*, **2017**, 1(1). 1-14.

## Chapter 4

# Estimating forest characteristics for longleaf pine restoration using normalized remotely sensed imagery in Florida USA

**Abstract:** Effective forest management is predicated on accurate information pertaining to the characteristics and condition of forests. Unfortunately, the cost of acquiring detailed, accurate information that adequately described the complex spatial and contextual nature of forests, across broad landscapes, can be cost prohibitive to collect. While significant advancements in using remotely sensed data to derive fine scale information about our forests have been made, those advancements often fall short of providing the spatial detail desired for making well-informed management decisions. In large part, this disparity between what information is desired to make well-informed decisions and what information has been extracted from remotely sensed data stems from interacting challenges in big data and data science, and technical gaps between what is easily identified within remotely sensed imagery and what characteristics of the forest are used when planning and managing forests. In this case study we addressed big data challenges and bridged technical gaps related to describing forest characteristics by incorporating field plot layouts specifically designed to be used with remotely sensed data when calibrating predictive models, describing a new image normalization procedure that brings images of varying spatial resolutions to a common radiometric scale, and implementing an ensemble generalized additive modeling technique to accurately estimate the spatial distribution of key forest characteristics for longleaf pine (*Pinus palustris*) restoration efforts in the panhandle of Florida, USA. This work overcomes several of the major barriers associated with linking remotely sensed imagery with plot data over large areas.

**Keywords:** restoration, longleaf, relative normalization, ensemble generalized additive models, forests, Big data, data science

---

## 1. Introduction

Longleaf ecosystems are some of the most endangered forest ecosystems in the world [1]. Historically, these ecosystems covered approximately 37 million ha in the southeast United States of America (USA) and provided habitat for a wide range of plants and animals due to their unique structure and adaptation to fire. The species composition and structure of these ecosystems is maintained by frequent fire, which preserves a diverse early successional understory and relatively open longleaf pine (*Pinus palustris*) dominated overstory [2]. Due to many anthropogenic factors such as land use change, the suppression of fire on the landscape, over harvesting of timber, and the replacement of longleaf pine with faster growing loblolly (*Pinus taeda*) and slash (*Pinus elliottii*) pines, less than 1.5 million ha of longleaf pine remain [2]. Because of this dramatic loss of habitat there has been a recent resurgence in longleaf ecosystem restoration and conservation that calls for more than doubling the existing area covered by longleaf ecosystems by 2024 [3].

Transforming such a large amount of the existing landscape back to a healthy longleaf ecosystem condition within today's social framework will require well planned forest and natural resource management and education targeted towards public, private, and nonprofit agencies, organizations, and landowners that are receptive to achieving restoration goals. Moreover, the identification of restoration opportunities will require detailed, spatially explicit information about not only existing longleaf ecosystems but also the condition of other forested and nonforested landscapes. While detailed information is necessary to compare and evaluate restoration projects and make informed decisions, such information currently does not exist and is cost prohibitive to acquire across broad extents using ground-based inventory methodologies (e.g. [4]). Because of this limitation and the lack



of quality information, optimal restoration opportunities can be difficult to identify and fully justify, leaving managers to rely on opinion and intuition when making important and potentially costly restoration decisions.

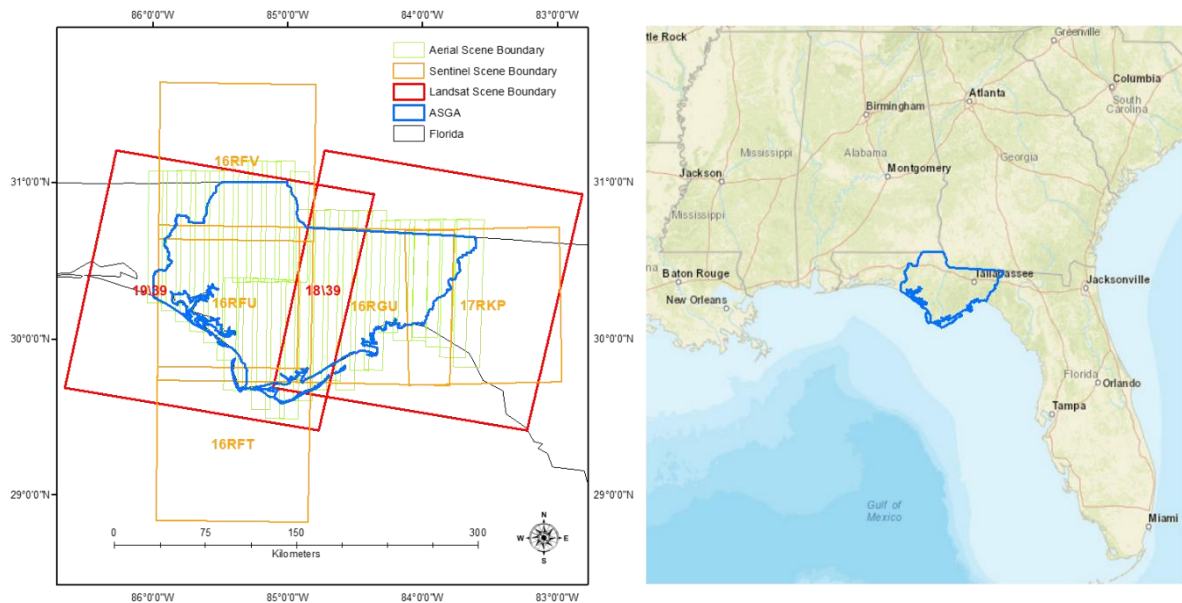
Alternatively, relating remotely sensed data to field measurements of existing forest characteristics to generate fine to medium grained spatially explicit information at a much lower cost than traditional inventory methodologies has shown great promise [5, 6]. However, in practice the implementation of these techniques and the use of modeled outputs has seen mixed success. In large part this discrepancy between promise and success stems from challenges and problems related to handling big data and the data science techniques used to transform these data into actionable information. Additionally, the lack of analysts and managers within natural resources with a requisite background in data science [7] and a legacy of the types of outputs historically created from remotely sensed imagery (e.g. [8, 9]) contribute to this inconsistency. While the scarcity of trained individuals in the field of data science is now being addressed at many universities [10], the broader issues related to handling big data, the techniques used to transform remotely sensed data streams into information to guide resource management, and the types of products created for informing restoration are active, ever evolving areas of research [11].

Within this framework, we present a case study that: 1) used field plots and multiple sources of remotely sensed data to convert spectral data into products to inform silviculture and forest management, 2) developed and applied new techniques for normalizing multi-spectral imagery obtained at various spatial resolutions and extents, 3) evaluated the impact of image normalization on mapping key forest characteristics, and 4) implemented an ensemble modeling approach to estimate key forest metrics and measures of estimation error. These techniques and evaluations are applied across the Apalachicola Significant Geographic Area (ASGA) in the panhandle of northwestern Florida, USA, and are used to derive estimates of tree density and basal area by species groups for longleaf pine restoration purposes.

## 2. Materials and Methods

### 2.1 Study Area

The ASGA is a large geographic area in the panhandle of Florida, USA, that consists of approximately 2 million ha of public and private land holdings (Figure 1). With regard to longleaf restoration, ASGA provides a unique combination of large intact remnant longleaf ecosystems clustered and intermixed with converted forest types, agricultural lands, and urban landscapes [12]. Longleaf ecosystems present within the boundary of the ASGA include sandhill, flatwoods, and upland pine communities that represent a national biodiversity “hot spot”, which provides habitat for various rare and threatened species such as the red-cockaded woodpecker (*Leuconotopicus borealis*), Bachman’s sparrow (*Peucaea aestivalis*), frosted flatwoods salamander (*Ambystoma cingulatum*), gopher tortoise (*Gopherus polyphemus*) and indigo snake (*Drymarchon spp.*) [12]. Equally distinctive within ASGA are the partnerships and coalitions dedicated to the restoration of longleaf pine, making this region an ideal location to quantify various aspects of the existing forest condition for the purposes of management and restoration.



**Figure 1.** Apalachicola Significant Geographic Area (ASGA) location map (right) and image acquisition dates and scene boundaries (left) used to bring images to a common relative radiometric scale. Landsat 8 and Sentinel II images were acquired for three seasons, representing a leaf on growing season (LGS), leaf on dormant season (LDS), and a leaf off winter season (LWS). Aerial imagery was only acquired for LDS. Image acquisition dates can be found in Table A1.

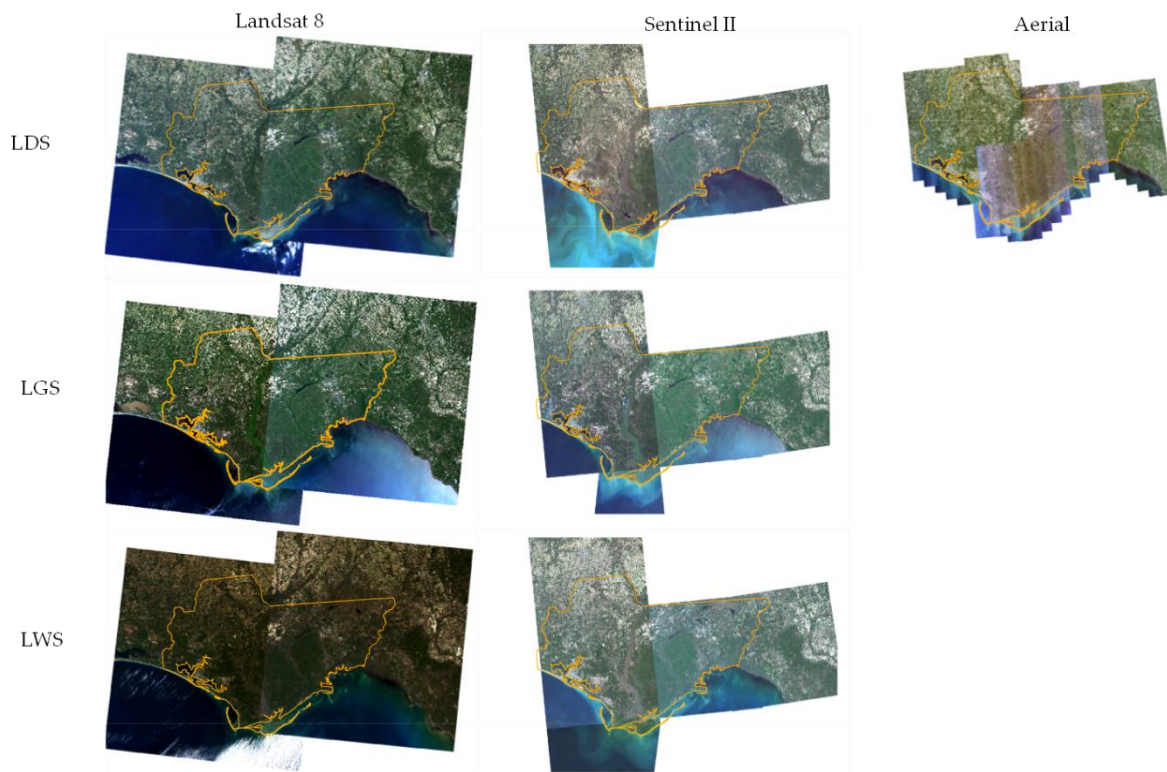
## 2.2 Data

Primary datasets used in this study consist of field inventory (plots), aerial based digital imagery [13], multi-temporal Sentinel II satellite imagery [14], and multi-temporal Landsat 8 satellite imagery with level 1 terrain precision (L1TP) correction [15, 16] for the extent of ASGA. Field plot measurements were collected by trained technicians [17] between the months of September 2017 and February 2018. Aerial imagery was flown by Quantum Spatial [13] at a nominal spatial resolution of 0.6 m for blue, green, red, and near infrared portion of the electromagnetic spectrum between October 26<sup>th</sup> and November 18<sup>th</sup>, 2017 (Figure 2). Multi-temporal Sentinel II images were manually selected for cloud free tiles and were downloaded from the European Space Agency (ESA) website Sentinel Online [18] for a leaf on growing season (LGS), leaf on dormant season (LDS) and a leaf off winter season (LWS) between the years of 2017 and 2018 (Figure 2). Sentinel II bands 2, 3, 4, and 8 at a nominal spatial resolution of 10 m were used for study comparisons. Landsat 8 L1TP images were also manually selected for cloud-free scenes and were downloaded from United States Geological Survey (USGS) website Earth Explorer [19] for similar LGS, LDS, and LWS periods between the years 2015 and 2018 (Figure 2). Landsat 8 bands 2 - 7 were used for study comparisons (Table 1).

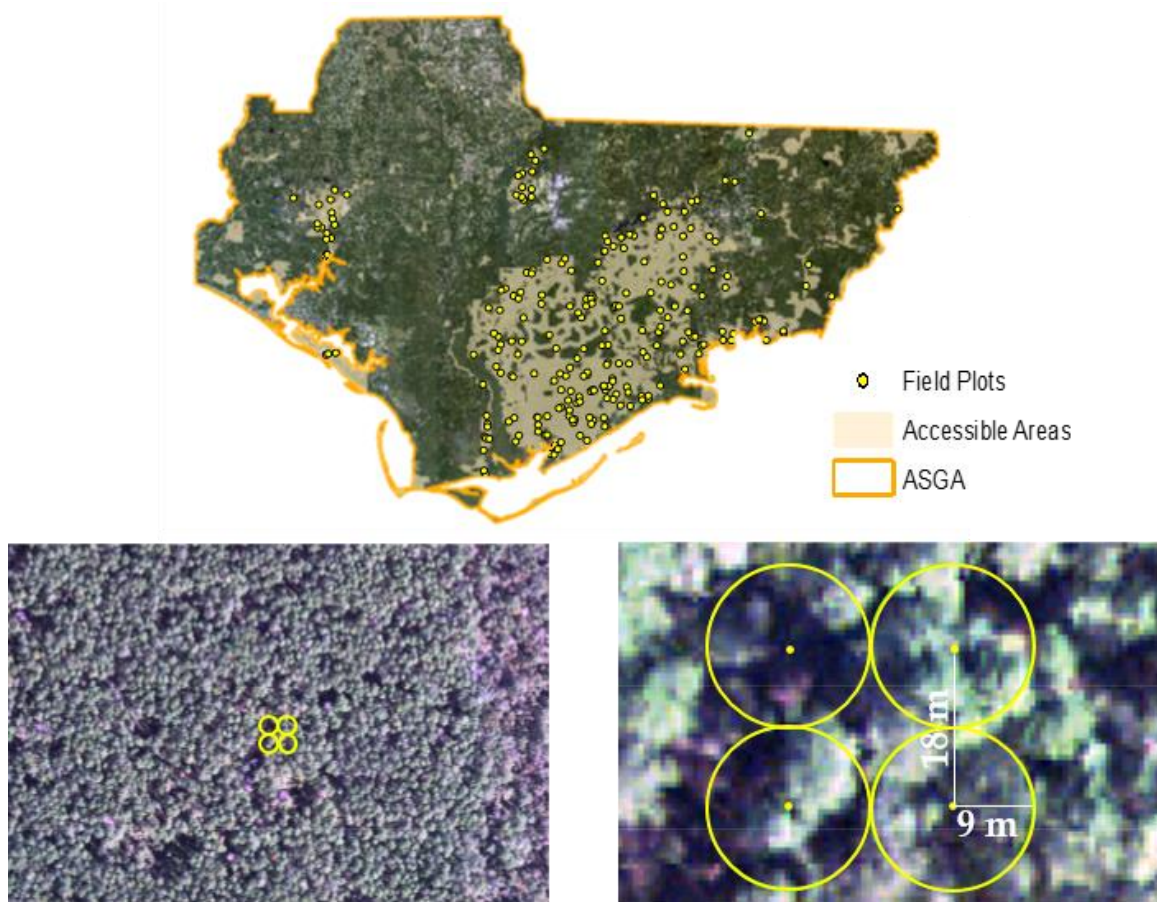
To reduce the negative impacts of co-registration [20] we used a field plot layout consisting of four adjacent circular subplots, each with a radius of 9 m. In total, 244 field plots were installed and used to measure tree species counts and diameters at breast height (dbh; 1.37 m) (Figure 3). For the extent of each field plot (36 m by 36 m, containing 4 subplots each with 9.0 m radius), spectral and textural metrics derived from source and normalized Landsat 8, Sentinel II, and aerial imagery (section 2.4) were extracted and used to calibrate models that estimate the dominant forest type (DFT), longleaf pine presence (LPP), and basal area per ha (BAH) and trees per ha (TPH) of pine and other tree species groups (section 2.5).

**Table 1.** Sources, spatial resolution, and labeling of imagery used in comparisons.

Source	Code	Resolution	Bands	Texture Metric	Season	ID
Landsat 8	L	30 m	2-7	Mean (cell)	LDS	1-6
					LGS	7-12
					LWS	13-18
				Standard deviation (3 by 3 cells)	LDS	19-24
					LGS	25-30
					LWS	31-36
Sentinel II	S	10 m	2, 3, 4, and 8	Mean (3 by 3 cells)	LDS	1-4
					LGS	5-8
					LWS	9-12
				Standard deviation (5 by 5 cells)	LDS	13-16
					LGS	17-20
					LWS	21-24
Quantum Spatial	A	0.6 m	1-4	Mean (61 by 61 cells)	LDS	1-4
				Standard deviation (61 by 61 cells)	LDS	5-8



**Figure 2.** Landsat 8, Sentinel II, and Quantum Spatial (Aerial) imagery true color displays for leaf on fall dormant (LDS), leaf on spring growing (LGS), and leaf off winter (LWS) season mosaics used in the study. The AGSA is outlined in orange.



**Figure 3.** Spatial allocation (Top), contextual scale (bottom left), and layout (bottom right) of field plots (yellow points) overlaid on aerial imagery used in the study. Target population (tan colored region) accounted for approximately 23% of the total area within the ASGA.

### 2.3 Sample design

A significant portion of ASGA is made up of private land holdings. Owing to access constraints, forests in these holdings could not be sampled. The remaining area, consisting primarily of public lands, constituted our target population (Figure 3). Plot locations within the target population were allocated spatially at random within 804 m of a road. In total a sample of 244 locations was drawn from the target population. This restriction on sampling necessitates the following assumptions: 1) the target population encompasses the spectral domain of the full ASGA and 2) the relationships between variables observed in the field and those derived from the imagery are the same across the target population and the full ASGA including all lands. Only the first of these can be empirically evaluated.

To evaluate the first assumption, we partitioned the multivariate distribution of the normalized spectral and textural metrics (section 2.4) over the ASGA into 100 classes using an unsupervised k-means classification [21, 22]. A simple random sample of 10,000 locations drawn from across the ASGA was used for this purpose. We then compared the class proportions for the full ASGA to the proportions across the 244 sample locations.

### 2.4 Field data

Plot data consisted of global position system (GPS) locations and tree measurements within the boundaries of subplots (Figure 3) at those locations. Basal area (BAH;  $\text{m}^2 \text{ha}^{-1}$ ) and tree density (TPH;  $\text{trees ha}^{-1}$ ) were summarized for trees greater than 5 cm in dbh by species group. Additionally, each plot was assigned a dominant forest type (DFT) label of Nonforest, Pine, or Other using the rules

defined in Table 2. Finally, longleaf pine presence or absence (trees above 5 cm dbh) was described at the plot level (Table 2).

**Table 2.** Definitions and queries used to label Pine, Other, and Nonforest dominant forest cover types and the presence of longleaf pine along with proportion of sampled observations meeting those criteria.

Classification	Label	Definition/Query	Proportion
DFT	Pine (2)	$(P\_BAH \geq O\_BAH)$ and not Nonforest	0.480
	Other (1)	$(O\_BAH > P\_BAH)$ and not Nonforest	0.340
	Nonforest (0)	$P\_BAH + O\_BAH < 2 \text{ m}^2 \text{ ha}^{-1}$	0.180
LPP	Present	$LP\_BAH > 0 \text{ m}^2 \text{ ha}^{-1}$ and not Nonforest	0.168

\*P\_BAH = pine, O\_BAH = other, and LP\_BAH = longleaf pine basal area per ha measure in  $\text{m}^2$ .

Field plots were located using a Wide Area Augmentation System (WAAS) enabled global position system (GPS) embedded within a Trimble Recon© data recorder. Plot GPS and tree data were collected using Environmental Systems Research Institute (ESRI)'s ArcPad© version 10.0 and a suite of mobile data collection applets developed specifically for this project (ArcPad Libraries; Field Plot Protocol). While every attempt was made to navigate to the exact field plot location, real time navigation can introduce geographic error. To minimize this error when relating field plot summaries to remotely sensed data, 20 GPS positions were collected and averaged to estimate the exact center of the southeastern subplot within each 36 m by 36 m field plot. From the southeastern plot location remaining subplots centers were located based on ground distance and compass bearings. On average the standard deviation of GPS positions for each plot, across all plots was less than 1 m with a maximum standard deviation in northing of 2.8 m and easting of 5.1 meters occurring at one plot location (average horizontal dilution of precision of 0.93).

### 2.5 Image normalization

Remotely sensed data were re-scaled to a common radiometric scale using an enhanced aggregate no-change regression (EANR) methodology. While similar to aggregate no-change regression (ANR) [6, 23], this procedure seeks to leverage strong linear relationships among Landsat, Sentinel II, and aerial image bands to bring finer spatial resolution imagery to the same relative radiometric scale as coarser imagery. The main differences between EANR and ANR are: 1) an added aggregation step to bring finer resolution imagery to the same spatial scale as the reference imagery, 2) a normalization of raw digital number (DN) values, 3) a trimming procedure to mitigate confounding effects of land use/cover changes for images acquired at different dates, and 4) a sampling scheme to extract spectral aggregates within overlapping image boundaries.

Like ANR, EANR extends the concepts of automatic scattergram-controlled regression [24] by applying an area slicing algorithm to identify no-change pixels and then spatially aggregating pixel values around selected locations to minimize the effect of co-registration errors [20, 23]. Aggregated values at selected locations of the subject image are then regressed against the corresponding values for a reference image using ordinary least squares regression (OLS) on a band-by-band basis. Regression coefficients are then applied to the subject image to bring it to the same radiometric scale as the reference image and to other fine-scaled images that overlap the same reference image.

Critical to this process is the elimination of areas and pixels within the region of image overlap that have been affected by land use or land cover change. This was accomplished in two steps. The first step required visually estimating the proportion ( $p_\Delta$ ) of area within the overlapping regions of the images that were impacted by changes due to phenomena such as clouds, land use change, or land cover change. For our study we used an estimate of  $p_\Delta = 20$  percent to mask Landsat 8 and Sentinel II pixels. For aerial imagery we used a  $p_\Delta$  ranging between 20 and 30 percent. Specifically, DN values were separately normalized ( $\widehat{DN}$ ) to the unit scale for each of the subject (S) and

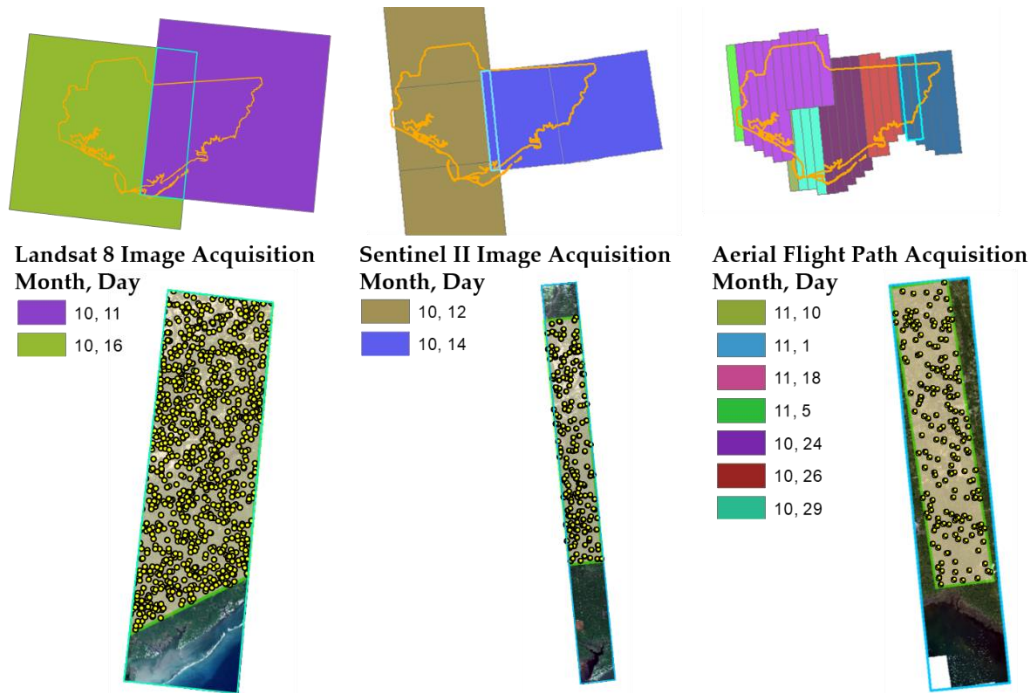
reference (R) images and bands (k). Differences in the normalized  $\widehat{DN}$  values between subject and reference images were then obtained for each band:

$$\Delta\widehat{DN}_k = \widehat{DN}_{R_k} - \widehat{DN}_{S_k} \quad (1)$$

After ordering the  $\Delta\widehat{DN}_k$  values from smallest to largest, the pixels in the lower 10% (i.e.,  $0.5p\Delta$ ) and upper 10% were identified as “change” pixels in the spatial aggregation process.

The spatial aggregation process minimizes the effect of co-registration errors [20] by extracting and calculating the mean values of unchanged pixels within the overlap of each image at a coarser grain size. In this procedure an aggregation block window size of nine by nine pixels was chosen based on simulated results found for Landsat 8 co-registration errors reported in Hogland and Affleck [20]. Aggregated blocks with more than 30 percent of the pixels identified as no-change were used for regression analyses of subject image blocks on reference image blocks for individual bands. The resulting regression coefficients for each band were then applied back to the non-aggregated pixel values of the subject image.

For Landsat 8 and Sentinel II images we used EANR with 1,000 randomly chosen locations within regions of overlap to bring images within a given season to a common relative radiometric scale. For normalization of Landsat 8 imagery, path/row 18/39 data were used as the reference images for each season. Reference images for normalization of Sentinel II imagery are specified in Table A1. For aerial imagery, normalized Landsat 8 imagery was used as reference. In instances when substantial land use or land cover change had occurred due to differences in image acquisition dates (e.g., changes in agricultural fields), manually defined spatial masks were used to remove additional pixels prior to implementation of the EANR procedure (Figure 4).



**Figure 4.** Example of spatial masks (semitransparent beige colored polygon with green boarder) and random locations (yellow points) used in the enhanced aggregate no-change regression procedure (EANR) for leaf on fall dormant Landsat 8, Sentinel II, and aerial imagery. Light blue polygons highlight the overlapping area between images acquired at different dates while masked areas highlight a subset of that area composed primarily of forest vegetation.

Once normalized, focal mean and standard deviation analyses [5, 6] were performed to quantify texture and mimic field plot extents for each band, season, and source of remotely sensed imagery (Table 1). For Landsat 8 imagery, mean band DN values were extracted based on the location of a given field plot and the nearest pixel (30m by 30m spatial resolution). Additionally at those locations,

Landsat 8 band standard deviations for a three by three moving window (90m by 90m) were calculated for each pixel and extracted. For Sentinel II imagery a three by three moving window (30m by 30m) was used to calculate mean values and a five by five window (50m by 50m) was used to calculate standard deviation. For aerial imagery a 61 by 61 window (grain size of 0.6 m, ~37m by 37m) was used to calculate image mean and standard deviation at each location. In total 68 different metrics were created and extracted from the normalized remotely sensed imagery using the averaged GPS location of each field plot and the nearest image pixel. As shown in Table 1, these metrics include 36 band values extracted from Landsat 8 based imagery (2 metrics for 6 bands and 3 seasons), 24 band values extracted from Sentinel II based imagery (2 metrics for 4 bands and 3 seasons), and 8 band values extracted from the aerial imagery (2 metrics for 4 bands and 1 season). To evaluate the impact of EANR on estimating DFT, LLP, BAH and TPH, this metric creation and extraction process was repeated for non-normalized imagery and the values were used to build and compare models.

### 2.6 Model Development, Comparisons, and Raster Surface Creation

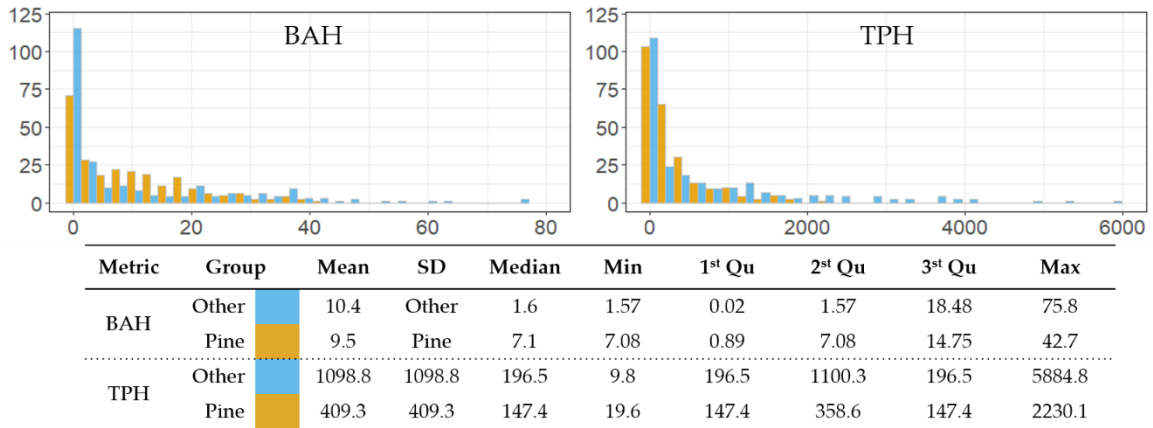
For each response variable (DFT and LLP labels, and pine and other tree species BAH and TPH) we used the variable selection routine described in [5, 6], to select EANR normalized and non-normalized remotely sensed metrics that significantly ( $\alpha = 0.05$ ) improved model fit of generalized additive models (GAMs), as defined by increased percent deviance explained. GAMs are a flexible modeling technique that can accommodate nonlinear relationships between response and predictor variables using penalized regression splines [25] and can be applied to non-Gaussian response data such as the DFT and LLP. While useful in identifying nonlinear, nonparametric relationships, GAMs can overfit sample data making estimates less generalizable to a given population [26]. To address the issue of overfitting, we employed a Monte Carlo re-sampling scheme to build a suite of 50 GAMs constructed from random subsets of our data. For each of the 50 GAMs, 75 percent of the observations within our sample were used to develop relationships between a given response and our previously selected predictor variables. The remaining 25 percent of the observations that were not used to build the GAM constituted an out of bag (OOB) subset of the data and were used to independently assess the accuracy of GAM estimates ( $\hat{P}$ ). Statistics calculated for assessment were root mean squared error (RMSE) for continuous response variables (BAH and TPH) and classification accuracy (calculated from a most likely class rule) for the binomial (LPP) and multinomial (DFT) categorical response variables. Once calibrated, we applied the ensemble of 50 GAMs (EGAM) to estimate each response variable ( $\hat{E}$ ) and corresponding standard error ( $\widehat{SE}$ ) at the pixel level as follows:

$$\hat{E} = \frac{1}{n} \sum_{i=1}^n \hat{P}_i \quad \text{and} \quad \widehat{SE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (\hat{P}_i - \hat{E})^2} . \quad (2)$$

EGAMs based on EANR normalized imagery were compared against EGAMs based on non-normalized imagery using trained and OOB, RMSE and classification accuracy statistics, and trained Akaike information criterion (AIC) [27, 28]. Our best fitting EGAMs (normalized versus raw image based predictors) for DFT, LLP and pine and other tree species BAH and TPH were used with the corresponding image metrics to produce raster surfaces depicting the condition of forests in the ASGA. Additionally, estimation errors from field samples were evaluated for spatial correlation using a global Moran's I test [29].

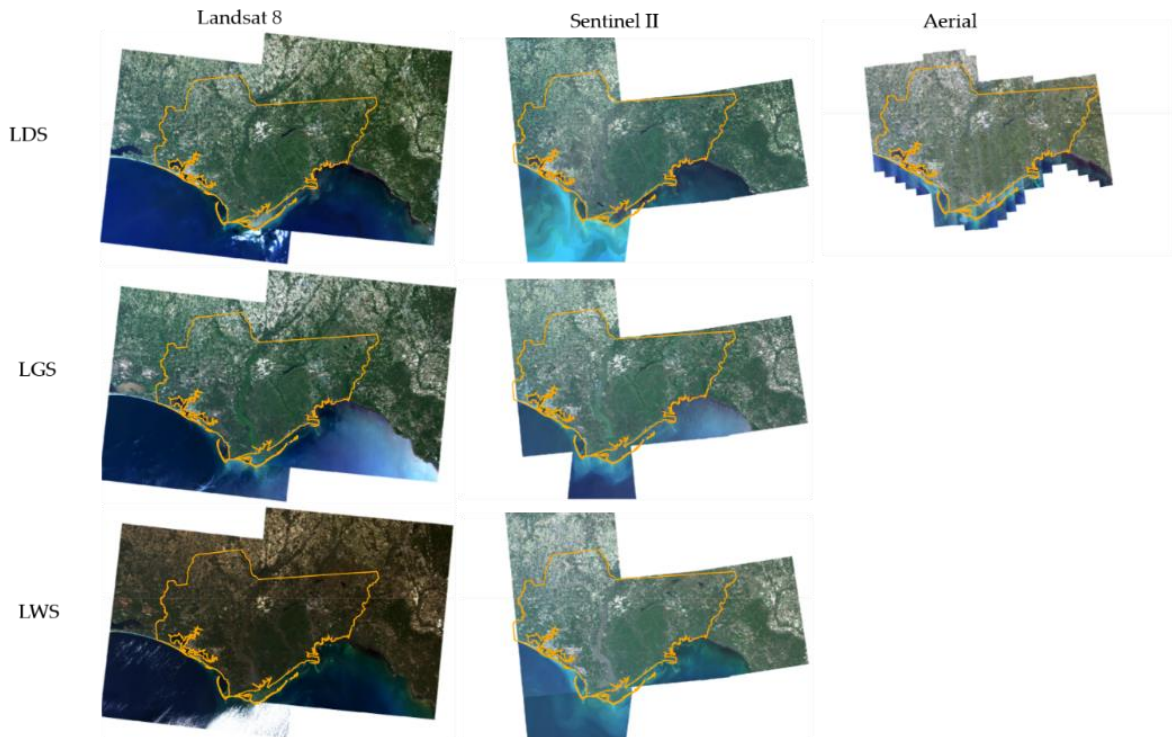
### 3. Results

#### 3.1 Field data and Image Normalization



**Figure 5.** Histograms and summary statistics of field plot basal area ha<sup>-1</sup> (BAH) and trees ha<sup>-1</sup> (BAH) for pine and other tree species groups.

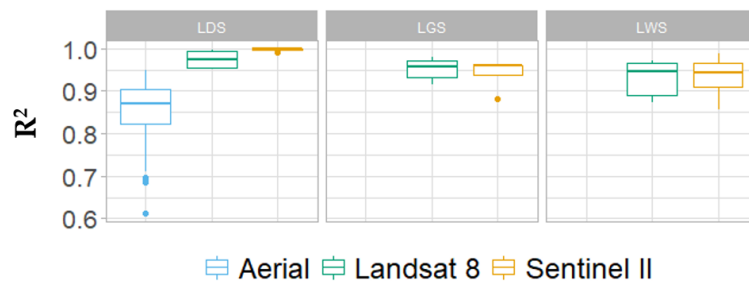
Plot data summaries are displayed in Figure 5 and Table 2. Almost half the plots were in the pine condition, but only 16.8% contained longleaf pine trees above 5 cm (and a total BAH above 2 m<sup>2</sup> ha<sup>-1</sup>). Mean BAH was only slightly higher in the DFT Other class than in the DFT Pine class, but BAH was more variable in the former. Mean TPH and variability in TPH was substantially lower in the Pine class relative to the Other forested condition. Taken together this indicates that tree diameter (dbh) tended to be larger in the Pine condition.



**Figure 6.** Landsat 8, Sentinel II, and Aerial true color displays for leaf on fall dormant (LDS), leaf on spring growing (LGS), and leaf off winter (LWS) season mosaics used in the study. Image mosaics depict imagery after performing the enhanced aggregation no-change regression procedure. The AGSA boundary is outlined in orange.



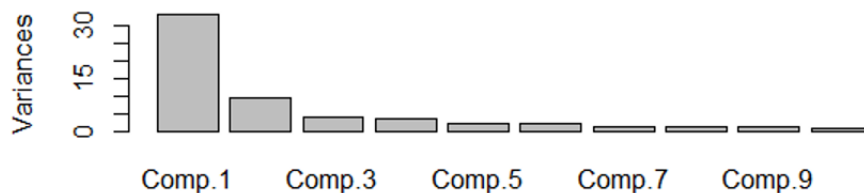
The EANR procedure brought Landsat 8, Sentinel II, and the aerial imagery to common radiometric scales (Figure 6). The  $R^2$  statistics for LDS, LGS, and LWS Landsat 8 normalization regressions were generally large (greater than 0.9; Figure 7) across all bands, with the lowest  $R^2$  coinciding with LWS analyses. Similarly, for Sentinel II imagery, the  $R^2$  statistics of the image-to-image overlap regressions were large (greater than 0.9) with the lowest occurring in LWS. The strength of these associations resulted in the improved radiometric consistency across Landsat 8 and Sentinel II images that is visually evident in Figure 6. In contrast, regressions of overlapping aerial and Landsat 8 Imagery were weak for several flight paths (Figure 7). The acquisition dates for these paths tended to be later (Figure 4) and potentially coincided with dramatic changes in plant phenology. As a result aerial scene boundaries were still apparent in normalized mosaics (Figure 6).



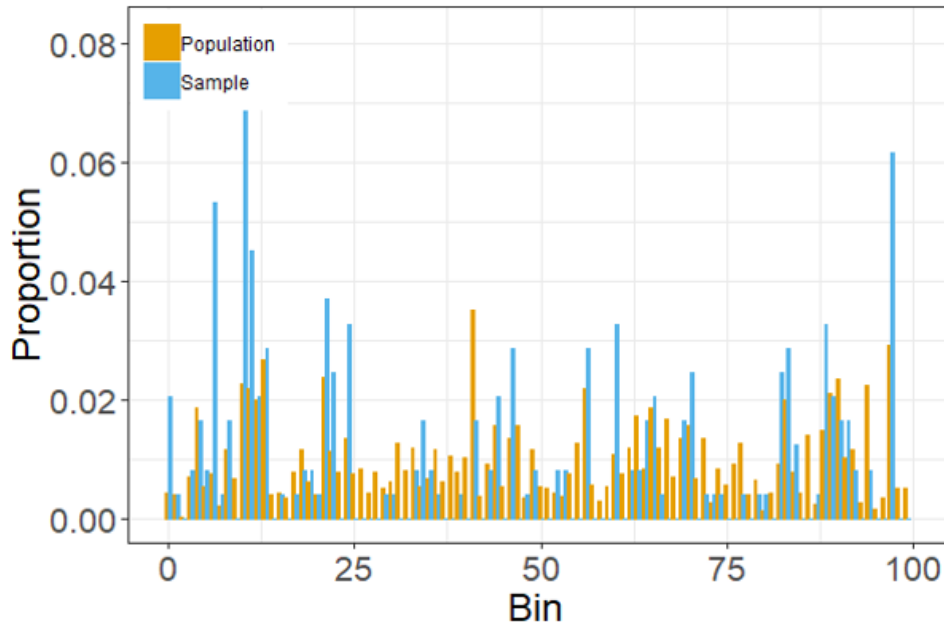
**Figure 7.** Box plots of enhanced aggregate no-change normalization (EANR) coefficient of determination ( $R^2$ ) results for each source of imagery and season across all image bands. Leaf on dormant (LDS), leaf on growing (LGS), and leaf off winter (LWS) season box plots are displayed from left to right. Aerial imagery was only acquired during LDS and is absent from LGS and LWS.

### 3.2 Sample distribution

Strong correlations existed among the 68 spectral and textural metrics derived from the imagery, with the first 10 principal components accounting for 86.6 percent of the variation within the 10,000 locations spread at random across the ASGA (Figure 8). Clusters derived from those metrics using the k-means unsupervised clustering algorithm divided the feature space into 100 classes of varying size (Figure 9). The distribution of the field plot locations across those same 100 classes shows that the field plots spanned a wide range of the feature space, but did not capture all the spectral classes. Many of the classes not represented at the field plot locations corresponded to urban and agricultural cover types.



**Figure 8.** Proportion of variance explained within the first 10 principal components.



**Figure 9.** Proportion of counts falling within each of 100 classes (Bins) derived from a k-means unsupervised classification of predictor variable value component scores from normalized Landsat 8, Sentinel II, and aerial imagery and a selection of 10,000 random locations representing the population (orange) compared to the counts of those same classes falling within a random sample of 244 accessible locations (blue).

### 3.3 Model Development, Comparisons, and Raster Surface Creation

Spectral and textural metrics selected for each EGAM varied by response variable and whether images were normalized to a common radiometric scale (Table 3). Using the variable selection procedure described by Hogland et al. [6], EGAMs selected between four and twelve metrics out of the potential 68 at a significance threshold of  $\alpha = 0.05$ . In all instances metrics were selected from each image source and from multiple seasons (Tables 1 and 3). Compared with EGAMs built using non-normalized texture metrics, EANR based EGAMs generally improved model fit as measured by classification error, RMSE, and AIC (Table 3). EGAMs based on non-normalized imagery generally selected fewer metrics and tended to utilize metrics derived from Landsat 8 and Sentinel II data sources.

Overall classification accuracies for DFT and LPP EGAMs were greater than 88 percent with the most common classification errors being the mapping of the Other forested condition as pine, and longleaf presence being mapped as absence (Figure 10). Training and OOB errors for DFT, LPP, BAH and TPH EGAMs are reported in Table 3 and suggest strong relationships among response and normalized predictor variables, particularly in Pine conditions. Estimated BAH and TPH values were also strongly correlated with observed values, though in densely stocked conditions (TPH above 1000 stems  $\text{ha}^{-1}$ , BAH above 20-30  $\text{m}^2 \text{ha}^{-1}$ ) EGAMs tended to underestimate BAH and TPH (Figure 11). Estimation errors for some EGAMs were positively spatially correlated, suggesting spatial patterns or trends in model errors (Table 4). Though tempting to address the spatial variability in estimation error using kriging [30], our sample design does not lend itself to accounting for spatial trends in the residuals across the largely privately-owned tracts of the AGSA that were outside the target population.

**Table 3.** Selected predictor variables and error statistics for normalized (EANR) and non-normalized (RAW) based Ensemble Generalized Additive Models (EGAMs).  $\overline{\text{Deviance}}$  denotes the average overall deviance of the sample (null model). Error statistics for DFT and LPP are in terms of 1 - accuracy of the classification while BAH and TPH errors are measured in terms of root mean squared error (RMSE) on the square root scale. Train and out of bag (OOB) errors denote whether error statistics were calculated from observations used to train a given model or withheld from training, respectively. Mean Akaike information criterion ( $\overline{\text{AIC}}$ ) was calculated from EGAMs and used to compare models built from EANR and RAW based predictors.

Response	Normalization	Predictors*	$\overline{\text{Deviance}}$	Train	OOB	$\overline{\text{AIC}}$
DFT	EANR	L3, L16, L17, S2, S5, S7, S21, A1	376.085	0.068	0.235	86.217
	RAW	L8, L9, L16, L18, S11, S5	375.140	0.103	0.263	110.023
LPP	EANR	L2, L17, S3, S4, A2, A5	166.455	0.056	0.129	94.683
	RAW	L2, L13, S9, A6	166.094	0.095	0.148	113.617
$\sqrt{\text{Pine BAH}}$	EANR	L2, L8, L10, L11, L16, L17, L18, L22, A2, A8	581.427	0.909	1.168	548.664
	RAW	L2, L5, L6, L8, L10, L11, L17, L22, S9, S10, S11, S12, A4	582.620	0.900	1.267	547.441
$\sqrt{\text{Other BAH}}$	EANR	L3, L5, L11, S7, S9, S21, S23, A2, A5	989.519	1.108	1.365	600.177
	RAW	L3, L5, L11, L13, L16, S12, A5, A7	992.229	1.188	1.541	637.448
$\sqrt{\text{Pine TPH}}$	EANR	L3, L5, L8, L10, L11, L17, L28, S4, S11, A2, A5	21,686.265	5.755	9.197	1,243.537
	RAW	L1, L4, L5, L6, L13, L22, L31, S11, S20	21,497.754	6.709	10.866	1,277.418
$\sqrt{\text{Other TPH}}$	EANR	L9, L13, S2, S6, S7, S20, A1	66,299.997	11.787	14.977	1,471.036
	RAW	L3, L5, L11, L13, S12, S20, A5	66,161.379	12.492	14.682	1,486.281

\*Predictor variable naming convention is based on the concatenation of Table 1 code and Id columns

**Table 4.** Global Moran's I (GMI) statistics and p-values for ensemble generalized additive model residuals (EGAMS).

EGAM*	GMI	p-value
DFT (Nonforest)	0.364	<0.001
DFT (Other)	0.067	0.190
DFT (Pine)	0.093	0.153
LPP (Present)	0.223	0.002
LPP (Absent)	0.223	0.002
Pine BAH	0.184	0.011
Pine TPH	0.068	0.188
Other BAH	0.087	0.131
Other TPH	0.140	0.037

\* DFT = Dominant Forest Type, LPP = Longleaf Pine Presence, BAH = basal area per ha (m<sup>2</sup>), TPH = trees per ha

**Dominant Forest Type (DFT)**

		Mapped			
		Other	Nonforest	Pine	
Reference	Other	0.843	0.000	0.036	<b>0.340</b>
		<b>0.885</b>	<b>0.015</b>	<b>0.100</b>	
	Nonforest	0.940	0.036	0.145	<b>0.180</b>
		<b>0.031</b>	<b>0.881</b>	<b>0.087</b>	
	Pine	0.091	0.955	0.159	<b>0.480</b>
		0.043	0.000	0.821	
		<b>0.079</b>	<b>0.024</b>	<b>0.897</b>	
		0.111	0.051	0.940	

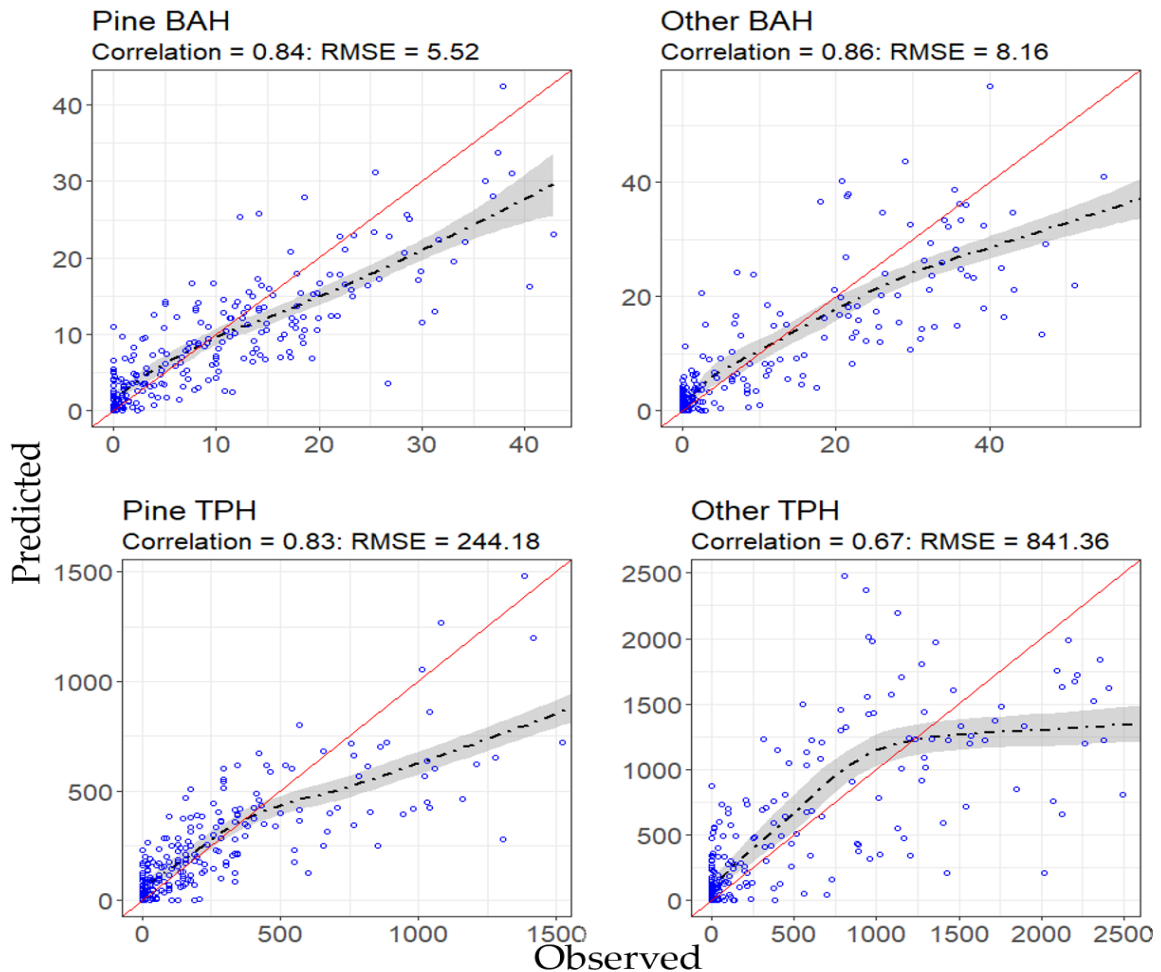
Overall Accuracy 0.890  
(0.816, 0.943)

**Longleaf Pine Presence (LPP)**

		Mapped		
		Absent	Present	
Reference	Absent	0.926	0.015	<b>0.832</b>
	<b>0.965</b>	<b>0.035</b>		
Present	0.985	0.059	<b>0.168</b>	
	0.122	0.537		
		<b>0.271</b>	<b>0.729</b>	
		0.415	0.829	

Overall Accuracy 0.923  
(0.861, 0.959)

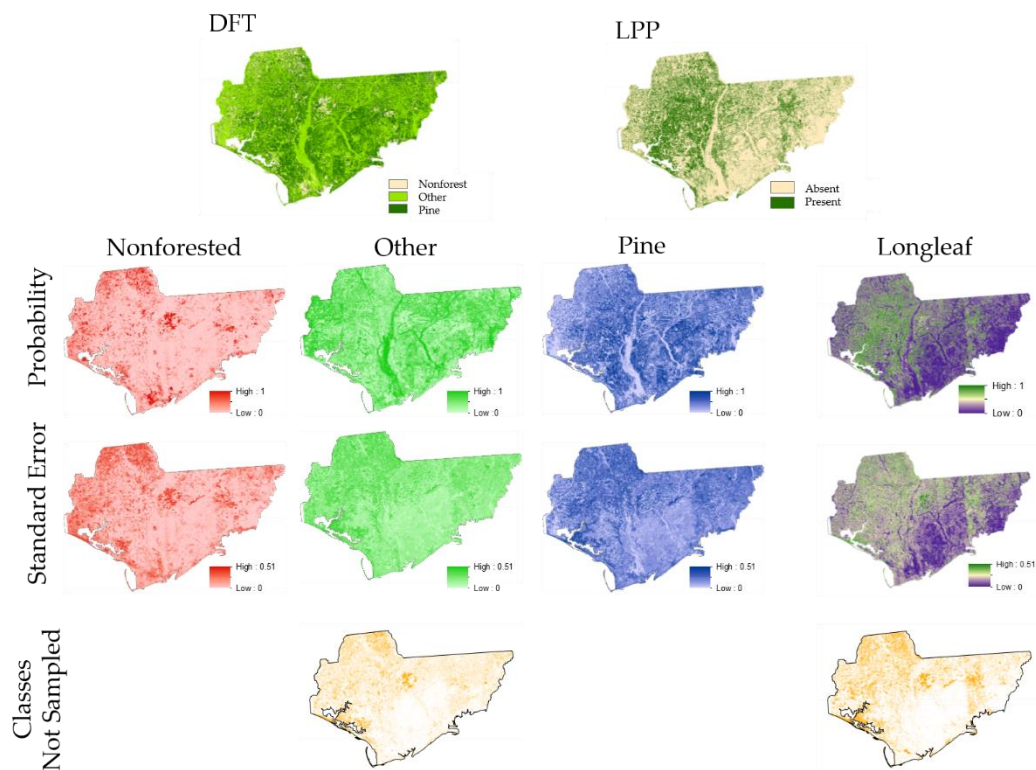
**Figure 10.** Accuracy assessment for dominant forest type (DFT) and presence of longleaf pine (LPP) using all 244 field plots and the corresponding ensemble generalized additive model (EGAM). Exact 95 percent lower and upper confidence limits for each class are reported above and below bold numbers within the error matrix and within parentheses of overall accuracy.



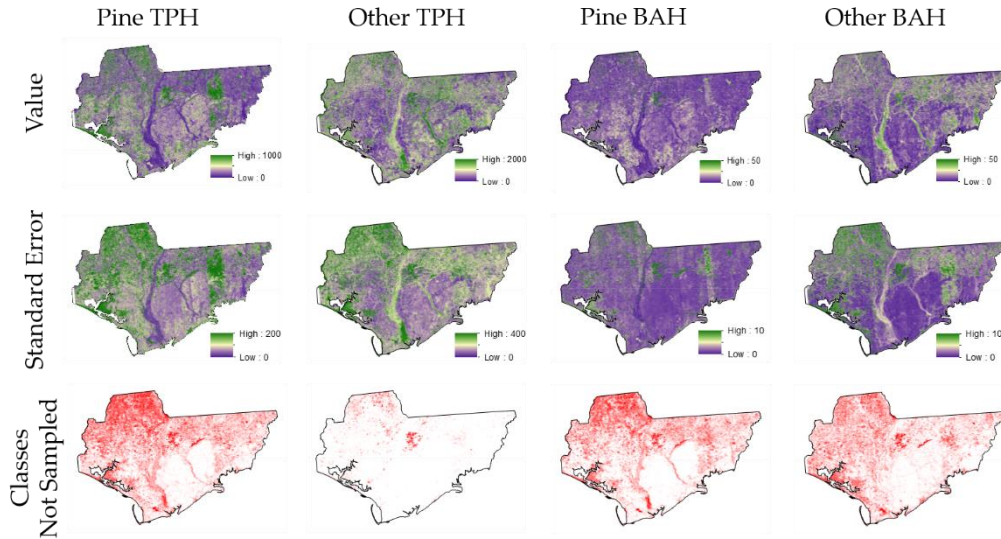
**Figure 11.** Predicted versus observed values (blue circles) of basal area per ha (BAH) and trees per ha (TPH) for pine and other tree species groups. The black dashed line shows the general trend based on a loess smooth estimator with grey shaded 95 percent confidence bands. The red line denotes a one to one line and provides reference for comparison between predicted and observed values.

EGAM raster surfaces were created at a spatial resolution of 30 m across the ASGA for all response variables. For DFT and LPP, EGAMs estimates include class probabilities and empirically derived standard errors for each cell within the study area. For pine and other  $\sqrt{BAH}$  and  $\sqrt{TPH}$ , estimates and standard errors were calculated using squared transformations of EGAM estimates.

DFT and LPP raster surfaces are presented in Figure 12 while BAH and TPH surfaces are presented in Figure 13. Additionally, Figures 12 and 13 display the spatial distribution of EGAM-specific feature space k-means classes that were not represented within the sample. Some banding of estimates and standard errors is evident in Figures 12 and 13 for the models using the A2 and A5 metrics (e.g., for LLP, Pine TPH, Other BAH) in the eastern portion of the ASGA. Those bands tend to appear in the corresponding sample representation maps, suggesting that certain flight paths in the aerial imagery could not be fully normalized and were not able to be adequately sampled in the field.



**Figure 12.** The spatial distribution of the dominant forest type most likely class (DFT), class probabilities, and class probability standard errors for Nonforested (red), Other (green), and Pine (blue) cover types (Table 2). Additionally, presence and absence of longleaf pine within a plot based on a most likely class rule (LPP), presence probabilities, and presence probability standard errors (purple to green color gradient) for LPP EGAM predictions. Finally, the spatial location of K-mean classes not represented in the sample used to train EGAM's are displayed as orange areas in the bottom row of graphics.



**Figure 13.** Display of BAH and TPH raster surfaces created from ensemble generalized additive models (EGAMs) for pine and other tree species groups. EGAM raster cell estimated values (top row of graphics) and standard errors (middle row of graphics) increase as colors transition from purple to green. The spatial location K-mean classes not represented in the sample used to train EGAM's are displayed as red areas in the bottom row of graphics.

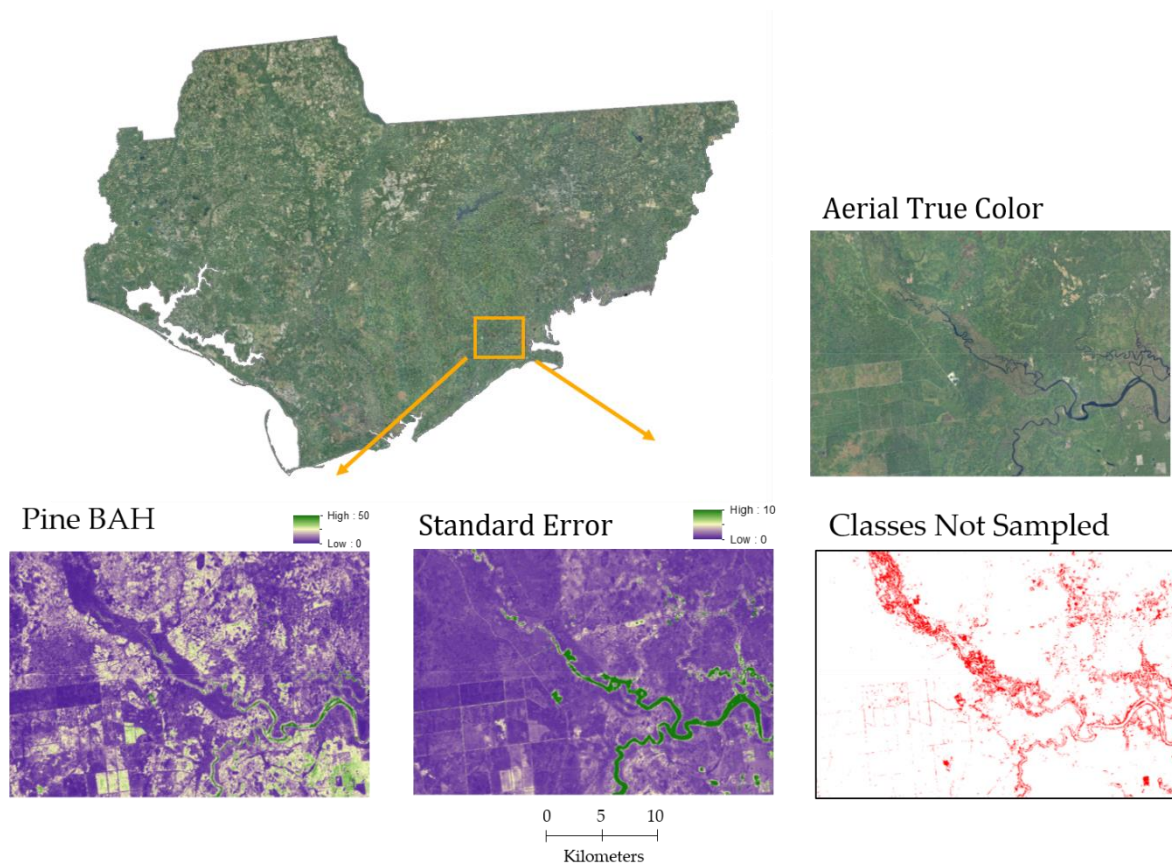
#### 4. Discussion

We have mapped key forest characteristics across ASGA that can be used to inform spatially explicit longleaf restoration decision making. These surfaces transform data that are relatively easy to collect into pertinent information for longleaf pine restoration, providing the fine level of spatial detail and accuracy needed to make well-informed restoration decisions. We have also described how to and demonstrated the importance of bring images to a common radiometric scale across sources, years, and spatial resolutions (Figure 6). Models built with EANR-normalized spectral and textural metrics utilized distinct sources and bands, and had less error than models built using the raw imagery (Table 3). Additionally, our study highlights how multiple image resolutions (temporal, spectral, and spatial) can improve model fit and estimation, especially after images have been brought to a common radiometric scale using a technique like EANR. Finally, we presented and implemented an ensemble generalized additive modeling (EGAM) approach to estimate DFT, LPP, and BAH and TPH for pine and other tree species groups that accommodates nonlinear relationships while mitigating the potential for overfitting. Combined, EANR and EGAM produce better estimates of key forest characteristics than previous longleaf pine mapping efforts [5, 6] (reductions in classification error and RMSE). Moreover, the EGAM procedure provided spatial depictions of empirically derived estimation error (standard error surfaces) that can be used by resource managers when making management decision.

Sample design in our study played an important role in inference. Because some areas were inaccessible, field sampling was limited to a subset of the total ASGA (Figure 3). This limitation means that inferences across ASGA must rely on the assumptions that the sample adequately characterizes the feature space and that the predictor-response relationships do not vary among different ownership types across the ASGA. Fortunately, our EGAMs could rely on relatively strong relationships among response and predictor variables and the sampled locations captured an appreciable extent of the feature space (Figure 8). For these reasons we feel relatively confident that model estimates provide an accurate depiction of forested conditions across the ASGA. However, our assessment did identify portions of feature space that were unsampled, and others that were overrepresented and underrepresented. From a model calibration perspective overrepresentation can be thought of as wasted effort when variability in the predictor-response relationship is relatively

minor. In those instances, observations from overrepresented regions of feature space could potentially be allocated to unsampled or underrepresented areas. Additionally, our assessment detected some geographic areas where the imagery could not be adequately normalized and thus may relate differently to the vegetation conditions. Future designs should attempt to spread sample locations across predictor variable space, while balancing the sample to mimic the distributional characteristics of the population [30]. Toward this end, the raster surfaces created from this study could be used to spread and balance sampling locations for future studies [31], thereby providing additional means to minimize error and reduce sample cost.

Quantitatively, DFT and LPP EGAM fit statistics suggest that our models accurately depicted forest cover types and the presence of longleaf pine. Similarly BAH and TPH EGAM estimates were strongly correlated with observed plot BAH and TPH over the lower ranges of these variables. Moreover, EANR based EGAMs outperformed EGAMs based on spectral and textural metrics from raw imagery that was not normalized using EANR. Qualitatively, our raster surfaces display more detail and are less impacted by variation in image acquisition dates when compared to previous mapping efforts [5]. However, there were instances when our EGAMs appeared to underestimate TPH (Figure 11) and to omit instances when longleaf pine was present (Figure 10). For example, when TPH was greater than 1,000 ha<sup>-1</sup> (e.g., as would be the case in young or overstocked stands) EGAMs underestimated TPH. Additionally, some areas dominated by cities, agricultural fields, water, and aquatic vegetation were underrepresented or not represented (Figures 12 and 13) in our sample and led to imprecise estimates (Figure 14). However, DFT, LPP, BAH and TPH models generally produced reliable, spatially explicit estimates.



**Figure 14.** A close up example of locations with large standard errors in estimated Pine BAH due to under or no representation of feature space classes within the sample used to train the Pine BAH ensemble generalized additive model.

While estimation error could be reduced for almost every EGAM by including additional EANR normalized spectral and texture predictive variables (i.e., setting an  $\alpha = 0.1$ ), the added complexity and processing associated with including more variables outweighed the marginal gains in accuracy and precision. Similarly, many of the predictor variables used in our EGAMs were highly correlated, suggesting marginal utility of additional metrics. Additionally, areas within feature space that were underrepresented or not represented generally produced extremely variable estimates, which should be viewed with skepticism. Moreover, EGAMs that were more complex (i.e., more predictor variables) tended to have larger proportions of the expanded feature space underrepresented. This suggests that for limited sample sizes, simpler models (i.e., fewer predictors) may be preferred for wall to wall mapping endeavors along with masking land cover types that are not forested. Finally, for some of our EGAMs, estimation errors exhibited minor positive spatial correlation. While tempting to use kriging methods [32] to remove global spatial trends in estimation error, our design did not adequately sample geographic space across ASGA, limiting the utility of kriging methods to contiguous accessible areas sampled within our study.

Though additional improvements can be made to our estimates of forest characteristics, tradeoffs between processing, model development, and sampling costs related to improvements in accuracy should be weighed to evaluate the level of precision needed to inform decision making [33]. Given the amount of model error potentially introduced by co-registration errors [20], our EGAMs explain the majority of the variation that can be accounted for within the field data and provide a substantial improvement over previous efforts [5, 6]. Additionally, by implementing an ensemble approach to generalized additive models, we were able to capture nonparametric trends in the data while mitigating overfitting and providing a technique to empirically estimate standard errors. The resultant estimates and standard errors provide the type of information needed to make both fine and coarse grained restoration decisions across the ASGA.

## 5. Conclusions

Adept forest management requires accurate information concerning the status and distribution of forest resources. To create accurate information pertinent to longleaf pine restoration, we developed procedures to bring multi-temporal images to a common radiometric scale, model nonparametric relationships between remotely sensed and field measured data, and produce spatial depictions of forest cover types, longleaf pine presence, and BAH and TPH by forest cover class. These procedures and sources of data were combined to produce the types of information needed to inform longleaf restoration planning and implementation at planning and tactical spatial scales across a relatively large area in northwestern Florida.

## Appendix A

**Table A1.** Landsat 8, Sentinel II, and Aerial image acquisition dates by season and path, row, or tile.

Source	Season	Path\Row\Tile	Acquisition Date
Landsat 8	LDS	18\39+	10/11/2016
Landsat 8	LDS	19\39	10/16/2015
Landsat 8	LGS	18\39+	5/7/2017
Landsat 8	LGS	19\39	4/9/2016
Landsat 8	LWS	18\39+	1/18/2018
Landsat 8	LWS	19\39	12/21/2016
Sentinel II	LDS	16RFV+	10/12/2018
Sentinel II	LDS	16RFU+	10/12/2018
Sentinel II	LDS	16RFT+	10/12/2018



Sentinel II	LDS	17RGU	10/14/2018
Sentinel II	LDS	17RKP	10/14/2018
Sentinel II	LGS	16RFV	3/16/2017
Sentinel II	LGS	16RFU	3/16/2017
Sentinel II	LGS	16RFT+	5/2/2017
Sentinel II	LGS	17RGU+	5/2/2017
Sentinel II	LGS	17RKP+	5/2/2017
Sentinel II	LWS	16RFV+	1/30/2018
Sentinel II	LWS	16RFU+	1/30/2018
Sentinel II	LWS	16RFT	2/24/2017
Sentinel II	LWS	17RGU	1/12/2017
Sentinel II	LWS	17RKP	1/12/2017
Aerial Imagery	LDS	7001	10/26/2017
Aerial Imagery	LDS	7002	10/26/2017
Aerial Imagery	LDS	7003	10/26/2017
Aerial Imagery	LDS	7004	10/26/2017
Aerial Imagery	LDS	8001	11/10/2017
Aerial Imagery	LDS	8002	11/18/2017
Aerial Imagery	LDS	8003	11/18/2017
Aerial Imagery	LDS	8004	11/18/2017
Aerial Imagery	LDS	8005	11/5/2017
Aerial Imagery	LDS	8006	11/5/2017
Aerial Imagery	LDS	8007	11/5/2017
Aerial Imagery	LDS	8008	11/5/2017
Aerial Imagery	LDS	8009	11/5/2017
Aerial Imagery	LDS	8010	11/1/2017
Aerial Imagery	LDS	8011	11/1/2017
Aerial Imagery	LDS	8012	11/1/2017
Aerial Imagery	LDS	8013	11/1/2017
Aerial Imagery	LDS	8014	10/26/2017
Aerial Imagery	LDS	9017	10/29/2017
Aerial Imagery	LDS	9018	10/24/2017
Aerial Imagery	LDS	9019	10/24/2017
Aerial Imagery	LDS	9020	10/24/2017
Aerial Imagery	LDS	9021	10/24/2017
Aerial Imagery	LDS	9022	10/24/2017
Aerial Imagery	LDS	9023	10/24/2017
Aerial Imagery	LDS	9024	10/24/2017
Aerial Imagery	LDS	9025	10/24/2017
Aerial Imagery	LDS	9026	10/24/2017
Aerial Imagery	LDS	9027	10/24/2017

---

+ Denotes reference image used to normalize a given image source. Aerial imagery was normalized to normalized Landsat 8 imagery.

## References

1. Noss, R.; LaRoe, E.; Scott, J. *Endangered Ecosystems of the United States: A Preliminary Assessment of Loss and Degradation*; Biological Report 28; National Biological Service: Washington, DC, USA, **1995**. Available online: <https://sciences.ucf.edu/biology/king/wp-content/uploads/sites/106/2011/08/Noss-et-al-1995.pdf> (accessed on 19 July 2019).
2. Oswalt, C.; Cooper, J.; Brockway, D.; Brooks, H.; Walker, J.; Connor, K.; Oswalt, S.; Conner, R. *History and Current Condition of Longleaf Pine in the Southern United States*; USDA Forest Service General Technical Report SRS-166; United States Forest Service: Ashville, NC, USA, **2012**. Available online: <http://www.srs.fs.usda.gov/pubs/42259> (accessed on 6 February 2019).
3. Regional Working Group for America's Longleaf. Range-Wide Conservation Plan for Longleaf. **2009**. Available online: [http://www.americaslongleaf.org/media/86/conservation\\_plan.pdf](http://www.americaslongleaf.org/media/86/conservation_plan.pdf) (accessed on 6 February 2019).
4. U.S. Forest Service Forest Inventory and Analysis Program: We Are the Nation's Forest Census. Available online: <https://www.fia.fs.fed.us/> (accessed on 6 February 2019).
5. Hogland, J.; Anderson, N.; St. Peter, J.; Drake, J.; Medley, P. Mapping forest characteristics at fine resolution across large landscapes of the southeastern United States using NAIP imagery and FIA field plot data. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 140. Available online: <https://www.mdpi.com/2220-9964/7/4/140/htm> (accessed on 6 February 2019).
6. Hogland, J.; Anderson, N.; Affleck, D.L.R.; St. Peter, J. Using Forest Inventory Data with Landsat 8 imagery to Map Longleaf Pine Forest Characteristics in Georgia, USA. *Remote Sens.* **2019**, *11*, 1803, doi: 10.3390/rs11151803
7. Gibert, K.; Horsburgh, J.S.; Athanasiadis, I.N.; Holmes, G. Environmental Data Science. *Environmental Modelling & Software.* **2018**, *106*, 4-12, doi: 10.1016/j.envsoft.2018.04.005.
8. Homer, C.; Dewitz, J.; Yang, L.; Jin, S.; Danielson, P.; Xian, G.; Coulston, J.; Herold, N.; Wickham, J.; Megown, K. Completion of the 2011 National Land Cover Database for the conterminous United States-Representing a decade of land cover change information. *Photogr. Eng. Remote Sens.* **2015**, *81*, 345–354.
9. LANDFIRE. Existing Vegetation Type Layer, LANDFIRE 1.1.0, U.S. Department of the Interior, Geological Survey, **2008**. Available online: <http://landfire.cr.usgs.gov/viewer/> (accessed on 28 October 2010).
10. Brunner, R.J.; Kim, E. Teaching Data Science. *Procedia Computer Science.* **2016**, *80*, 1947-1956, doi: 10.1016/j.procs.2016.05.513
11. Lokers, R.; Knapen R.; Janssen, S.; van Randen, Y.; Jansen, J. Analysis of Big Data technologies for use in agro-environmental science. *Environmental Modelling & Software.* **2016**, *84*, 494-504, doi:10.1016/j.envsoft.2016.07.017.
12. The Longleaf Alliance. About ARSA. Available online: <https://www.longleafalliance.org/arsa/about-arsa> (accessed on 23 of October, 2019).
13. Quantum Spatial. About Use. Available online: <https://www.quantumspatial.com/about-us> (accessed on 23 or October)
14. Earth Observing System [EOS]. Sentinel-2. Available online: <https://eos.com/sentinel-2/> (accessed on 23 of October 2019).
15. United States Geological Survey [USGS]. Landsat 8. Available online: [https://www.usgs.gov/land-resources/nli/landsat/landsat-8?qt-science\\_support\\_page\\_related\\_con=0#qt-science\\_support\\_page\\_related\\_con](https://www.usgs.gov/land-resources/nli/landsat/landsat-8?qt-science_support_page_related_con=0#qt-science_support_page_related_con) (accessed on 23 of October, 2019).
16. USGS. Landsat 8 Surface Reflectance Code (LASRC) Product Guide. General Technical Report LSDS-1368 Version 2.0. **2019**. Available online: [https://prd-wret.s3-us-west-2.amazonaws.com/assets/palladium/production/atoms/files/LSDS-1368\\_L8\\_Surface\\_Reflectance\\_Code\\_LASRC\\_Product\\_Guide-v2.0.pdf](https://prd-wret.s3-us-west-2.amazonaws.com/assets/palladium/production/atoms/files/LSDS-1368_L8_Surface_Reflectance_Code_LASRC_Product_Guide-v2.0.pdf) (accessed on 19 July 2019).
17. Florida Natural Areas Inventory [FNAI]. About Us. Available online: <https://www.fnai.org/about.cfm> (accessed on 23 of October, 2019).
18. ESA Sentinel Online. Copernicus Open Access Hub. Available online: <https://scihub.copernicus.eu/dhus/#/home> (accessed on 23 of October 2019).
19. USGS. EarthExplorer – Home. Available online: <https://earthexplorer.usgs.gov/> (accessed on 23 of October 2019).

20. Hogland, J.; Affleck, D.L.R. Mitigating the Impact of Field and image Registration Errors through Spatial Aggregation. *Remote Sens.* **2019**, *11*(3), 222, doi:10.3390/rs11030222.
21. Souza, C. Accord.Net Framework, online: <http://accord-framework.net/>, last accessed 9/27/2013.
22. Hogland, J.; Anderson, N. Function Modeling Improves the Efficiency of Spatial Modeling Using Big Data from Remote Sensing, *Big data and cognitive computing*, **2017**, *1*(1). 1-14.
23. Hogland, J. Creating spatial probability distributions for longleaf pine ecosystems across east Mississippi, Alabama, The Panhandle of Florida, and west Georgia, thesis. **2005**, Available online: [https://etd.auburn.edu/bitstream/handle/10415/603/HOGLAND\\_JOHN\\_19.pdf?sequence=1&isAllowed=y](https://etd.auburn.edu/bitstream/handle/10415/603/HOGLAND_JOHN_19.pdf?sequence=1&isAllowed=y), last accessed 12/20/2017.
24. Elvidge, C.D.; Yuan, D.; Weerackoon, R.D.; Lunnetta, R.S. Relative radiometric normalization of Landsat Multispectral Scanner (MSS) data using an automatic scattergram-controlled regression. *Photogramm. Eng. Remote Sens.* **1995**, *61*, 1255–1260.
25. Wood, S.N. Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models. *J. R. Stat. Soc.* **2011**, *73*, 3–36.
26. Wood, S.N.; Augustin, N.H. GAMs with integrated model selection using penalized regression splines and applications to environmental modeling. *Ecological Modeling*. **2002**, *157*, 157-177.
27. Akaike, H. Information theory and an extension of the maximum likelihood principle. In Proceedings of the 2nd International Symposium on Information Theory, Tsahkadsor, Armenia, 2–8 September 1971; Petrov, B.N., Csaki, F., Eds., Akadémiai Kiadó: Budapest, Hungary, **1973**; pp. 267–281.
28. Akaike, H. A new look at the statistical model identification. *IEEE Trans. Autom. Control* **1974**, *19*, 716–723.
29. Moran, P. 1950. Notes on Continuous Stochastic Phenomena. *Biometrika*. **1950**, *37* (1): 17–23. doi:10.2307/2332142. JSTOR 2332142.
30. Tille, Y.; Wilhelm, M. Probability Sampling Designs: Principles for Choice of Design and Balancing, *Statistical Science*, **2017**, *32*(2), 176-189.
31. Gregoire, T.; Valentine, H. *Sampling Strategies for Natural Resources and the Environment*, Chapman & Hall, Boca Raton London, New York, **2008**, 474 p.
32. Ruotsalainen, R.; Pukkala, T.; Kangas, A.; Vauhkonen, J.; Tuoiminen, S. The effects of sample plot selection strategy and the number of sample plots on inoptimality losses in forest management planning based on airborne laser scanning data. *C.an. J. For. Res.* **2019**, *49*, 1135-1146.

## Chapter 5

# Transforming data into information for natural resource decision making: Improving the utility of remote sensing products at tactical and planning scales

**Abstract:** Big data and the information and knowledge we glean from it are fundamentally changing the way in which resource management decisions are being made. The use of remotely sensed data, ever expanding computer technology, and various processing techniques are helping to provide natural resource managers with depictions of various aspects of ecosystems at unprecedented spatial and temporal resolutions. While the technologies used to gather data about natural resources have arguably outpaced our abilities to efficiently produce and utilize the many types of information that can now be generated, fundamentally there are still important questions related to scale, relevance, and understanding of the information within various data streams that must be addressed before empirically-driven decision making will reach its full potential within the natural resource community. In this communications, we identify some of the obstacles to adopting data driven decision-making within the natural resource community and highlight recent studies and solutions that advance our ability to transform data into useful information to facilitate informed natural resource decisions.

**Keywords:** Big data; decision science; data science; estimation; remote sensing; sub domain; natural resources; spatial modeling

---

## 1. Introduction

Our terrestrial environment is constantly being monitored. Today with satellite and airborne sensors such as MODIS [1], Landsat [2], Sentinel [3], and NAIP [4] we are able to acquire data about our environment at spatial, spectral, and temporal resolutions that were recently hard to imagine. Similarly, with advancements in drone technology and sensor hardware, the amount of remotely sensed data that can and is being acquired on a planned and ad hoc basis and used to quantify aspects of natural resources is staggering. In other fields such as finance and medicine, the recognition that large volumes of data are not being fully leveraged to inform the decision making process has led to an increased awareness of the fields of data science and the potential of what has become known as “big data” [5]. While still relatively new in concept and application, the tenets of data science and decision science with regards to big data streams appears to be lagging behind within the natural resources management community [6]. Though the products of many sensors can be manually interpreted by analysts to help visualize our environment, today the volume, velocity, and variety of data being collected requires an automated approach to interpretation [7]. Moreover, with advancements in image processing, many of the complex relationships that are difficult for a human analyst to visually identify can be made apparent through data mining, image processing, and statistical and machine learning techniques [8].

However, there are many obstacles that prevent the common use of data science principles within the natural resources community. Commonly recognized impediments include the lack of education and skills associated with integrating the various mathematical, statistical, machine learning techniques, computer programming languages, data formats, and the size of the data [6]. Less understood issues revolve around how data transformed into information (e.g., modeled outputs) can be leveraged to inform efficient decisions, the impact of using models calibrated for large spatial domains to produce estimates for subdomains, the propagation of errors, and the impact of model misspecification. Within this context, many of the big data and processing challenges that

plague other fields apply to natural resources. However, issues of scale, domain, error, and relevance can have additional meanings within a natural resource setting that are tightly coupled with inherent complexities associated with dynamic natural systems. Because of these complexities and lack of personnel trained in advanced data analytics, studies and methodologies that convert data into pertinent forest related information such as described in Chapter 4 are seldom used to their potential to identify and justify planning and management decisions. Moreover, it is often the case that resource managers ignore these sources of information when making decisions, instead valuing expert opinion over data-driven information. Worse yet, because resource managers are often unfamiliar with many of the modeling techniques and the strengths and weaknesses associated with various estimates, modeled outputs can be easily misused to erroneously justify or negate management action. In other words, it is often difficult for practitioners to identify misspecified or misused models and outputs.

In large part, the choice between expert opinion and data driven decision making is not binary. Instead, natural resource planning and management is better framed within the context of being justifiable, repeatable, and accurate [9], which requires expert interpretation and opinion regarding what can be measured, monitored, and documented. Within this context, appreciation for and understanding of what data science, remote sensing, and big data can provide for natural resource managers is paramount to meeting most resource based objectives. To highlight the possibilities provided by a data science approach to generating information for natural resource decision making, I synthesize the findings of this dissertation and discuss how data and decision science need to be integrated within the field of natural resources, and offer a path forward for such integration.

## **2. Synthesis**

An important aspect of converting data into useful information is fundamentally understanding how the two are distinct from but related to one another. Here I make a distinction between data and useful information in that data represent numbers or values while useful information applies meaning to those numbers or values that bridges the gap between information and knowledge. For example, a tree diameter can be measured and recorded as a datum. However, when that datum is supplemented with the context that it defines the girth at breast height of a given tree, then it also has inherent meaning and contains information to a forester who can use their knowledge about tree growth to infer the size, age, and value of that tree from that measurement. Similarly, spectral reflectance can be recorded by a satellite and stored as data. In this setting the data values may have little meaning to a forester but when rendered spatially, patterns emerge based on electromagnetic reflectance across locations, generating useful information that conveys meaning related to trees and forests. Inherently, patterns are connected to the spatial scale of one's vantage point. To convert patterns into useful information pertinent to a natural resource manager, measurements pertaining to trees and forests should be collected at similar spatial scales as the emergent patterns. While intuitive and straight forward, the implementation of a study design that directly relates what is found on the ground to the same location within an image is confounded by the difficulty of geographically registering the two sources of data, which leads to co-registration errors. In Chapter 2, I explored the impact of co-registration error on converting image-based data into useful information. My findings indicated that substantial variation can be attributed to co-registration error and that spatial aggregation can help to reduce this source of error. Additionally, I identified that this source of error is dependent on the degree of spatial correlation within images, which in turn can be used to estimate the reduction in potential model explanatory power due to co-registration errors. Moreover, I quantified the impact of subsampling within a defined extent on model error and used those results to define a field plot layout for Chapter 4 that minimized the impact of co-registration errors, while at the same time being practical to implement when using satellite and aerial imagery to model forest cover types, basal areas, and tree density and generate accurate estimates of these metrics.

Another vital facet to converting data into useful information when modeling is sample design. In Chapter 3 I explored the impact to estimation when samples are spread and balanced in predictor variable space for multiple sample designs, amounts of modeling error, functional relationships, and modeling techniques. My findings suggested that as error is introduced into the relationship between variables, model and expansion-based estimates converge for probability based sampling. This further indicates that including ancillary variables in the estimation process can only improve parameter estimation when using samples based on a probabilistic design. While sample design had less of an impact on model based estimates, substantial reductions in estimation error could also be attributed to spreading samples within feature space for both expansion and model-based estimates. Moreover, when functional relationships matched theoretical model based assumptions, those models tended to outperform, or perform equally to, data centric machine learning techniques. However, machine learning techniques across all functional relationships tended to perform better when the underlying relationships between variables were unknown. Finally, for the majority of comparisons within Chapter 3, the generalized additive modeling approach (GAMs) produced the least amount of estimation error, suggesting that this technique can be used to successfully “mine” complex patterns in multidimensional data.

Using the findings from Chapters 2 and 3, I conducted a case study in the panhandle of Florida (Chapter 4) that converted multi-temporal satellite imagery into useful information relevant to forest management and longleaf pine conservation. Information produced in this study included estimates of dominant forest cover types, the presence of longleaf pine, and pine and other tree species basal area ( $\text{m}^2 \text{ha}^{-1}$ , BAH) and tree density ( $\text{trees ha}^{-1}$ , TPH). Additionally, in this study I developed and presented a new procedure to bring images to a common radiometric scale (enhanced aggregate no-change regression, EARN) and implemented an ensemble GAM (EGAM) modeling technique. My estimates and associated standard errors of forest cover, longleaf presence, BAH, and TPH provide detailed spatial depictions of key forest information needed to inform planning and management. Moreover, in this chapter I highlight the negative impact of complex models and demonstrate that as the number of predictor variables increase, the proportion of feature space underrepresented or not represented within a sample increases, leading to reduced precision.

### **3. Discussion and Future Research**

The intensity with which we can acquire data about our environment is constantly increasing, with expanding volume, variety, and velocity. Transforming those data streams into useful information relevant to natural resources requires skillsets that expand what is currently taught within the fields of natural resources to include what is commonly taught in computer science, information systems, mathematics, statistics, and machine learning. Emphasizing these skillsets within natural resources recognizes the potential of big data and how the tenets of data science can be used to inform better decisions. In this dissertation, I addressed knowledge gaps in relating field data to remotely sensed imagery, sample design, image normalization, and converting data into useful information. However, to convert the types of information described in Chapter 4 into decisions that can be implemented on the ground will require translating findings into knowledge and communicating their importance to managers and practitioners [7]. While I acknowledge that in the short run specialists will need to help translate information, perform many of the analyses, provide results to decision makers, and even market the need and potential of these technologies, I argue that decision makers will eventually need to be educated in the tenets of data and decision science to fully take advantage of the volume, velocity, and variety of the ever expanding nexus of available data streams.

To this end future research should be equally focused on translating big data into useful information and building new technologies to address big data challenges, as using the types of information created in Chapter 4 in the decision making process to improve organizational operations and outcomes. While the former topic tends to dominate the scientific literature, the operational and translational piece [10, 11] continues to lag far behind [6]. This is not to suggest that

the applied or practitioner role should become the focus of scientists and researchers, but instead highlights that further manipulation of useful information such as described in Chapter 4 is often needed to incorporate complex and often competing objectives within natural resource management.

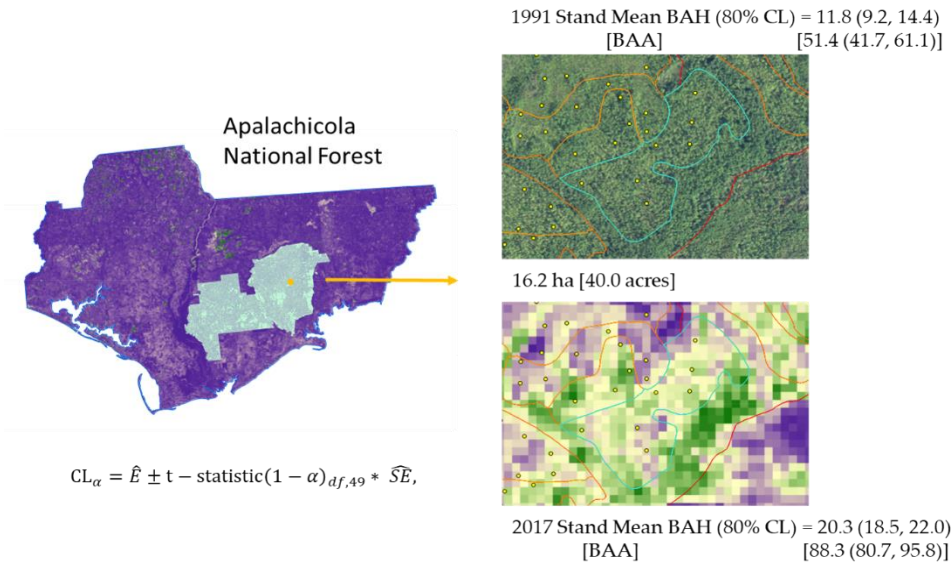
For the scientist, researcher, and developer this type of research requires a larger upfront investment and appreciation for the existing organizational structure, workflows, goals, and objectives of a given organization (e.g., agency, non-governmental organization, business, or company) followed by a larger role in implementing recommendations and change. For managers, professionals, technicians, and practitioners this big data transition requires a deeper appreciation for the potential utility of technological advances and research findings and an ongoing desire to learn and continually adapted to new concepts and ideas that are proven to help reach, improve upon, and promote organizational goals and objectives. Of special interest here is communication and the flows of knowledge back and forth from the managers, practitioner, technicians, analysts, researchers, scientists, and developers.

Specifically, within a data centric construct, it is false to assume that the organizational structure is independent from the operations occurring within the organization. In reality, both are tightly coupled and data, information, knowledge, and action are shared commodities flowing through a digital information system designed to digitally represent an organization and its workflows. This means that data are part of and at the forefront of transactions and interactions occurring within the entity and that digital systems must be designed such to store and access those transactions and interactions, not simply for reporting or retrieval but for the purpose of informed decision making, learning, and continual improvement in the organization.

For example, Chapter 4 could be published in the scientific literature and the raster surfaces created in Chapter 4 could be stored on a file server for download by the Apalachicola National Forest (ANF) analysts in Florida. On one level this provide managers, professionals, technicians, and practitioners access to concepts and ideas used to create the datasets and the surfaces themselves. However, because the techniques and datasets are not directly integrated into the digital representation of ANF (the organization), it is unlikely that they will be directly used by employees to actively meet forest objectives. Alternatively, if the techniques and datasets described within Chapter 4 were packaged as part of the ANF digital framework and were integrated into ANF workflows as functions, tools, user forms, and datasets, it is much more likely that those data and techniques will inform forest management decisions (Figure 1-3). Similarly, knowing the objectives, workflows, and goals of ANF, and tailoring research questions in part to address those goals and objectives makes scientific discoveries and inventions relevant to a large user base. Equally important though, adopting such an integrated approach has the potential to streamline the scientific method while simultaneously moving scientific discovery and development forward at a faster pace.

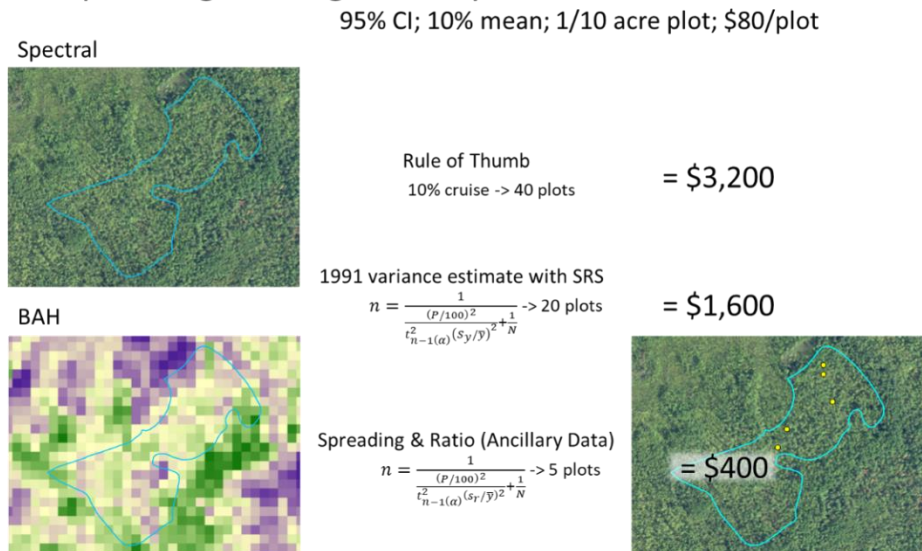


## Calculating Pine BAH Confidence Intervals



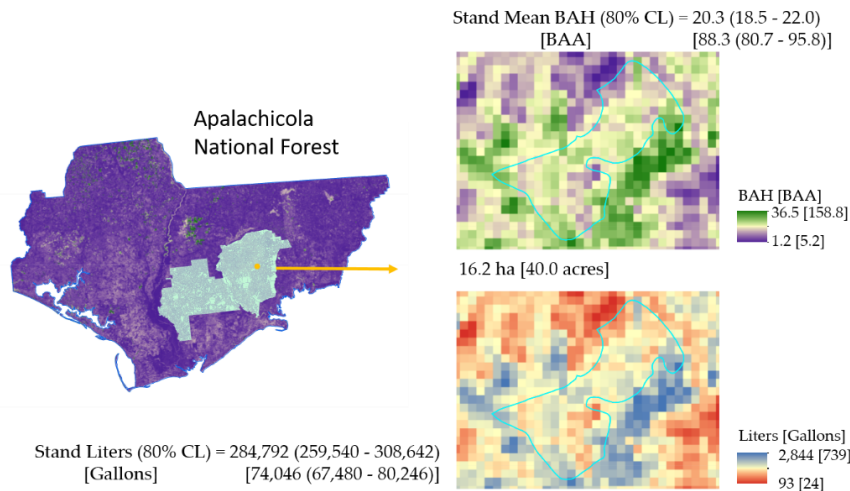
**Figure 1.** An example of informing decision making. If the raster surfaces from chapter 4 were directly integrated into Apalachicola National Forest (ANF) digital enterprise with corresponding summary tools, stand estimates of pine basal area  $\text{m}^2, \text{ha}^{-1}$  (BAH) or  $\text{ft}^2, \text{ac}$  [BAA] could be compared against previous estimates to assess if management objectives are being met. In this example a 16.2 ha ANF stand measured in 1991 had a pine BAH of  $11.8 \text{ m}^2 \text{ ha}^{-1}$ . Using an integrated approach analysts could summarize raster surface cell values presented in chapter 4 for the area within a given stand (blue outline) to determine that in 2017 pine BAH has increased to  $20.3 \text{ m}^2 \text{ ha}^{-1}$ .

## Spreading & Using Ancillary Data



**Figure 2.** A second example of informing decision making. If the raster surfaces from chapter 4 were directly integrated into Apalachicola National Forest (ANF) digital enterprise and a plot allocation tool was developed based on findings from chapter 3, ANF employees could substantially reduce stand inventory costs. In this example if a forester wanted to be 95% sure their stand estimate of pine basal area ( $\text{m}^2 \text{ ha}^{-1}$ ) was within 10% of the true mean (blue outlined polygon), they could spread their samples using the raster surface values from chapter 4 and further use a ratio estimator [12] to spatially allocate plots (yellow points within the blue polygon in southeastern graphic) and reduce inventory cost from \$3,200 to \$400, assuming each tenth acre plot cost \$80 to collect.

## Pine BAH to Maximum Liters of Water/Day



**Figure 3.** A third example of informing decision making. If the raster surfaces from chapter 4 were directly integrated into Apalachicola National Forest (ANF) digital enterprise and tools were developed to transform estimates of basal area ( $\text{m}^2 \text{ha}^{-1}$ ) to maximum liters of water  $\text{day}^{-1}$  transpired by pine trees (based on [13]), hydrologists could quantify at fine spatial detail where and how much water is released into the atmosphere by pine trees and make informed recommendations related to tradeoffs in management and water quantity. In this example the 16.2 ha stand outlined in light blue transpires a maximum of 284,792 liters of water  $\text{day}^{-1}$  with 80% confidence limits reported in parentheses.

## References

1. MODIS. TERRA The EOS Flagship, Available online: <https://terra.nasa.gov/about>, (accessed 31 of October, 2019)
2. United States Geological Survey [USGS]. Landsat 8. Available online: [https://www.usgs.gov/land-resources/nli/landsat/landsat-8?qt-science\\_support\\_page\\_related\\_con=0#qt-science\\_support\\_page\\_related\\_con](https://www.usgs.gov/land-resources/nli/landsat/landsat-8?qt-science_support_page_related_con=0#qt-science_support_page_related_con) (accessed on 23 of October, 2019).
3. Earth Observing System [EOS]. Sentinel-2. Available online: <https://eos.com/sentinel-2/> (accessed on 23 of October 2019).
4. National Agriculture Imagery Program [NAIP]. National Agriculture Imagery Program (NAIP) Information Sheet. Available online: [http://www.fsa.usda.gov/Internet/FSA\\_File/naip\\_info\\_sheet\\_2013.pdf](http://www.fsa.usda.gov/Internet/FSA_File/naip_info_sheet_2013.pdf) (accessed on 14 May 2014).
5. Markwo, S.; Braganza, S.; Taska, B. The Quant Crunch - How the Demand for Data Science Skills is Disrupting the Job Market, Burning Glass Technologies Technical Report, 2017, online: [https://www.burning-glass.com/wp-content/uploads/The\\_Quant\\_Crunch.pdf](https://www.burning-glass.com/wp-content/uploads/The_Quant_Crunch.pdf)
6. Gibert, K.; Horsburgh, J.S.; Athanasiadis, I.N.; Holmes, G. Environmental Data Science. *Environmental Modelling & Software*. 2018, 106, 4-12, doi: 10.1016/j.envsoft.2018.04.005.
7. Elshawi, R.; Sakr, S.; Talia, D.; Trunfio, P. Big Data Systems Meet Machine Learning Challenges: Towards Big Data Science as a Service, *Big Data Research*. 2018, 1, 1-11.
8. Sonka, M.; Hlavac, V.; Boyle, R. *Image Processing Analysis, and Machine Vision: Fourth Edition*. Cengage Learning, Stamford, CT. 2015, pp 869.
9. Davis, L.; Johnson, N. *Forest management, third edition*. McGraw Hill, New York, New York, 1987, pp 790.
10. Jackson, T.; Garfin, G. Enquist, C. Toward an effective practice of translational ecology. *Frontiers in Ecology and the Environment*. 2017. 15(10), 540.
11. Wall, T.; McNie, E.; Garfin, G. Use-inspired science: making science usable by and useful to decision makers. *Frontiers in Ecology and the Environment*. 2017, 15(10), 551-559.

12. Gregoire, T.; Valentine, H. *Sampling Strategies for Natural Resources and the Environment*, Chapman & Hall, Boca Raton London, New York, **2008**, 474 p.
13. Knight, D. H.; Fahey, T. J.; Running, S. W.; Harrison, A. T; Wallace, L. L. Transpiration From 100-yr-old Lodgepole Pine Forests Estimated with Whole-Tree Potometers. *Ecology*. **1981** 62, 717–726. doi:10.2307/1937740

## **Simulations & Statistical Analyses Libraries (R)**

## General Functions

#R library developed and used in Coregistration Simulations

#John Hogland 12/3/2018

```
library(raster)
library(gstat)
library(RStoolbox)
library(spsurvey)
library(rgdal)
library(rgeos)

rasterOptions(maxmemory = 1e+09)
rasterOptions(tmpdir = paste(getwd(), "tmp", sep = "/"))
tmpDir(create = TRUE)

###creates spatially correlated raster image.
#rws = number of rows
#clms = number of columns
#r = range
#mdl = model-> Sph, Exp, Gau, Mat
#mn = mean values
#vr = variance
#maxDist = maximum distance of spatial correlation (number of cells)
#sim = number of simulations'
getSpCorrImages <- function(rws, clms, r, mdl = "Sph", mn = 0, vr = 1, maxDist = 30,
ng = 0, sim = 1) {
  xy <- expand.grid(1:rws, 1:clms)
  names(xy) <- c("x", "y")
  g.dummy <- gstat(formula = z ~ 1, locations = ~x + y, dummy = T, beta = c(mn),
model = vgm(psil1 = vr, model = mdl, range = r, nugget = ng), nmax = maxDist)
  yy <- predict(g.dummy, newdata = xy, nsim = sim)
  outSt <- stack()
  for (k in seq(sim)) {
    yys <- subset(yy, select = c("x", "y", paste("sim", k, sep = "")))
    gridded(yys) = ~x + y
    rs <- raster(yys)
    outSt <- addLayer(outSt, rs)
  }
  return(outSt)
}
#sample a raster and return a spatial points layer
#rs = raster
```

```

#sampleSize = number of samples
#type = srs, sys, grts, grts_b, stratified, mask'
#stRS = stratum raster only for stratified
#prs= predictor raster only for grts_b
sampRaster <- function(rs, sampleSize, type = "srs", sp = TRUE, stRS = NULL, prs =
NULL, mask = NULL, buffDist=50) {
  outv1 <- NULL
  if (tolower(type) == "srs") {
    outv1 <- sampleRandom(rs, sampleSize, sp = sp)
  }
  if (tolower(type) == "sys") {
    outv1 <- sampleRegular2(rs, sampleSize)
  }
  if (tolower(type) == "stratified") {
    outv1 <- stratifiedSample(rs, sampleSize, stRs)
  }
  if (tolower(type) == "grts") {
    outv1 <- grtsSample(rs, sampleSize)
  }
  if (tolower(type) == "grts_b") {
    outv1 <- grtsSampleB(rs, sampleSize, prs)
  }
  if (tolower(type) == "mask") {
    outv1 <- maskSample(rs, sampleSize, mask,buffDist)
  }
  return(outv1)
}
maskSample <- function(rs, sampleSize, mask, buffDist=50) {
  flNm <- strsplit(basename(mask),".shp")[[1]]
  dirPath <- dirname(mask)
  roadLayer <- readOGR(dsn = dirPath, layer=flNm)
  rdBuff <- gBuffer(roadLayer, width = buffDist)
  rsm <- mask(rs, rdBuff)
  srs <- sampRaster(rsm, sampleSize, 'srs')
  return(srs)
}
stratifiedSample <- function(rs, sampleSize, stRs) {
  return(outv1)
}
grtsSample <- function(rs, sampleSize) {
  ext <- extent(rs)
  poly <- as(ext, 'SpatialPolygons')
  crs(poly) <- crs(rs)
}

```

```

    Equaldsgn <- list(None = list(panel = c(PanelOne = sampleSize), seltype =
"Equal"))
    df <- data.frame(st = 1)
    spdf <- SpatialPolygonsDataFrame(poly, df)
    outvl <- grts(design = Equaldsgn, DesignID = 'st', type.frame = "area",
src.frame = "sp.object", sp.object = spdf, shapefile = F)
    tb <- as.data.frame(extract(rs, outvl@coords))
    spdf <- SpatialPointsDataFrame(outvl@coords, tb)
    return(spdf)
}
grtsSampleB <- function(rs, sampleSize, prs) {
  srs <- sampRaster(prs, 10000, "srs", sp = T)
  pca <- princomp(srs@data, cor = TRUE)
  srs@data$uid <- 1:nrow(srs@data)
  sc <- pca$scores
  df <- data.frame(uid = 1:nrow(sc), Comp.1 = sc[, 1], Comp.2 = sc[, 2])
  Equaldsgn <- list(None = list(panel = c(PanelOne = sampleSize), seltype =
"Equal"))
  outvl <- grts(design = Equaldsgn, type.frame = "finite", src.frame =
'att.frame', att.frame = df, xcoord = "Comp.1", ycoord = "Comp.2", shapefile = F)
  outdf <- merge(outvl@data, df, by = c('Comp.1', 'Comp.2'))
  coords <- as.data.frame(srs@coords)
  coords$uid <- (1:nrow(sc))
  scoords <- subset(coords, coords$uid %in% outdf$uid.x)
  coordinates(scoords) = c("x", "y")
  tb <- as.data.frame(extract(rs, scoords@coords))
  scoords@data = tb
  return(scoords)
}
#adjustment to regular sample routine to implement random start
sampleRegular2 <- function(rs3, sampSize) {
  xl <- xmin(rs3)
  xr <- xmax(rs3)
  yb <- ymin(rs3)
  yt <- ymax(rs3)
  a <- (xr - xl) * (yt - yb)
  skip <- sqrt(a / sampSize)
  r_rc <- runif(2, 1, (skip / 2))
  iterx <- seq(xl + r_rc[1], xr, skip)
  itery <- seq(yb + r_rc[2], yt, skip)
  x <- vector(mode = 'double', length(iterx) * length(itery))
  y <- vector(mode = 'double', length(x))
  cnt <- 1

```

```

for (i in iterx) {
  for (j in itery) {
    x[cnt] <- i
    y[cnt] <- j
    cnt <- cnt + 1
  }
}
coords <- cbind(x, y)
tb <- as.data.frame(extract(rs3, coords))
spdf <- SpatialPointsDataFrame(coords, tb, proj4string = crs(rs3))
return(spdf)
}

#shift an existing set of locations to simulate co-registration errors and returns
the shifted locations as a dataframe
#pts = Spatial points dataset
#gpsError = RMSE in gps expressed as cells
#imgError = RMSE in image rectification expressed as cells'
shiftXY <- function(pts, d1, d2) {
  x <- pts$X
  y <- pts$Y
  cnt <- length(x)
  #d1 <- rnorm(cnt, mean = 0, sd = gpsError)
  #d2 <- rnorm(cnt, mean = 0, sd = imgError)
  b1 <- runif(cnt, min = 0, (2 * pi))
  b2 <- runif(cnt, min = 0, (2 * pi))
  xp1 <- sin(b1) * d1
  yp1 <- cos(b1) * d1
  xp2 <- sin(b2) * d2
  yp2 <- cos(b2) * d2
  xpd <- xp1 + xp2
  ypd <- yp1 + yp2
  df <- data.frame(X = round(xpd + x), Y = round(ypd + y), D = round(sqrt(xpd ** 2
+ ypd ** 2)))
  return(df)
}

#get full image matrix
getImgMatrix <- function(rs) {
  return(getValuesBlock(rs, row = 1, nrows = nrow(rs), col = 1, ncols = ncol(rs),
format = "matrix", byrow = TRUE))
}

#extract list of point block size matrix

```



```

#rcDf = locations
#rs = raster
extractRcLst <- function(rcDf, rsMatrix, blcSize) {
  outLst <- vector(mode = "list", length = nrow(rcDf))
  xc <- rcDf$X
  yc <- rcDf$Y
  for (r in 1:length(outLst)) {
    x <- xc[r]
    y <- yc[r]
    outLst[[r]] <- rsMatrix[y:(y + blcSize), x:(x + blcSize)]
#getValueBlock(rs, row = y, nrows = blcSize, col = x, ncols = blcSize, format =
'matrix')
  }
  return(outLst)
}
#gets multiple values and returns value lst
getMeanBlockValues <- function(lstBlc, lyt, prop) {
  rws <- length(lstBlc)
  mt <- matrix(ncol = 2, nrow = rws)
  for (r in 1:rws) {
    vl <- getMeanBlockValue(lstBlc[[r]], lyt, prop)
    mt[r,] <- vl
  }
  return(mt)
}
#extract values from raster: return dataframe
#rcDf = locations
#rs = raster'
extractRC <- function(rcDf, rs, blcSize, pltLayout = 1, pltProp = 1) {
  rws <- nrow(rcDf)
  mt <- matrix(ncol = 2, nrow = rws)
  xc <- rcDf$X
  yc <- rcDf$Y
  for (r in 1:rws) {
    x <- xc[r]
    y <- yc[r]
    vl <- getMeanBlockValue(getValuesBlock(rs, row = y, nrows = blcSize, col =
x, ncols = blcSize, format = 'matrix'), pltLayout, pltProp)
    mt[r,] <- vl
  }
  return(mt)
}
#summarizes values within a block given a specified layout

```

```

#blc=matrix of cell values
#lyt = plot layout [1,2,3,4,5,6] 1=center, 2=4 corners, 3=4 random, 4=FIA,
5=9equal,6=corners & center
#prop = proportion of area sampled within the plot extent
getMeanBlockValue <- function(blc, lyt, prop) {
  outVl <- 0
  outProp <- 0
  rws <- nrow(blc)
  clm <- ncol(blc)
  tCells <- sum(!is.na(blc))
  #print(paste("total cells:",tCells,sep=" "))
  if (prop == 1) {
    outVl <- mean(blc, na.rm = TRUE)
    outProp <- 1
  }
  else {
    if (lyt == 1) {
      bSize <- round(sqrt(tCells * prop))
      #print(paste("blockSize:", bSize, sep = " "))
      bSize2 <- bSize - 1
      hb <- round(bSize / 2)
      brw <- round(rws / 2) - hb
      bclm <- round(clm / 2) - hb
      outVl <- mean(blc[brw:(brw + bSize2), bclm:(bclm + bSize2)], na.rm =
TRUE)

      ccnt <- sum(!is.na(blc[brw:(brw + bSize2), bclm:(bclm + bSize2)]))
      #print(paste("ccnt:", ccnt, sep = " "))
      outProp <- ccnt / tCells
    }
    else if (lyt == 2) {
      bSize <- round(sqrt((tCells * prop) / 4))
      bSize2 <- bSize - 1
      ccnt <- 0 #bSize**2*4
      #print(paste("BSize2 =",bSize2,sep=" "))
      ms <- blc[1:bSize, 1:bSize]
      m1 <- sum(ms, na.rm = TRUE)
      ccnt <- ccnt + sum(!is.na(ms))
      ms <- blc[(rws - bSize2):rws, (clm - bSize2):clm]
      m2 <- sum(ms, na.rm = TRUE)
      ccnt <- ccnt + sum(!is.na(ms))
      ms <- blc[(rws - bSize2):rws, 1:bSize]
      m3 <- sum(ms, na.rm = TRUE)
      ccnt <- ccnt + sum(!is.na(ms))
    }
  }
}

```

```

ms <- b1c[1:bSize, (c1m - bSize2):c1m]
m4 <- sum(ms, na.rm = TRUE)
ccnt <- ccnt + sum(!is.na(ms))
outV1 <- (m1 + m2 + m3 + m4) / ccnt
outProp <- ccnt / tCells
}
else if (lyt == 3) {
  bSize <- round(sqrt((tCells * prop) / 4))
  bSize2 <- bSize - 1
  ccnt <- 0 #bSize**2*4
  rC1m <- as.integer(runif(4, 1, c1m - bSize2))
  rRw <- as.integer(runif(4, 1, rws - bSize2))
  sTot <- 0
  for (i in 1:4) {
    r <- rRw[i]
    c <- rC1m[i]
    ms <- b1c[r:(r + bSize2), c:(c + bSize2)]
    sTot <- sTot + sum(ms, na.rm = TRUE)
    ccnt <- ccnt + sum(!is.na(ms))
    #print(paste("rc:", r, c, sep = " "))
    #print(paste("tot:", sTot, sep = " "))
  }
  outV1 <- sTot / ccnt
  outProp <- ccnt / tCells
}
else if (lyt == 4) {
  bSize <- round(sqrt((tCells * prop) / 4))
  bSize2 <- bSize - 1
  ccnt <- 0 #bSize**2*4
  hbSize <- round(bSize / 2)
  hc <- round(c1m / 2) - hbSize
  hr <- round(rws / 2) - hbSize
  ms <- b1c[hr:(hr + bSize2), hc:(hc + bSize2)]
  m1 <- sum(ms, na.rm = TRUE)
  ccnt <- ccnt + sum(!is.na(ms))
  ms <- b1c[(rws - bSize2):rws, (c1m - bSize2):c1m]
  m2 <- sum(ms, na.rm = TRUE)
  ccnt <- ccnt + sum(!is.na(ms))
  ms <- b1c[(rws - bSize2):rws, 1:bSize]
  m3 <- sum(ms, na.rm = TRUE)
  ccnt <- ccnt + sum(!is.na(ms))
  ms <- b1c[1:bSize, hc:(hc + bSize2)]
  m4 <- sum(ms, na.rm = TRUE)

```

```

ccnt <- ccnt + sum(!is.na(ms))
sTot <- m1 + m2 + m3 + m4
#print(paste("tot:", sTot, sep = " "))
outV1 <- sTot / ccnt
outProp <- ccnt / tCells
}
else if (lyt == 5) {
  bSize <- round(sqrt((tCells * prop) / 9))
  bSize2 <- bSize - 1
  ccnt <- 0 #bSize**2*9
  hbSize <- round(bSize / 2)
  hc <- round(rws / 2) - hbSize
  hr <- round(clm / 2) - hbSize
  rClm <- c(1, hc, (clm - bSize))
  rRw <- c(1, hr, (rws - bSize))
  sTot <- 0
  for (i in 1:3) {
    for (j in 1:3) {
      r <- rRw[i]
      c <- rClm[j]
      ms <- blc[r:(r + bSize2), c:(c + bSize2)]
      sTot <- sTot + sum(ms, na.rm = TRUE)
      ccnt <- ccnt + sum(!is.na(ms))
      #print(sTot)
    }
  }
  outV1 <- sTot / ccnt
  outProp <- ccnt / tCells
}
else if (lyt == 6) {
  bSize <- round(sqrt((tCells * prop) / 5))
  bSize2 <- bSize - 1
  ccnt <- 0 #bSize**2*5
  hbSize <- round(bSize / 2)
  hc <- round(clm / 2) - hbSize
  hr <- round(rws / 2) - hbSize
  ms <- blc[hr:(hr + bSize2), hc:(hc + bSize2)]
  m1 <- sum(ms, na.rm = TRUE)
  ccnt <- ccnt + sum(!is.na(ms))
  ms <- blc[1:bSize, 1:bSize]
  m2 <- sum(ms, na.rm = TRUE)
  ccnt <- ccnt + sum(!is.na(ms))
  ms <- blc[(rws - bSize2):rws, (clm - bSize2):clm]

```

```

    m3 <- sum(ms, na.rm = TRUE)
    ccnt <- ccnt + sum(!is.na(ms))
    ms <- b1c[(rws - bSize2):rws, 1:bSize]
    m4 <- sum(ms, na.rm = TRUE)
    ccnt <- ccnt + sum(!is.na(ms))
    ms <- b1c[1:bSize, (clm - bSize2):clm]
    m5 <- sum(ms, na.rm = TRUE)
    ccnt <- ccnt + sum(!is.na(ms))
    sTot <- m1 + m2 + m3 + m4 + m5
    #print(sTot)
    outV1 <- sTot / ccnt
    outProp <- ccnt / tCells
  }
  else {
    outV1 <- 0
    outProp <- 0
  }
}
return(c(outV1, outProp))
}
#get vector of images
#dir = search directory
#ext=extension of images'
getImgPath <- function(imgDir, ext = ".tif") {
  outVec = c()
  imgNames <- list.files(path = imgDir, pattern = ext)
  for (i in imgNames) {
    ext2 <- extension(i)
    if (tolower(ext2) == ext) {
      outPath <- paste(imgDir, i, sep = "\\")
      outVec = c(outVec, outPath)
    }
  }
  return(outVec)
}
#returns max sill (nugget + psill), max (range) in cells
getSampledMaxSillRange <- function(rsPath, sSize = 20, blSize = 200) {
  r <- raster(rsPath)
  outDf <- data.frame(nm = character(), bnd = double(), mn = double(), sill =
double(), rng = double(), nug = double(), cnt = integer())
  rndX <- as.integer(runif(sSize, blSize, nrow(r) - blSize))
  rndY <- as.integer(runif(sSize, blSize, ncol(r) - blSize))
  for (b in 1:nbands(r)) {

```

```

print(paste("Working on band", b, sep = " "))
mVar <- 0
mRange <- 0
mMean <- 0
mNugget <- 0
rs <- raster(rsPath, b)
dn <- 0
for (bl in 1:sSize) {
  mtV1 <- getValuesBlock(rs, row = rndX[bl], nrow = blSize, col =
rndY[bl], ncol = blSize, format = 'matrix', byrow = T)
  rs2 <- raster(mtV1)
  mMean <- mMean + cellStats(rs2, mean)
  cellSize <- res(rs2)[1]
  frm <- formula(paste(names(rs2), "1", sep = "~"))
  v <- variogram(frm, as(rs2, "SpatialPixelsDataFrame"))
  f <- tryCatch(fit.variogram(v, vgm("Sph")), warning = function(w)
return(NA), error = function(e) return(NA))
  if (is.na(f)) {
    print(paste("Error with", names(r), b, bl, sep = " "))
  }
  else {
    mVar <- mVar + f$psill[2]
    mNugget <- mNugget + f$psill[1]
    mRange <- mRange + (f$range[2] / cellSize)
    dn <- dn + 1
  }
}
outDf <- rbind(outDf, data.frame(Name = names(r), Band = b, Mean = mMean /
dn, Sill = mVar / dn, Rng = mRange / dn, Nugget = mNugget / dn, cnt = dn))
}
names(outDf) <- c("Name", "Band", "Mean", "Sill", "Rng", "Nugget", "Count")
return(outDf)
}
#creates a random set of cells, returns data frame of row and column
getRandomCells <- function(rs, numCells, insideBuffer, moff) {
  w <- ncol(rs)
  h <- nrow(rs)
  rx <- as.integer(runif(numCells, moff, w - (insideBuffer + moff)))
  ry <- as.integer(runif(numCells, moff, h - (insideBuffer + moff)))
  return(data.frame(X = rx, Y = ry))
}
#perform simulations

```

```

SimSampling <- function(ImgPaths, sampSize, blockWidth, shiftGps, shiftImg, prop =
1) {
  outDf <- data.frame(NAME = character(), BAND = double(), BLOCKSIZE = double(),
LAYOUT = integer(), EPROP = double(), TPROP = double(), TYPE = integer(), RMSE =
double(), MAE = double(), INTERCEPT = double(), SLOPE = double(), R2 = double(), N =
integer(), MORI = double(), OVERLAP = double())
  for (i in seq(length(ImgPaths))) {
    pt <- ImgPaths[i]
    print(paste("Working on ", pt, sep = ""))
    rs <- raster(pt)
    nm <- names(rs)
    bndCnt <- nbands(rs)
    d1 <- rnorm(sampSize, mean = 0, sd = shiftGps)
    d2 <- rnorm(sampSize, mean = 0, sd = shiftImg)
    d3 <- max(d1) + max(d2) + 1
    pnts <- getRandomCells(rs, sampSize, max(blockWidth), d3)
    spnts <- shiftXY(pnts, d1, d2)
    for (b in 1:bndCnt) {
      print(paste(" Band ", b, sep = ""))
      rsb <- convertRasterBandToFloat(raster(pt, b))
      mori <- Moran(rsb, matrix(c(0, 1, 0, 1, 0, 1, 0, 1, 0), nrow = 3))
#rooks case
      rsM <- getImgMatrix(rsb)
      for (bl in blockWidth) {
        print(paste(" Block ", bl, sep = ""))
        shiftMt <- extractRcLst(spnts, rsM, bl)
        #shiftVls <- extractRC(spnts, rsb, bl, 1, 1)
        shiftVls <- getMeanBlockValues(shiftMt, 1, 1)
        vlsMt <- extractRcLst(pnts, rsM, bl)
        chv <- prop[1]
        ovlp = 0
        if (chv == 1 || is.null(chv)) {
          #vls <- extractRC(pnts, rsb, bl, 1, 1)
          ovlp <- getOverLap(pnts, spnts, bl)
          vls <- getMeanBlockValues(vlsMt, 1, 1)
          tDf <- getLmDf(vls, shiftVls)
          outDf <- rbind(outDf, data.frame(NAME = nm, BAND = b, BLOCKSIZE
= bl, LAYOUT = 0, EPROP = 1, TPROP = 1, TYPE = 0, RMSE = tDf$RMSE, MAE = tDf$MAE,
INTERCEPT = tDf$INTERCEPT, SLOPE = tDf$SLOPE, R2 = tDf$R2, N = tDf$N, MORI = mori,
OVERLAP = ovlp))
        }
        else {
          for (j in seq(1, 6, 1)) {

```

```

print(paste("      Layout ", j, sep = ""))
for (p in prop) {
  print(paste("      Proportion ", p, sep = ""))
  vls <- getMeanBlockValues(vlsMt, j, p)
  mprop <- mean(vls[, 2])
  shiftVls2 <- getMeanBlockValues(shiftMt, j, p)
  tDf <- getLmDf(vls, shiftVls)
  outDf <- rbind(outDf, data.frame(NAME = nm, BAND = b,
BLOCKSIZE = b1, LAYOUT = j, EPROP = p, TPROP = mprop, TYPE = 1, RMSE = tDf$RMSE, MAE
= tDf$MAE, INTERCEPT = tDf$INTERCEPT, SLOPE = tDf$SLOPE, R2 = tDf$R2, N = tDf$N,
MORI = mori, OVERLAP = ovlp))
  tDf <- getLmDf(vls, shiftVls2)
  outDf <- rbind(outDf, data.frame(NAME = nm, BAND = b,
BLOCKSIZE = b1, LAYOUT = j, EPROP = p, TPROP = mprop, TYPE = 2, RMSE = tDf$RMSE, MAE
= tDf$MAE, INTERCEPT = tDf$INTERCEPT, SLOPE = tDf$SLOPE, R2 = tDf$R2, N = tDf$N,
MORI = mori, OVERLAP = ovlp))
}
}
}
}
}
}
removeTmpFiles(h = 0)
return(outDf)
}

```

#Calculates mean overlap between shifted points given a block size

```

getOverLap <- function(pnts, spnts, b1) {
  s <- 0
  ta <- b1 ^ 2
  for (i in 1:nrow(pnts)) {
    l1 <- pnts[i, 1]
    r1 <- l1 + b1
    b1 <- pnts[i, 2]
    t1 <- b1 + b1
    l2 <- spnts[i, 1]
    r2 <- l2 + b1
    b2 <- spnts[i, 2]
    t2 <- b2 + b1
    ext1 <- extent(l1, r1, b1, t1)
    ext2 <- extent(l2, r2, b2, t2)
    int2 <- intersect(ext1, ext2)
    if (!is.null(int2)) {
      x <- xmax(int2) - xmin(int2)
    }
  }
}

```



```

        y <- ymax(int2) - ymin(int2)
        s <- s + x * y
    }
}
return(s / (nrow(pnts) * ta))
}
#performs linear model
getLmDf <- function(vls, shiftVls) {
  md <- lm(vls[, 1] ~ shiftVls[, 1])
  mae <- mean(residuals(md))
  rmse <- mean(residuals(md) ** 2) ** 0.5
  coef <- md$coefficients
  int <- coef[1]
  slp <- coef[2]
  r2 <- (summary(md))$r.squared
  n <- nrow(vls)
  return(data.frame(RMSE = rmse, MAE = mae, INTERCEPT = int, SLOPE = slp, R2 = r2,
N = n))
}
#printMoransI
printMoransI <- function(imgPaths, m = matrix(c(0, 1, 0, 1, 0, 1, 0, 1, 0), nrow =
3)) {
  for (i in imgPaths) {
    rs <- raster(i)
    bndCnt <- nbands(rs)
    for (j in 1:bndCnt) {
      rsb <- convertRasterBandToFloat(raster(i, j))
      mori <- Moran(rsb)
      print(paste(i, j, mori, sep = " "))
    }
  }
  removeTmpFiles(h = 0)
}
#setFloat
convertRasterBandToFloat <- function(rsb) {
  dt <- dataType(rsb)
  outRs <- rsb
  if (dt == "FLT4S" || dt == "FLT8S") {
  }
  else {
    outRs <- rsb * 1.0
  }
  return(outRs)
}

```

```

}
#compare to a set number and return a vector of numbers
compareValues <- function(df, fldIndex = 13, stepValue = 40) {
  if (is.character(fldIndex)) {
    fldIndex <- match(fldIndex, names(df))
  }
  rIndx <- 0
  dr2v <- vector(mode = "numeric", length = nrow(df))
  for (r in 1:nrow(df)) {
    r2 <- df[r, fldIndex]
    chx <- r %% stepValue
    if (chx == 1) {
      rIndx <- rIndx + stepValue
    }
    r2c <- df[rIndx, fldIndex]
    dr2v[r] <- r2c - r2
  }
  return(dr2v)
}
#estimate spatial offsets
estOffset <- function(gpsError = 6, imageError = 7, iterations = 1000, sampSize =
200, domainSize = 1000) {
  ss <- 0
  for (i in seq(iterations)) {
    xyDf <- data.frame(X = runif(sampSize, 0, domainSize), Y = runif(sampSize,
0, domainSize))
    sDf <- shiftXY(xyDf, rnorm(sampSize, 0, imageError), rnorm(sampSize, 0,
gpsError))
    ss <- ss + sum(sDf$D)
  }
  return(ss / (sampSize * iterations))
}
#transform raster
transformRaster <- function(rs, outName, type = "linear") {
  #rs = input raster
  #type = string [linear, exp, nonlinear]
  bndCnt <- nbands(rs)
  outRs <- convertRasterBandToFloat(subset(rs, 1))
  for (b in 2:bndCnt) {
    outRs <- outRs + subset(rs, b)
  }
  outRs <- outRs / bndCnt
  if (type == "squared") {

```

```

        outRs <- outRs ^ 2
    }
    if (type == "nonlinear") {
        tRs <- subset(rs, 4)
        tRs <- (tRs < 210 & tRs > 170)
        outRs <- outRs * tRs
    }
    outRs <- writeRaster(outRs, paste(outName, ".tif", sep = ""), format = "GTiff",
overwrite = TRUE)
    removeTmpFiles(h = 0)
    return(outRs)
}
#create a clustered image and kmeans model
KmeansClassificationRs <- function(rs, outName, k = 10, ss = 10000) {
    #rs = raster Brick
    #outName = string
    #k = number of clusters
    #returns raster and model list(rs,model)
    km <- unsuperClass(rs, nClasses = k, nSamples = ss)
    km_model <- km$model
    outRs <- writeRaster(km$map, paste(outName, ".tif", sep = ""), format = "GTiff",
overwrite = TRUE)
    removeTmpFiles(h = 0)
    return(list(outRs, km_model))
}
#create a principal component analysis from a sample of points in a raster
pcaTrans <- function(rs, outName, ss = 10000) {
    #rs=raster Brick
    #outName = string
    #ss=sample size
    #returns a raster and PCS model; list(rs,PCA)
    bndCnt <- nbands(rs)
    samp <- sampRaster(rs, ss)
    pca <- princomp(samp@data, cor = TRUE)
    outRs <- predict(rs, pca, index = 1:bndCnt, paste(outName, ".tif", sep = ""),
format = "GTiff", overwrite = TRUE)
    removeTmpFiles(h = 0)
    return(list(outRs, pca))
}
#create continuous error for image
ErrorRs <- function(rs, outName, type = 'norm', p_error = 0.2) {
    #rs=raster surface
    #type = type of error (norm, poisson, gamma)

```

```

#p_error = percent of standard deviation
#returns a raster
mu <- cellStats(rs, mean)
std <- cellStats(rs, sd)
adj <- std * p_error
cellCnt <- ncell(rs)
outRs <- raster(rs)
if (type == 'poisson') {
  outRs <- setValues(outRs, (rpois(cellCnt, mu) - mu)) + rs
}
if (type == 'gamma') {
  outRs <- setValues(outRs, (rgamma(cellCnt, 5, 5) * adj) - adj) + rs
}
if (type == 'norm') {
  outRs <- setValues(outRs, rnorm(cellCnt, mean = 0, sd = adj)) + rs
}
}
outRs <- writeRaster(outRs, paste(outName, ".tif", sep = ""), format = "GTiff",
overwrite = TRUE)
removeTmpFiles(h = 0)
return(outRs)
}
#creates classification errors
ErrorRsClass <- function(rs, outName, p_error = 0.3, k = 10) {
  #rs = raster type
  #p_error = percent cells with error
  #returns a raster
  cellCnt <- ncell(rs)
  r1 <- raster(rs)
  r1 <- setValues(r1, as.integer(runif(cellCnt) > p_error))
  r4 <- raster(rs)
  r4 <- setValues(r4, as.integer(runif(cellCnt, 1, k)))
  outRs <- (rs * r1) + (r4 * (1 - r1))
  outRs <- writeRaster(outRs, paste(outName, ".tif", sep = ""), format = "GTiff",
datatype = 'INT1U', overwrite = TRUE)
  removeTmpFiles(h = 0)
  return(outRs)
}
#KS test for two populations
#pop = data frame of columns, all columns will be used
#samp = data frame of columns with same names as pop
ksPcaTest <- function(pop, samp) {
  pca <- princomp(pop, cor = T)

```

```

ws<-pca$sdev^2/sum(pca$sdev^2)
pValLst <- vector(mode = "double", length = ncol(pop))
ksValLst <- vector(mode = "double", length = ncol(pop))
for (i in 1:length(ksValLst)) {
  ks <- ksValLst[i] <- ks.test(pop[[i]], samp[[i]],)
  ksValLst[i] <- ks$statistic * ws[i]
  pValLst[i]<-ks$p.value * ws[i]
  return(c(sum(pValLst),sum(ksValLst)))
}
}

#' Get best GAM
#'
#' @param indata = input data frame
#' @param resp = response field name
#' @param pred = vector of potential predictor variables
#' @param alpha = significance level used to select variables (0.05)
#' @param fam = family distribution (gaussian())
#' @param improveby = the amount needed to improve % deviance explained to include a
predictor variable (0)
#'
#' @return = list of significant variable vector and GAM
#' @export
#'
#' @examples
getGamSigFldNames<-function(indata, resp, pred, alpha = 0.05, fam = gaussian(),
improveby = 0) {
  nlp = 1
  if (fam$family == "multinom") {
    nlp = fam$nlp
  }
  sigVar <- c()
  pdiv <- 0
  pr2 <- 0
  for (i in seq(length(pred))) {
    vars <- c(sigVar, pred[i])
    fm <- getFormula(resp,vars,nlp)
    md <- gam(fm, data = indata, family = fam)
    smry <- summary(md)
    div <- smry$dev.expl
    if (div > (pdiv + improveby)) {
      print(paste("Adding variable", pred[i], collapse = " "))
      pvalues <- c(smry$s.pv)

```

```

sigVar <- c()
nonSigVar <- c()
for (j in seq(length(vars))) {
  pv <- getSmallestPvalue(j,pvalues,nlp)
  if (pv <= alpha) {
    sigVar <- c(sigVar, vars[j])
  }
  else {
    nonSigVar <- c(nonSigVar, vars[j])
  }
}
if (length(nonSigVar) > 0) {
  for (k in nonSigVar) {
    print(paste(cat("\t"), "Rechecking non significant variables",
k, collapse = " "))
    vars2 <- c(sigVar, k)
    cfm <- getFormula(resp,vars2,nlp)
    nmd <- gam(cfm, data = indata, family = fam)
    nsmry <- summary(nmd)
    ndiv <- nsmry$dev.expl
    pvalues <- c(nsmry$s.pv)
    cpvalue <- getSmallestPvalue(length(vars2),pvalues,nlp)
    if (cpvalue <= alpha) {
      sigVar <- c(sigVar, k)
      print(paste(cat("\t"), "adding", k, "back to the model",
collapse = " "))
    }
  }
  ndiv <- pdiv
  if (length(sigVar) > 0) {
    cfm <- getFormula(resp,sigVar,nlp) #as.formula(paste(resp, " ~
", paste("s(", sigVar, ")", collapse = " + ")))
    nmd <- gam(cfm, data = indata, family = fam)
    nsmry <- summary(nmd)
    ndiv <- nsmry$dev.expl
  }
  if (ndiv < (pdiv + improveby)) {
    print(paste(cat("\t"), "No improvement. Changing sig variables
back to what they previously were"))
    sigVar <- vars[1:length(vars) - 1]
  }
  else {
    pdiv <- ndiv

```

```

        }
    }
    else {
        pdiv <- div
    }
    print(paste(cat("\t"), "sig var for iter ", i, "(%Div = ", pdiv, "):",
paste(sigVar, collapse = " "))
    }
}
print(paste(cat("\t"), "Removing variables that do not meet significance
level", sep = ""))
fmd <- removeLeastSignificantVar(resp, indata, sigVar, fam, alpha, nlp)
while (!is.null(fmd[[1]])) {
    sigVar = fmd[[1]]
    fmd = removeLeastSignificantVar(resp, indata, sigVar, fam, alpha, nlp)
}
return(list(sigVar, fmd[[2]]))
}
removeLeastSignificantVar <- function(resp, indata, sigVar, fam, alpha, nlp) {
    cfm <- getFormula(resp, sigVar, nlp)
    nmd <- gam(cfm, data = indata, family = fam)
    nsmry <- summary(nmd)
    pvalues <- c(nsmry$s.pv)
    fVar <- c()
    malpha = NULL
    nSigVar = NULL
    for (i in 1:length(sigVar)) {
        tpv <- getSmallestPvalue(i, pvalues, nlp)
        fVar <- c(fVar, tpv)
    }
    fVarT = fVar > alpha
    if (sum(fVarT) > 0) {
        malpha = max(fVar)
    }
    if (!is.null(malpha)) {
        mVarIndex = which(fVar == malpha)
        nSigVar = sigVar[-mVarIndex]
    }

    return(list(nSigVar, nmd))
}
getSmallestPvalue <- function(vlIndex, pvalues, nlp) {
    mVl = pvalues[vlIndex]

```

```

if (nlp > 1) {
  cVec <- vector(mode = "numeric", length = nlp)
  cnt = 1
  for (i in seq(vlIndex, length(pvalues), length(pvalues)/nlp)) {
    pv1 = pvalues[[i]]
    cVec[cnt] = pv1
    cnt = cnt + 1
  }
  mV1 = min(cVec)
}
return(mV1)
}
getFormula <- function(rVar, pVars, numlp) {
  fm=as.formula(paste(rVar, " ~ ", paste("s(", pVars, ")", collapse = " + ")))
  if (numlp > 1) {
    fml = list(length = numlp)
    fml[[1]] = fm
    for (f in 2:numlp) {
      fml[[f]] = as.formula(paste("~ ", paste("s(", pVars, ")", collapse = " +
")))
    }
    fm <- fml
  }
  return(fm)
}

```



## Chapter 2

#Empirically determines the plot size and layout for relating plots to in-situ data  
#John Hogland 12/3/2018

#setup directories

baseDir <-

"C:\\Users\\jshogland\\Documents\\John\\projects\\UMGradSchool\\Project\\Papers\\Dissertation\\chapter1\\data"

LandsatImgDir <- paste(baseDir, "Landsat", sep = "\\")

NaipImgDir <- paste(baseDir, "NAIP", sep = "\\")

setwd(baseDir)

source("C:\\Users\\jshogland\\Documents\\John\\projects\\UMGradSchool\\Project\\Papers\\Dissertation\\Rscripts\\Dissertation\\Dissertation\\jhLib.R")

#get general numbers for max continuity

#df.gNumbers<-

data.frame(img=character(),band=integer(),mean=double(),sill=double(),range=double()  
)

#for(p in imgPaths)

{

# print(tDf<-getSampledMaxSillRange(p))

# df.gNumbers<-rbind(df.gNumbers,tDf)

}

#create rasters

#for (i in seq(0.5,30,5))

{

# rs<-getSpCorrImages(1000,1000,i,mn=2080,vr=245416,maxDist=60,ng=?)

# writeRaster(rs,paste("Range",i,".tif",sep=""),format="GTiff",overwrite=TRUE)

}

LandsatImgPaths<-getImgPath(LandsatImgDir)

NaipImgPaths <- getImgPath(NaipImgDir)

#setup simulations

sampSize <- 200

blockWidth <- c(1, seq(3, 15, 3), seq(20, 55, 5), seq(60, 100, 10))

#Landsat 8 and GPS errors in pixels

shiftImg<-48/30#\*rmseImage #95% of the data (pixels) Landsat 37m 90% CI = 47.48m @  
95%

shiftGps<-7/30#\*rmseGps #95% of the data (pixels)

```

#using all the data within a plot boundary (blockSize) to determine size of the plot
for Landsat
dfRegL <- SimSampling(LandsatImgPaths, sampSize, blockWidth, shiftGps, shiftImg)
for (i in 1:9) {
  dfRegL <- rbind(dfRegL, SimSampling(LandsatImgPaths, sampSize, blockWidth,
  shiftGps, shiftImg))
}
write.csv(dfRegL, "LandsatBlockSizeReg10.csv")

#NAIP and GPS errors in pixels
shiftImg <- 6 #*rmseImage #95% of the data (pixels) NAIP
shiftGps <- 7 #*rmseGps #95% of the data (pixels)

#using all the data within a plot boundary (blockSize) to determine size of the plot
for NAIP
dfRegN <- SimSampling(NaipImgPaths, sampSize, blockWidth, shiftGps, shiftImg)
for (i in 1:9) {
  dfRegN <- rbind(dfRegN, SimSampling(NaipImgPaths, sampSize, blockWidth,
  shiftGps, shiftImg))
}
write.csv(dfRegN, "NaipBlockSizeReg10.csv")

#Using BlockSizes between 20 and 50 cells with varying layouts and proportions of
area sampled (0.5-0.95 by 0.5) NAIP
blockWidth <- seq(20, 50, 5)
dfRegLayN <- SimSampling(NaipImgPaths, sampSize, blockWidth, shiftGps, shiftImg,
seq(0.05, 1, 0.05))
write.csv(dfRegLayN, "NaipBlockSizeRegLayout.csv")

#Landsat
#Changing shift back to Landsat
blockWidth <- seq(5, 35, 5)
shiftImg <- 48/30 #*rmseImage #95% of the data (pixels) Landsat
shiftGps <- 7/30 #*rmseGps #95% of the data (pixels)
dfRegLayL <- SimSampling(LandsatImgPaths, sampSize, blockWidth, shiftGps, shiftImg,
seq(0.05, 1, 0.05))
write.csv(dfRegLayL, "LandsatBlockSizeRegLayout.csv")

#diff between total R2
lsLayout <- read.csv("LandsatBlockSizeRegLayout.csv")
lsLayout$DR2 <- compareValues(lsLayout, "R2", 40)
lsLayout$DRMSE <- compareValues(lsLayout, "RMSE", 40)

```

```
write.csv(lsLayout, "LandsatBlockSizeRegLayout.csv")

naipLayout <- read.csv("NaipBlockSizeRegLayout.csv")
naipLayout$DR2 <- compareValues(naipLayout, "R2", 40)
naipLayout$DRMSE <- compareValues(naipLayout, "RMSE",40)
write.csv(naipLayout, "NaipBlockSizeRegLayout.csv")
```

## Chapter 3

#Impact of sample design and modeling technique

#John Hogland 12/3/2018

```
library(raster)
library(parallel)
library(mgcv)
library(randomForest)
library(nnet)
library(kernlab)
library(ggplot2)
library(grid)
library(gridExtra)

sampGam <- function(rlst, naipC, ss, iter = 10, stype = 'srs', mask = NULL) {
  rs <- stack(rlst)
  crs(rs) <- crs(naipC)
  rsStack <- stack(rs, naipC)
  outDf <- data.frame(ids = integer(), dtrel = character(), design = character(),
mdl = character(), stat = double(), value = double(), samplesize = integer())
  itCnt <- 1
  for (j in 1:iter) {
    print(paste("iteration", itCnt))
    srs <- sampRaster(rsStack, ss, stype, prs = naipC, mask = mask)
    ndf <- na.omit(srs@data)
    fldNm <- names(ndf)
    nmSplit <- length(fldNm) - 3
    respNm <- fldNm[1:nmSplit]
    predNm <- fldNm[(nmSplit + 1):length(fldNm)]
    predStr <- paste(predNm, collapse = " + ")
    for (i in 1:length(respNm)) {
      ors <- subset(rs, i)
      tt <- cellStats(ors, sum) #true total
      #modeled estimates (t_dif, rmse)
      frm2 <- formula(paste(respNm[i], '~', paste('s(', predNm, ')', collapse
= "+"), sep = ""))
      g_vls <- getGam(frm2, ndf, naipC, tt, ors) #general additive model
      outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'gam', stat = 'total', value = g_vls[1], samplesize = ss))
      outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'gam', stat = 'rmse', value = g_vls[2], samplesize = ss))
    }
  }
}
```

```

        itCnt <- itCnt + 1
    }
    removeTmpFiles(h = 0)
    return(outDf)
}

sampSim <- function(rlst, naip, smp) {
    #rlst = rlstNm
    #naip = rsB
    #smp =sampleLst

    rs <- stack(rlst)
    crs(rs) <- crs(naip)
    rsStack <- stack(rs, naip)
    outDf <- data.frame(ids = integer(), dtrel = character(), design = character(),
mdl = character(), stat = double(), value = double(), samplesize = integer())
    itCnt <- 1
    scnt = 1
    for (j in 1:length(smp)) {
        print(paste("iteration", itCnt))
        stype = "srs"
        ss = nrow(smp[[j]])
        if (scnt == 2) stype = "sys"
        if (scnt == 3) stype = "grts"
        if (scnt == 4) stype = "rsnr"
        scnt = scnt + 1
        if(scnt>4) scnt = 1
        srs <- cbind(as.data.frame(extract(rs, smp[[j]])), smp[[j]]@data)
        ndf <- na.omit(srs)
        fldNm <- names(ndf)
        nmSplit <- length(fldNm) - 4
        respNm <- fldNm[1:nmSplit]
        predNm <- fldNm[(nmSplit + 1):length(fldNm)]
        predStr <- paste(predNm, collapse = " + ")
        for (i in 1:length(respNm)) {
            ors <- subset(rs, i)
            tt <- cellStats(ors, sum) #true total
            d_m <- mean(ndf[[i]]) #design mean
            d_t <- d_m * ncell(ors) #design total
            d_dif <- tt - d_t #design total difference
            d_rmse <- sqrt(cellStats(((ors - d_m) ^ 2), mean)) #design rmse
difference

```

```

        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'design', stat = 'total', value = d_dif, samplesize = ss))
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'design', stat = 'rmse', value = d_rmse, samplesize = ss))
        #modeled estimates (t_dif, rmse)
        frm <- formula(paste(respNm[i], '~', predStr, sep = ""))
        l_vls <- getLinear(frm, ndf, naip, tt, ors) #linear
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'linear', stat = 'total', value = l_vls[1], samplesize = ss))
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'linear', stat = 'rmse', value = l_vls[2], samplesize = ss))
        frm2 <- formula(paste(respNm[i], '~', paste('s(', predNm, ')', collapse
= "+"), sep = ""))
        g_vls <- getGam(frm2, ndf, naip, tt, ors) #general additive model
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'gam', stat = 'total', value = g_vls[1], samplesize = ss))
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'gam', stat = 'rmse', value = g_vls[2], samplesize = ss))
        s_vls <- getSvm(frm, ndf, naip, tt, ors) #support vector machine
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'svm', stat = 'total', value = s_vls[1], samplesize = ss))
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'svm', stat = 'rmse', value = s_vls[2], samplesize = ss))
        n_vls <- getNn(frm, ndf, naip, tt, ors) #nueral networks
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'nn', stat = 'total', value = n_vls[1], samplesize = ss))
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'nn', stat = 'rmse', value = n_vls[2], samplesize = ss))
        r_vls <- getRf(frm, ndf, naip, tt, ors) #random forest
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'rf', stat = 'total', value = r_vls[1], samplesize = ss))
        outDf <- rbind(outDf, data.frame(ids = itCnt, dtrel = respNm[i], design
= stype, mdl = 'rf', stat = 'rmse', value = r_vls[2], samplesize = ss))
    }
    itCnt <- itCnt + 1
}
removeTmpFiles(h = 0)
return(outDf)
}
getLinear <- function(frm, df, naip, total, ors) {
  t_md1 <- lm(frm, data = df) #linear model
  prs <- predict(naip, t_md1) #predicted raster surface
  m_t <- cellStats(prs, sum) #modeled total

```

```

    t_dif <- total - m_t #modeled total difference
    m_rmse <- sqrt(cellStats(((ors - prs) ^ 2), mean)) #residuals
    return(c(t_dif, m_rmse))
}
getGam <- function(frm, df, naip, total, ors) {
  t_md1 <- gam(frm, data = df) #gam
  prs <- predict(naip, t_md1) #predicted raster surface
  m_t <- cellStats(prs, sum) #modeled total
  t_dif <- total - m_t #modeled total difference
  m_rmse <- sqrt(cellStats(((ors - prs) ^ 2), mean)) #residuals
  return(c(t_dif, m_rmse))
}
getSvm <- function(frm, df, naip, total, ors) {
  t_md1 <- ksvm(frm, data = df, kernel = 'rbfdot') #svm model
  prs <- predict(naip, t_md1) #predicted raster surface
  m_t <- cellStats(prs, sum) #modeled total
  t_dif <- total - m_t #modeled total difference
  m_rmse <- sqrt(cellStats(((ors - prs) ^ 2), mean)) #residuals
  return(c(t_dif, m_rmse))
}
getNn <- function(frm, df, naip, total, ors) {
  s <- nrow(df) / 2
  d <- s * 0.1
  t_md1 <- nnet(frm, data = df, size = s, decay = d, linout = T) #nnet model
  prs <- predict(naip, t_md1) #predicted raster surface
  m_t <- cellStats(prs, sum) #modeled total
  t_dif <- total - m_t #modeled total difference
  m_rmse <- sqrt(cellStats(((ors - prs) ^ 2), mean)) #residuals
  return(c(t_dif, m_rmse))
}
getRf <- function(frm, df, naip, total, ors) {
  t_md1 <- randomForest(frm, data = df, ntree = 20, mtry = 1) #random forest
  prs <- predict(naip, t_md1) #predicted raster surface
  m_t <- cellStats(prs, sum) #modeled total
  t_dif <- total - m_t #modeled total difference
  m_rmse <- sqrt(cellStats(((ors - prs) ^ 2), mean)) #residuals
  return(c(t_dif, m_rmse))
}

predict.getClosestPoint <- function(centers, newdata, iCvar, maxn=10000) {
  #newdata = rsDf
  #cvar = cvar
  #centers = centers

```

```

#maxn = 10000

n = nrow(newdata)
nc = nrow(centers)
if (n > maxn) {
  n2 = nc * 1000
  if (n2 > maxn) {
    n2 = maxn
  }
  newdata[sample(n, n2),]
  n=n2
}
lblv = vector(mode = "integer", length = n)
d2v = vector(mode = "numeric",length = n)
for (i in 1:n) {
  #i=1
  x = as.matrix(newdata[i,])
  if (sum(is.na(x)) > 0) next
  lbl = -1
  sdist = .Machine$double.xmax
  for (j in 1:nc) {
    #j=1
    c = as.matrix(centers[j,])
    if(sum(is.na(c)) > 0) next
    xdif = x - c
    d2 = (xdif%*%iCvar%*%t(xdif))[1]
    #d2 = mahalanobis(x,c, cvar)
    if (d2 < sdist) {
      lbl = j
      sdist = d2
    }
  }
  lblv[i] = lbl
  d2v[i] = sdist
}
return(cbind(lblv,d2v))
}

createClosestPointRaster <- function(Pred, centers, cvar) {
  beginCluster(detectCores()-1)
  closestRs = clusterR(Pred, predict, args = list(centers = centers, cvar = cvar,
fun = predict.getClosestPoint, index = 1:2), progress = 'text')
  endCluster()
}

```



```

    return(closestRs)
}

getBstat = function(lbls,sampleSize) {
  tb = as.data.frame(table(lbls))[, 2]
  totaln = sum(tb)
  equalSize = totaln/sampleSize
  tbCnt = length(tb)
  vi = tb/equalSize
  dif = sampleSize - tbCnt
  b = (sum((vi-1)^2) + dif*1) / sampleSize
  return(b)
}

grtsSampleDataFrame<- function(df, sampleSize) {
  tdf = df
  pca <- princomp(tdf, cor = TRUE)
  tdf$uid <- 1:nrow(tdf)
  sc <- pca$scores
  tdf$Comp.1 = sc[, 1]
  tdf$Comp.2 = sc[,2]
  Equaldsgn <- list(None = list(panel = c(PanelOne = sampleSize), seltype =
"Equal"))
  outv1 <- grts(design = Equaldsgn, type.frame = "finite", src.frame =
'att.frame', att.frame = tdf, xcoord = "Comp.1", ycoord = "Comp.2", shapefile = F)
  outdf <- outv1@data #merge(outv1@data, , by = c('Comp.1', 'Comp.2'))
  return(outdf)
}

wsPath <-
"C:\\Users\\jshogland\\Documents\\John\\projects\\UMGradSchool\\Project\\Papers\\Dis
sertation\\chapter2\\data"
setwd(wsPath)
source("C:\\Users\\jshogland\\Documents\\John\\projects\\UMGradSchool\\Project\\Pape
rs\\Dissertation\\Rscripts\\Dissertation\\Dissertation\\jhLib.R")
rasterOptions(tmpdir = paste(getwd(), "tmp", sep = "/"))
tmpDir(create = TRUE)
roadsPath = roadsPath <-
"C:\\Users\\jshogland\\Documents\\John\\projects\\UMGradSchool\\Classes\\SamplingMet
hods\\FinalProject\\roads.shp"

#raster
rsB <- brick("test.tif")

```

```

#roads
flNm <- strsplit(basename(roadsPath), ".shp")[[1]]
dirPath <- dirname(roadsPath)
roadLayer <- readOGR(dsn = dirPath, layer = flNm)
crs(rsB)=crs(roadLayer)
rdBuff <- gBuffer(roadLayer, width = 50)
rsm <- mask(rsB, rdBuff)

covValue = layerStats(rsB, 'cov', na.rm = TRUE)
cvar = covValue$covariance
iCvar = solve(cvar)
ss = 50

#test B stat
sys = sampleRegular(rsB, 50, sp = T) #sampRaster(rsB, 45, 'sys')
ndf = data.frame(x = runif(10000, xmin(rsB), xmax(rsB)), y = runif(10000, ymin(rsB),
ymax(rsB)))
lbl = predict.getClosestPoint(as.data.frame(sys@coords), ndf, matrix(c(1, 0, 0, 1),
nrow = 2, ncol = 2))
Btest = getBstat(lbl[, 1], nrow(sys))

#get sample locations, B, and D2 statistics
niter = 100
valueMatrix = matrix(nrow = niter * 4, ncol = 3) #sType, B, D2
sampleLst = list(length=niter*4)
rCnt = 1
for (i in 1:niter) {
  print(paste("Getting sample ",as.character(i),sep = ""))
  srs = sampRaster(rsB, ss, 'srs') #rsDf[sample(1:nrow(rsDf), ss),]
  sampleLst[rCnt]=srs
  sys = sampRaster(rsB, ss, 'sys')
  sampleLst[rCnt+1] = sys
  gs = sampRaster(rsB, ss, 'grts_b', prs = rsB) #grtsSampleDataFrame(rsDf,ss)
  sampleLst[rCnt+2] = gs
  nr = sampRaster(rsm, ss, 'srs')
  sampleLst[rCnt+3] = nr

  rsSubset = sampRaster(rsB, 10000)
  rsDf = rsSubset@data
  for (j in 1:4) {
    cnt = srs@data
    if (j == 2) cnt = sys@data
  }
}

```

```

    if (j == 3) cnt = gs@data
    if (j == 4) cnt = nr@data
    lbls = predict.getClosestPoint(centers = cnt, newdata = rsDf, iCvar = iCvar,
maxn = 10000)
    valueMatrix[rCnt,] = c(j,getBstat(lbls[, 1], ss), mean(lbls[, 2]))
    rCnt = rCnt + 1
  }
}

saveRDS(valueMatrix, "BS_D2_stats_100.vls")
saveRDS(sampleLst, "samples_50.smp")

#histograms
valueMatrix = readRDS("BS_D2_stats_50.vls")
par(mfrow = c(4, 2))
df = as.data.frame(valueMatrix)
names(df)=c("stype", "BS", "D2") #srs, sys, gs, nr
for (i in 1:4) {
  tdf = df[which(df$stype == i),]
  print(summary(tdf))
  print(paste(sd(tdf$BS), " --- ", sd(tdf$D2), sep = ""))
  hist(tdf$BS)
  hist(tdf$D2)
}

#sample size for spread statistic
cnt = 10000
mv1 = 99.48
pmv1 = 0.001
sdv1 = 1.30
ss = 10
alpha = 0.05
(tse = sqrt(((cnt - ss) / cnt) * sdv1 ^ 2 / ss))
(n = qt((1 - (alpha / 2)), ss) * tse ^ 2 / (pmv1 * mv1))

#simulations
sampleLst = readRDS("samples_50_100000.smp")
rlstNm <- c('linearL0.tif', 'linearL1.tif', 'linearL2.tif', 'linearL3.tif',
'squaredL0.tif', 'squaredL1.tif', 'squaredL2.tif', 'squaredL3.tif',
'nonlinearL0.tif', 'nonlinearL1.tif', 'nonlinearL2.tif', 'nonlinearL3.tif')
outDf = sampSim(rlstNm, rsB, sampleLst)
saveRDS(outDf, "outDf.sim")

```

```

outDf = readRDS("outDf_100000.sim")
rmseDesign = subset(outDf, stat == "total")
sumTblM = aggregate(rmseDesign, by = list(rmseDesign$dtrel, rmseDesign$design,
rmseDesign$mdl), FUN = "mean")
sumTblS = aggregate(rmseDesign, by = list(rmseDesign$dtrel, rmseDesign$design,
rmseDesign$mdl), FUN = "sd")

#####
#Figures
#####
buildDf <- function(smpLst, rspNmVec, ndf=NULL, subsetPred=F) {
  outDf <- data.frame()
  useInput = F
  if (is.null(ndf)) {
    useInput = T
  }
  for (smp in smpLst) {
    #smp = smpLst[[1]]
    smpdf = smp@data
    for (rsp in rspNmVec) {
      #rsp = rspNmVec[1]
      rspRs = raster(rsp)
      rsNm = names(rspRs)
      crs(rspRs) = crs(smp)
      train <- cbind(smpdf, resp = extract(rspRs, smp@coords))
      train.df <- na.omit(train)
      if (useInput) {
        ndf = train.df
      }
      n = nrow(ndf)
      fldNm <- names(train.df)
      resp <- fldNm[length(fldNm)]
      pred <- fldNm[1:(length(fldNm) - 1)]
      if (subsetPred) pred = fldNm[1:(length(fldNm) - 2)]
      (frm <- formula(paste(resp, "~", paste(pred, collapse = "+"), sep =
"")))

      #mean
      meanVl <- mean(train.df[match(resp, fldNm)][[1]])
      tdf <- ndf
      tdf$sc1 = rsNm
      tdf$pred = rep(meanVl, n)
      tdf$mdl = rep("design", n) #data.frame(value = tvl, pred = rep(meanVl,
n), test.4 = train.df$test.4, test.3=train.df$test.3, mdl = rep("design", n))

```

```

outDf <- rbind(outDf, tdf)
#gam
frm2 <- formula(paste(resp, '~', paste('s(', pred, ')', collapse = "+"),
sep = ""))
md1 <- gam(frm2, data = train.df)
prd1 <- predict(md1, newdata = ndf)
tdf <- ndf
tdf$sc1 = rsNm
tdf$pred = prd1
tdf$mdl = rep("gam", n) # data.frame(value = tv1, pred = prd1, test.4 =
b4, test.3 = b3, mdl = rep("gam", length(prd1)))
outDf <- rbind(outDf, tdf)
#linear
md1 <- lm(frm, data = train.df)
prd1 <- predict(md1, newdata = ndf)
tdf <- ndf
tdf$sc1 = rsNm
tdf$pred = prd1
tdf$mdl = rep("linear", n) #data.frame(value = tv1, pred = prd1, test.4
= b4, test.3 = b3, mdl = rep("linear", length(prd1)))
outDf <- rbind(outDf, tdf)
#n-net
s <- nrow(train.df) / 2
d <- s * 0.1
md1 <- nnet(frm, data = train.df, size = s, decay = d, linout = T)
prd1 <- predict(md1, newdata = ndf)
tdf <- ndf
tdf$sc1 = rsNm
tdf$pred = prd1
tdf$mdl = rep("nn", n) # data.frame(value = tv1, pred = prd1, test.4 =
b4, test.3 = b3, mdl = rep("nn", length(prd1)))
outDf <- rbind(outDf, tdf)
#rf
md1 <- randomForest(frm, data = train.df, ntree = 20, mtry = 1)
prd1 <- predict(md1, newdata = ndf)
tdf <- ndf
tdf$sc1 = rsNm
tdf$pred = prd1
tdf$mdl = rep("rf", n) # data.frame(value = tv1, pred = prd1, test.4 =
b4, test.3 = b3, mdl = rep("rf", length(prd1)))
outDf <- rbind(outDf, tdf)
#svm
md1 <- ksvm(frm, data = train.df, kernel = 'rbfdot')

```

```

        prd1 <- predict(md1, newdata = ndf)
        tdf <- ndf
        tdf$sc1 = rsNm
        tdf$pred = prd1
        tdf$mdl = rep("svm", n) # data.frame(value = tv1, pred = prd1, test.4 =
b4, test.3 = b3, mdl = rep("svm", length(prd1)))
        outDf <- rbind(outDf, tdf)
    }
}
return(outDf)
}

library(raster)
library(mgcv)
library(randomForest)
library(nnet)
library(kernlab)
library(ggplot2)
library(grid)
library(gridExtra)
library(ggExtra)

wsPath <-
"C:\\Users\\jshogland\\Documents\\John\\projects\\UMGradSchool\\Project\\Papers\\Dis
sertation\\chapter2\\data"
setwd(wsPath)
rasterOptions(tmpdir = paste(getwd(), "tmp", sep = "/"))
tmpDir(create = TRUE)

valueMatrix = readRDS("BS_D2_stats_50_100000.vls")
sampleLst = readRDS("samples_50_100000.smp")
outDf = readRDS("outDf_100000.sim")

#Figure 5
#Balance in feature and geographic space
cbPalette <- c("black", "blue", "green", "red")
rsB <- brick("test.tif")
rsStats = cellStats(rsB, mean)
ext = extent(rsB)
xc = (xmax(ext) - xmin(ext))/2
yc = (ymax(ext) - ymin(ext))/2
ftrDf = data.frame()
cnt = 1

```

```

for (s in sampleLst) {
  tdf = s@data
  crd = s@coords
  tdf$x = (crd[, 1]-xmin(ext))
  tdf$y = (crd[, 2]-ymin(ext))
  vls = as.data.frame(t(apply(tdf, 2, mean)))
  txt = "SRS"
  if (cnt == 2) txt = "SYS"
  if (cnt == 3) txt = "GRTS"
  if (cnt == 4) txt = "RSNR"
  vls$lbl = txt
  ftrDf = rbind(ftrDf, vls)
  cnt = cnt + 1
  if(cnt>4) cnt = 1
}
png("fig_Bias_50_naip.png", width = 800, height = 300, res = 100)
p1 = ggplot(data = ftrDf, aes(x = test.4, y = test.1, col = lbl)) +
  geom_point() +
  geom_hline(yintercept = rsStats[1], col = "darkgrey", linetype = "dashed") +
  geom_vline(xintercept = rsStats[4], col = "darkgrey", linetype = "dashed") +
  theme_bw() +
  theme(legend.position = "none", axis.title.y = element_blank(), axis.title.x =
element_blank(), axis.text = element_text(size = 15), axis.title = element_text(size
= 20)) +
  scale_colour_manual(values = cbPalette)

p2 = ggplot(data = ftrDf, aes(x = x, y = y, col = lbl)) +
  geom_point() +
  geom_hline(yintercept = yc, col = "darkgrey", linetype = "dashed") +
  geom_vline(xintercept = xc, col = "darkgrey", linetype = "dashed") +
  theme_bw() +
  theme(legend.position = "none",axis.title.y = element_blank(), axis.title.x =
element_blank(), axis.text = element_text(size = 15), axis.title = element_text(size
= 20)) +
  scale_colour_manual(values = cbPalette)

grid.arrange(p1, p2, nrow = 1)
dev.off()
plot.new()

#Figure 6
#Average model trend for linear, quadratic, and nonlinear SC1 surface from 100
iterations (SRS; n=50; error = 40%)

```

```

rsB <- brick("test.tif")
rsStats = cellStats(rsB, mean)
rsB4Min = minValue(rsB)[4]
rsB4Max = maxValue(rsB)[4]
rsB4 = seq(rsB4Min, rsB4Max, 1)
ndf = data.frame(test.1=rsStats[1],test.2=rsStats[2],test.3=rsStats[3],test.4=rsB4)
rspNms <- c("linearL1.tif", "squaredL1.tif", "nonLinearL1.tif")
#SRS
sdf <- buildDf(sampleLst[seq(1, length(sampleLst), 4)], rspNms, ndf)

cbPalette <- c("#000000", "#E69F00", "#999999", "#009E73", "red", "#56B4E9")

#image
png("fig_Average_50_model_normal.png", width = 1200, height = 300, res = 100)

#nonlinear
sdfSub = subset(sdf,sc1=="nonLinearL1")
tv1 = apply(sdfSub[,1:4],1,sum) / 4
tv12 = sdfSub$test.4 > 170 & sdfSub$test.4 < 210
tv1 = tv1 * tv12
sdfSub$value = tv1
p3 <- ggplot(sdfSub, aes(x = test.4, y = pred, col = mdl)) +
  theme_bw() +
  geom_line(aes(x = test.4, y = value), col = "blue", size = 0.75) +
  geom_smooth(aes(x = test.4, y = pred), level = 0.99, size = 1, linetype =
"dotdash") +
  theme( legend.position="none",legend.title = element_text(size = 17),
axis.title.y = element_blank(), axis.title.x = element_blank(), axis.text =
element_text(size = 15), axis.title = element_text(size = 20)) +
  labs(title = "Nonlinear", cex.main = 2.2) +
  scale_colour_manual(values = cbPalette) +
  coord_cartesian(ylim = c(-50, 170))

#linear
sdfSub = subset(sdf, sc1 == "linearL1")
tv1 = apply(sdfSub[, 1:4], 1, sum) / 4
sdfSub$value = tv1
p1 <- ggplot(sdfSub, aes(x = test.4, y = pred, col = mdl)) +
  theme_bw() +
  geom_line(aes(x = test.4, y = value), col = "blue", size = 0.75) +
  geom_smooth(aes(x = test.4, y = pred), level = 0.99, size = 1, linetype =
"dotdash") +

```



```

    theme(legend.position = "none", legend.title = element_text(size = 17),
axis.title.y = element_blank(), axis.title.x = element_blank(), axis.text =
element_text(size = 15), axis.title = element_text(size = 20)) +
    labs(title = "Linear", cex.main = 2.2) +
    scale_colour_manual(values = cbPalette) +
    coord_cartesian(ylim = c(100, 170))
#squared
sdfSub = subset(sdf, sc1 == "squaredL1")
tv1 = apply(sdfSub[, 1:4], 1, sum) / 4
sdfSub$value = tv1^2
p2 <- ggplot(sdfSub, aes(x = test.4, y = pred, col = mdl)) +
    theme_bw() +
    geom_line(aes(x = test.4, y = value), col = "blue", size = 0.75) +
    geom_smooth(aes(x = test.4, y = pred), level = 0.99, size = 1, linetype =
"dotdash") +
    theme(legend.position = "none", legend.title = element_text(size = 17),
axis.title.y = element_blank(), axis.title.x = element_blank(), axis.text =
element_text(size = 15), axis.title = element_text(size = 20)) +
    labs(title = "Squared", cex.main = 2.2) +
    scale_colour_manual(values = cbPalette)

grid.arrange(p1,p2,p3,nrow=1)
dev.off()
plot.new()

#Figure 7 & 8
pc <- c("20%", "40%", "60%", "80%")
rnm <- c("linear", "nonlinear", "squared")
totalVls <- c()
sdVls <- c()
for (nm in rnm) {
    rs <- raster(paste(nm, ".tif", sep = ""))
    tv <- cellStats(rs, stat = 'sum')
    totalVls <- c(totalVls, tv)
    sv <- cellStats(rs, stat = 'sd')
    sdVls <- c(sdVls, sv)
}

n = 50
dfcmb <- outDf
t <- dfcmb$dtrel
tlen <- nchar(as.character(t))
t1 <- substr(t, 1, tlen - 2)

```

```

t2 <- substr(t, tlen, tlen)
dfcmb$error <- t2
dfcmb$dtrel <- t1

dtypes <- as.factor(unique(dfcmb$design))
rtypes <- as.factor(unique(dfcmb$dtrel))
stypes <- as.factor(unique(dfcmb$stat))
etypes <- as.factor(unique(dfcmb$error))

cbPalette <- c("#000000", "#999999", "#E69F00", "#56B4E9", "#009E73", "red")
for (s in levels(stypes)) {
  for (r in levels(rtypes)) {
    #make picture
    png(paste("fig_100_", as.character(n), "_", s, "_", r, ".png", sep = ""),
width = 1200, height = 1200, res = 100)
    gv <- vector("list", 16)
    gCnt <- 1
    tv1 <- subset(dfcmb, stat == s & dtrel == r)$value
    mav1 = sdVls[match(r, rnm)]
    if (s == "total") mav1 = totalVls[match(r, rnm)]
    lvl <- min(tv1) / mav1 * 100
    uv1 <- max(tv1) / mav1 * 100
    if (s == "rmse") uv1 = 150
    e = "L"
    eNm = "NORMAL"
    for (p in rev(1:length(pc))) {
      for (d in levels(dtypes)) {
        dNm <- toupper(d)
        tdf <- subset(dfcmb, stat == s & dtrel == r & design == d & error ==
as.character(p-1))
        tdf$vl2 <- tdf$value / mav1 * 100
        if (gCnt == 1 | gCnt == 5 | gCnt == 9) {
          pl <- ggplot(tdf, aes(x = mdl, y = vl2, col = mdl)) +
geom_boxplot() + theme_light() + theme(axis.title.x = element_blank(), axis.title =
element_text(size = 17), axis.text = element_text(size = 15), axis.text.x =
element_blank(), axis.ticks.x = element_blank(), axis.title.y = element_blank()) +
scale_colour_manual(values = cbPalette) + coord_cartesian(ylim = c(lvl, uv1))
        }
        if (gCnt == 13) {
          pl <- ggplot(tdf, aes(x = mdl, y = vl2, col = mdl)) +
geom_boxplot() + theme_light() + theme(legend.position = "none", axis.title =
element_text(size = 17), axis.text = element_text(size = 15), axis.text.x =
element_blank(), axis.ticks.x = element_blank(), axis.title.y = element_blank()) +

```

```

xlab(dNm) + scale_colour_manual(values = cbPalette) + coord_cartesian(ylim = c(lvl,
uvl))
    }
    if (gCnt == 14 | gCnt == 15 | gCnt == 16) {
        pl <- ggplot(tdf, aes(x = mdl, y = vl2, col = mdl)) +
geom_boxplot() + theme_light() + theme(legend.position = "none", axis.title =
element_text(size = 17), axis.text = element_text(size = 15), axis.text.x =
element_blank(), axis.text.y = element_blank(), axis.ticks.x = element_blank(),
axis.title.y = element_blank()) + xlab(dNm) + scale_colour_manual(values =
cbPalette) + coord_cartesian(ylim = c(lvl, uvl))
    }
    if (gCnt != 1 & gCnt != 5 & gCnt != 9 & gCnt != 13 & gCnt != 14 &
gCnt != 15 & gCnt != 16) {
        pl <- ggplot(tdf, aes(x = mdl, y = vl2, col = mdl)) +
geom_boxplot() + theme_light() + theme(legend.position = "none", axis.title.x =
element_blank(), axis.text = element_text(size = 15), axis.text.x = element_blank(),
axis.text.y = element_blank(), axis.ticks.x = element_blank(), axis.title.y =
element_blank()) + scale_colour_manual(values = cbPalette) + coord_cartesian(ylim =
c(lvl, uvl))
    }
    gv[[gCnt]] <- pl
    gCnt <- gCnt + 1
}
}
grid.arrange(gv[[1]], gv[[2]], gv[[3]], gv[[4]], gv[[5]], gv[[6]], gv[[7]],
gv[[8]], gv[[9]], gv[[10]], gv[[11]], gv[[12]], gv[[13]], gv[[14]], gv[[15]],
gv[[16]], nrow = 4, ncol = 4, top = textGrob(paste(toupper(s), " - ", toupper(r), "
box plots for 100 iterations (n = ", as.character(n), ")"), sep = "")), gp =
gpar(fontsize = 20, font = 3))
    dev.off()
    plot.new()
}
}

```

#Figure 9 observed vs predicted for 40%, nonlinear, GRTS, 3 and 4 band predictions for each of the models

```

rspNms = c("NonlinearL1.tif")
resRs = raster(rspNms)
crs(resRs) = crs(smpLst[[1]])
newDf = data.frame()
smpLst = sampleLst[seq(4, length(sampleLst), 4)]
for (smp in smpLst) {
    pnts = smp[sample(1:(length(smp)), 1),]
}

```

```

    respVls = extract(resRs, pnts)
    vls = pnts@data
    vls$resp = respVls
    newDf = rbind(newDf,vls)
}
cbPalette <- c("#000000", "#E69F00", "#999999", "#009E73", "red", "#56B4E9")
newDf = na.omit(newDf)
sdf = buildDf(sampleLst[seq(4, length(sampleLst), 4)], rspNms, ndf=newDf)
sdf2 = buildDf(sampleLst[seq(4, length(sampleLst), 4)], rspNms, ndf = newDf,
subsetPred = T)
p1 <- ggplot(sdf, aes(x = resp, y = pred, col = mdl)) +
    theme_bw() +
    geom_smooth(size = 1, linetype = "solid", se = F) +
    geom_abline(slope = 1, intercept = 0, color = "gray", size = 0.5,
linetype="dashed") +
    theme(legend.position="none", legend.title = element_blank(), axis.text
= element_text(size = 15), axis.title = element_text(size = 17), axis.title.x =
element_blank(), axis.title.y=element_blank()) +
    scale_colour_manual(values = cbPalette) +
    coord_cartesian(ylim = c(0, 150), xlim = c(0, 150))

p2 <- ggplot(sdf2, aes(x = resp, y = pred, col = mdl)) +
    theme_bw() +
    geom_smooth(size = 1, linetype = "solid", se = F) +
    geom_abline(slope = 1, intercept = 0, color = "gray", size = 0.5,
linetype = "dashed") +
    theme(legend.position = "none", legend.title = element_blank(),
axis.text.y = element_blank(), axis.ticks.y = element_blank(), axis.text =
element_text(size = 15), axis.title = element_text(size = 17), axis.title.x =
element_blank(), axis.title.y = element_blank()) +
    scale_colour_manual(values = cbPalette) +
    coord_cartesian(ylim = c(0, 150), xlim = c(0, 150))

p3 <- ggplot(newDf, aes(x = resp, stat(density))) +
    theme_bw() +
    geom_density(col = "black", linetype="dashed")+
    theme(axis.ticks.x = element_blank(), axis.ticks.y = element_blank(),
panel.border = element_blank(), panel.grid.major = element_blank(), panel.grid.minor
=
element_blank(), axis.text.x=element_blank(), axis.text.y=element_blank(), axis.title.x
= element_blank(), axis.title.y = element_blank())

png("predictedV_Observed_100_GRTS.png", width = 800, height = 300, res = 100)

```

```

grid.arrange(p1, p2, nrow = 1)
dev.off()
plot.new()

#Figure 10 Spread vs RMSE by Estimator
rspNms = c("NonlinearL1.tif")

mdls = c("gam", "rf", "svm")
gv <- vector("list", 3)
gCnt=1
for (m in mdls) {
  outDfsub = subset(outDf, dtrel == "nonlinearL1" & stat == "rmse" & mdl == m )

  outDfsub$B = valueMatrix[,2]
  outDfsub$D2 = valueMatrix[, 3]
  tmd = lm(value ~ design, data = outDfsub)
  summary(tmd)
  outDfsub2 = subset(outDfsub,value>=45 & value<=85 & B> 0.1 & B < 0.6)
  cbPalette <- c("green", "red", "black", "blue")
  p = ggplot(outDfsub2, aes(x = B, y = value, col = design)) +
    geom_point() +
    theme_bw() +
    theme(legend.position = "none", legend.title = element_blank(), axis.text =
element_text(size = 15), axis.text.x = element_blank(),axis.ticks.x =
element_blank(),axis.title = element_text(size = 17),
axis.text.y=element_blank(),axis.ticks.y=element_blank(),axis.title.x =
element_blank(), axis.title.y = element_blank()) +
    scale_colour_manual(values = cbPalette) +
    coord_cartesian(xlim = c(0.1, 0.6), ylim=c(45,85))

  p1 = ggMarginal(p, outDfsub2, x = B, y = value, type = c("density"), groupColour
= T)
  gv[[gCnt]] = p1
  gCnt=gCnt+1
}

png("fig_spread_vs_rmse.png", width = 600, height = 200, res = 100)
grid.arrange(gv[[1]], gv[[2]], gv[[3]],nrow=1,ncol=3)
dev.off()
plot.new()

#Table top fitting design
mdls = c("gam", "rf", "svm", "nn","linear","design")

```

```

for (m in mdl) {
  outDfsub = subset(outDf, dtrel == "linearL2" & stat == "rmse" & mdl == m)

  outDfsub$B = valueMatrix[, 2]
  outDfsub$D2 = valueMatrix[, 3]
  tmd = lm(value ~ design, data = outDfsub)
  print(paste("Model = ",m,sep=""))
  print(summary(tmd))
}

#Table Moran's I
sc1 = c("linearL", "squaredL", "nonlinearL")
pc = c("0.tif", "1.tif", "2.tif", "3.tif")
for (sc in sc1) {
  for (p in pc) {
    nm = paste(sc, p, sep = "")
    rs = raster(nm)
    mi = Moran(rs, matrix(c(0, 1, 0, 1, 0, 1, 0, 1, 0), 3, 3))
    print(paste(nm, " = ", as.character(mi), sep = ""))
  }
}

```

## Chapter 4

#BAH and TPH estimation for ANF

#John Hogland 9/10/2019

```
library(mgcv)
library(ggplot2)
library(rgdal)
library(sp)
library(spdep)
baseDir <-
"C:\\Users\\jshogland\\Documents\\John\\projects\\UMGradSchool\\Project\\Papers\\Dis
sertation\\chapter3\\data"
setwd(baseDir)
source("C:\\Users\\jshogland\\Documents\\John\\projects\\UMGradSchool\\Project\\Pape
rs\\Dissertation\\Rscripts\\Dissertation\\Dissertation\\jhLib.R")

#' Title createEnsembledGam
#'
#' @param frm = formula
#' @param df = data frame
#' @param fam = family default = gaussian()
#' @param nmdl = number of models default = 50
#' @param ptrain = percent of data used to train the model default = 0.75
#' @param kfact = keep factor for models: used to select models that have a similar
RMSE training and testing datasets (default 1.25 * training RMSE)
#'
#' @return vector of gam models and oob and training rmse
#' @export
#'
#' @examples
createEnsembleGam <- function(frm, df, fam = gaussian(), nmdl = 50, ptrain = 0.75,
kfact = 20) {
  mdlV = list(length = nmdl)
  rmseV = vector(mode = "double", length = nmdl)
  rmseT = vector(mode = "double", length = nmdl)
  n = round(ptrain * nrow(df))
  mdlCnt = 0
  while (mdlCnt < nmdl) {
    sIndex = sample(nrow(df), n)
    tdf = df[sIndex,]
    vdf = df[-sIndex,]
    try({
      mdl = gam(frm, family = fam, data = tdf)
```

```

        pv1V = getPredictedValues(md1, vdf) #predict(md1, newdata = vdf)#add
multinom estimates
        ov1V = getResponseValues(vdf, md1) #vdf[all.vars(frm)[1]][[1]]#adjust
for multinom estimates
        t_rmseV = getErrorEstimate(pv1V, ov1V, md1) #sqrt(mean((pv1V - ov1V) ^
2))

        pv1T = getPredictedValues(md1, tdf) #predict(md1, newdata = tdf)
        ov1T = getResponseValues(tdf, md1) #tdf[all.vars(frm)[1]][[1]]
        t_rmseT = getErrorEstimate(pv1T, ov1T, md1) #sqrt(mean((pv1T - ov1T) ^
2))

        if (t_rmseV <= (t_rmseT * kfact)) {
            mdlCnt = mdlCnt + 1
            mdlV[[mdlCnt]] = md1
            rmseV[mdlCnt] = t_rmseV
            rmseT[mdlCnt] = t_rmseT
            print(paste("Found model ", mdlCnt, sep = ""))
        }
    }, silent = TRUE)
}
return(list(mdlV,rmseV,rmseT))
}
getTrainOBBAIC <- function(EGAM) {
    n = length(EGAM[[1]])
    sv1 = 0
    sv12 = 0
    for (m in 1:n) {
        md1 = EGAM[[1]][[m]]
        sv1 = sv1 + md1$aic
        sv12 = sv12+ md1$null.deviance
    }
    return(c(mean(EGAM[[3]]), mean(EGAM[[2]]), sv1 / n, sv12/n))
}
getPredictedValues <- function(md, df, t = "response") {

    pv1s = predict(md, newdata = df, type = t)
    fm = md$family
    if (fm$family == "multinom") {
        pv1s <- apply(pv1s, 1, function(x) which(max(x) == x)[1]) - 1
    }
    if (fm$family == "binomial") {

```



```

        pVls <- as.integer(pVls>0.5)
    }
    return (pVls)
}
getResponseValues <- function(df, md) {
    fm = md$family
    f = md$formula
    if (fm$family == "multinom") {
        f = f[[1]]
    }
    return(df[all.vars(f)[1]][[1]])
}
getErrorEstimate <- function(p, o, md) {
    fm = md$family
    outVl = NULL
    if (fm$family == "multinom" | fm$family == "binomial") {
        outVl = 1-sum(p==o)/length(p)
    }
    else {
        outVl = sqrt(mean((p - o) ^ 2))
    }
    return(outVl)
}
#' Transform data
#' Transforms data using pca cor
#' @param df = data frame
#' @param response = response variable
#' @param pred = vector of predictor variables
#'
#' @return list (data frame, pca
#' @export
#'
#' @examples
transformData <- function(df, response, pred) {
    frm = formula(paste("~", paste(pred, collapse = "+"), sep = ""))
    pca = princomp(frm,data=df,cor=TRUE)
    return(list(data.frame(resp=df[response][[1]],pca$scores),pca))
}
#' Predict Bagged Gam model values
#'
#' @param bGamMdl = list of models
#' @param df = new data data frame
#'

```

```

#' @return = data frame of mean predictions and standard errors
#' @export
#'
#' @examples
predictEnsembleGam <- function(bGamMdl, df, trunc = 0) {
  fm = bGamMdl[[1]]$family
  m = NULL
  s = NULL
  mdls = length(bGamMdl)
  n = nrow(df)
  if (fm$family == "multinom") {
    nlp = fm$nlp
    sm = matrix(rep(0,n*(nlp+1)),nrow = n, ncol = nlp +1)
    s2m = sm
    for (i in seq(mdls)) {
      mdl = bGamMdl[[i]]
      p = predict(mdl, df, type = "response")
      sm = sm + p
      s2m = s2m + p ^ 2
    }
    m = sm / mdls
    s = sqrt((s2m - ((sm ^ 2) / mdls)) / (mdls - 1))
    return(list(m, s))
  }
  else {
    sV = vector(mode = "double", length = n)
    s2V = vector(mode = "double", length = n)
    for (i in seq(mdls)) {
      mdl = bGamMdl[[i]]
      p = predict(mdl, df, type = "response")
      if (fm$family != "binomial") {
        p = p ^ 2
      }
      sV = sV + p
      s2V = s2V + p ^ 2
    }
    m = sV / mdls
    s = sqrt((s2V - ((sV ^ 2) / mdls)) / (mdls - 1))
    return(list(m,s))
  }
}

```

```

runSim <- function(resp, pred, df, nmdl = 10, ptrain = 0.75, kfact = 10, fam =
gaussian(), outMdl = "") {

  respV1 = df[resp][[1]]
  nlp = 1
  if (fam$family == "multinom") {
    nlp = fam$nlp
  }
  fm <- getFormula(resp, pred, nlp) #as.formula(paste(resp, " ~ ", paste("s(",
pred, ")", collapse = " + ")))
  mdls = createEnsembleGam(fm, df, nmdl = nmdl, ptrain = ptrain, kfact = kfact,
fam = fam)
  if (outMdl != "") {
    saveRDS(mdls, outMdl)
  }
  p = predictEnsembleGam(mdls[[1]], df)
  est = p[[1]]
  lest = (est - p[[2]] * 1.96)
  uest = (est + p[[2]] * 1.96)

  if (fam$family == "multinom" | fam$family == "binomial") {
    pca = princomp(formula(paste("~", paste(pred, collapse = "+"), sep = "")),
data = df, cor = TRUE)
    p1 = pca$scores[,1]
    if (fam$family == "multinom") {
      for (i in 1:ncol(est)) {
        fts = createGraph(p1, est[,i], resp, categorical = TRUE, pcol =
respV1 + 3)
      }
    }
    else {
      fts = createGraph(p1, est, resp, categorical = TRUE, pcol = respV1+3)
    }
  }
  else {
    fts = createGraph(respV1, est, resp)
  }

  #(rmse = sqrt(mean(res ^ 2)))
  #(r2 = 1 - ((mean(res ^ 2)) / var(respV1)))

```

```

    return(list(mdls[[1]], data.frame(TotRMSE = fts[2], TotR2 = fts[1] ^ 2, oobError
= mean(mdls[[2]]), trainError = mean(mdls[[3]])))
}

```

```

createGraph <- function(obs, est, graphName, axisTitleBlank = TRUE,
categorical=FALSE,pcol="blue") {
  xTitle = "Observed"
  if (categorical==TRUE) {
    xTitle = "Comp_1"
  }
  yTitle = "Predicted"
  if (axisTitleBlank == FALSE) {
    xTitle = ""
    yTitle = ""
  }

  cr = cor(est, obs)
  rmse = sqrt(mean((obs - est) ^ 2))
  sbt = paste("Correlation = ",round(cr,digits=2),": RMSE = ",
round(rmse,digits=2),sep = "")
  p4 = ggplot(data.frame(Observed = obs, Predicted = est), aes(x = Observed, y =
Predicted)) +
    theme_bw() +
    geom_smooth(size = 1, linetype = "dotdash", col = "black") +
    geom_point(col = pcol, pch = 1) +
    geom_abline(slope = 1, intercept = 0, col = "red", size = 0.5) +
    theme(legend.position = "none", plot.subtitle = element_text(size = 15),
plot.title = element_text(size = 18), axis.text = element_text(size = 15),
axis.title = element_text(size = 17)) +
    #coord_cartesian(ylim = c(0, 45), xlim = c(0, 45)) +

    xlab(xTitle) +
    ylab(yTitle) +

  labs(title = graphName,subtitle = sbt)
  print(p4)
  return(c(cr,rmse))
}

```

```

#get dataset and predictor names
plts = readOGR("PlotsShifted244.shp")
AnfDf = plts@data
knn = knearneigh(plts@coords, k = 1)

```

```

nb = knn2nb(knn)
AnfDf$DFT = 2
AnfDf[(AnfDf$Pine1BAH + AnfDf$Other1BAH) < 2, "DFT"] = 0
AnfDf[AnfDf$Other1BAH > AnfDf$Pine1BAH & AnfDf$DFT != 0, "DFT"] = 1
AnfDf$LPP = 0
AnfDf[AnfDf$LP1BAH >= 2, "LPP"] = 1
crd = plts@coords
AnfDf$X = crd[,1]
AnfDf$Y = crd[, 2]
AnfDf$XY = AnfDf$X * AnfDf$Y
(predLs <- names(AnfDf)[17:52])
(predSn <- names(AnfDf)[55:78])
(predNp <- names(AnfDf)[79:86])
(predXY <- names(AnfDf)[92:94])
(allPred <- c(predLs, predSn, predNp, predXY))
(allImagePred <- c(predLs, predSn, predNp))
tdf = transformData(AnfDf, "Pine1BAH", allImagePred)
pcaImage = tdf[[2]]
pcaDf = tdf[[1]]
(pcaPred = names(pcaDf)[2:length(names(pcaDf))])

#get dataset and predictor names for raw
AnfDfRaw = read.csv("ANFplotsRaw3.csv")
AnfDfRaw = subset(AnfDfRaw, !(OBJECTID == 189 | OBJECTID == 197 | OBJECTID == 204 |
OBJECTID == 207))
#AnfVRaw = subset(AnfDfAllRaw, sample == 0)
AnfDfRaw$DFT = AnfDf$DFT
AnfDfRaw$LPP = AnfDf$LPP
AnfDfRaw$X = AnfDf$X
AnfDfRaw$Y = AnfDf$Y
AnfDfRaw$XY = AnfDf$XY
(predLsRaw <- names(AnfDfRaw)[22:57])
(predSnRaw <- names(AnfDfRaw)[58:81])
(predNpRaw <- names(AnfDfRaw)[82:89])
(predXYRaw <- names(AnfDfRaw)[92:94])
(allPredRaw <- c(predLsRaw, predSnRaw, predNpRaw, predXYRaw))
(allImagePredRaw <- c(predLsRaw, predSnRaw, predNpRaw))
tdfRaw = transformData(AnfDfRaw, "Pine1BAH", allImagePredRaw)
pcaImageRaw = tdfRaw[[2]]
pcaDfRaw = tdfRaw[[1]]
(pcaPredRaw = names(pcaDfRaw)[2:length(names(pcaDfRaw))])

#getDFTModels

```

```

(resp = "DFT")
bmdl = getGamSigFldNames(AnfDf, resp, allImagePred, 0.05, fam = multinom(K = 2))
(pred = bmdl[[1]])
summary(bmdl[[2]])
DftStats = runSim(resp, pred, AnfDf, 50, ptrain = 0.75, kfact =
100, fam=multinom(K=2), outMdl = "Dft244.egm")
DftStats[[2]]
DftStats = readRDS("Dft244.egm")
getTrainOBBAIC(DftStats)
pDFT = predictEnsembleGam(DftStats[[1]], AnfDf)
AnfDf$r_pDFT0 = as.integer(AnfDf$DFT == 0) - pDFT[[1]][, 1]
AnfDf$r_pDFT1 = as.integer(AnfDf$DFT == 1) - pDFT[[1]][, 2]
AnfDf$r_pDFT2 = as.integer(AnfDf$DFT == 2) - pDFT[[1]][, 3]
(mt = moran.test(AnfDf$r_pDFT0, nb2listw(nb)))
(mt = moran.test(AnfDf$r_pDFT1, nb2listw(nb)))
(mt = moran.test(AnfDf$r_pDFT2, nb2listw(nb)))

(resp = "DFT")
bmdlRaw = getGamSigFldNames(AnfDfRaw, resp, allImagePredRaw, 0.05, fam = multinom(K
= 2))
(predRaw = bmdlRaw[[1]])
summary(bmdlRaw[[2]])
DftStatsRaw = runSim(resp, predRaw, AnfDfRaw, 50, ptrain = 0.75, kfact = 100,
fam=multinom(K=2), outMdl = "DftRaw244.egm")
DftStatsRaw[[2]]
DftStatsRaw = readRDS("DftRaw244.egm")
getTrainOBBAIC(DftStatsRaw)
pDFTRaw = predictEnsembleGam(DftStatsRaw[[1]], AnfDfRaw)
AnfDfRaw$r_pDFT0 = as.integer(AnfDfRaw$DFT == 0) - pDFTRaw[[1]][, 1]
AnfDfRaw$r_pDFT1 = as.integer(AnfDfRaw$DFT == 1) - pDFTRaw[[1]][, 2]
AnfDfRaw$r_pDFT2 = as.integer(AnfDfRaw$DFT == 2) - pDFTRaw[[1]][, 3]
(mt = moran.test(AnfDfRaw$r_pDFT0, nb2listw(nb)))
(mt = moran.test(AnfDfRaw$r_pDFT1, nb2listw(nb)))
(mt = moran.test(AnfDfRaw$r_pDFT2, nb2listw(nb)))

#getLPPModels
(resp = "LPP")
bmdl = getGamSigFldNames(AnfDf, resp, allImagePred, 0.05, fam = binomial())
(pred = bmdl[[1]])
summary(bmdl[[2]])
LppStats = runSim(resp, pred, AnfDf, 50, ptrain = 0.75, kfact = 100, fam =
binomial(), outMdl = "LPP244.egm")

```

```

LppStats = readRDS("LPP244.egm")
getTrainOBBAIC(LppStats)
pLPP = predictEnsembleGam(LppStats[[1]], AnfDf)[[1]]
AnfDf$r_pLPP0 = as.vector((1 - pLPP) - as.integer(AnfDf$LPP == 0))
AnfDf$r_pLPP1 = as.vector(pLPP - as.integer(AnfDf$LPP == 1))
(mt = moran.test(AnfDf$r_pLPP0, nb2listw(nb)))
(mt = moran.test(AnfDf$r_pLPP1, nb2listw(nb)))

(resp = "LPP")
bmdl = getGamSigFldNames(AnfDfRaw, resp, allImagePredRaw, 0.05, fam = binomial())
(predRaw = bmdl[[1]])
summary(bmdl[[2]])
LppStatsRaw = runSim(resp, predRaw, AnfDfRaw, 50, ptrain = 0.75, kfact = 10,
fam=binomial(), outMdl = "LppRaw244.egm")
LppStatsRaw = readRDS("LppRaw244.egm")
getTrainOBBAIC(LppStatsRaw)
pLPPRaw = predictEnsembleGam(LppStatsRaw[[1]], AnfDfRaw)[[1]]
AnfDfRaw$r_pLPP0 = as.vector((1 - pLPPRaw) - as.integer(AnfDfRaw$LPP == 0))
AnfDfRaw$r_pLPP1 = as.vector(pLPPRaw - as.integer(AnfDfRaw$LPP == 1))
(mt = moran.test(AnfDfRaw$r_pLPP0, nb2listw(nb)))
(mt = moran.test(AnfDfRaw$r_pLPP1, nb2listw(nb)))

#getPineModels
#PineBAH
AnfDf$sqrt_Pine1BAH = sqrt(AnfDf$Pine1BAH)
(resp = "sqrt_Pine1BAH")
bmdl = getGamSigFldNames(AnfDf, resp, allImagePred, 0.05, fam = gaussian())
(pred = bmdl[[1]])
summary(bmdl[[2]])
PineBahStats = runSim(resp, pred, AnfDf, 50, ptrain = 0.75, kfact = 10, outMdl =
"PineBah244_n.egm")
PineBahStats = readRDS("PineBah244_n.egm")
getTrainOBBAIC(PineBahStats)
pBAH = predictEnsembleGam(PineBahStats[[1]], AnfDf)
AnfDf$r_pBAH = as.vector(pBAH[[1]]^2 - AnfDf$Pine1BAH)
(mt = moran.test(AnfDf$r_pBAH, nb2listw(nb)))

AnfDfRaw$sqrt_Pine1BAH = sqrt(AnfDfRaw$Pine1BAH)
(resp = "sqrt_Pine1BAH")
predRaw = getGamSigFldNames(AnfDfRaw, resp, allImagePredRaw, 0.05, fam = gaussian())
PineBahStatsRaw = runSim(resp, predRaw[[1]], AnfDfRaw, 50, ptrain = 0.75, kfact =
10, outMdl = "PineBah244Raw_n.egm")

```

```

PineBahStatsRaw = readRDS("PineBah244Raw_n.egm")
getTrainOBBAIC(PineBahStatsRaw)
pBAHraw = predictEnsembleGam(PineBahStatsRaw[[1]], AnfDfRaw)
AnfDfRaw$r_pBAH = as.vector(pBAHraw[[1]] ^ 2 - AnfDfRaw$Pine1BAH)
(mt = moran.test(AnfDfRaw$r_pBAH, nb2listw(nb)))

#PineTPH
AnfDf$sqrt_Pine1TPH = sqrt(AnfDf$Pine1TPH)
(resp = "sqrt_Pine1TPH")
predTph = getGamSigFldNames(AnfDf, resp, allImagePred, 0.05, fam = gaussian())
PineTphStats = runSim(resp, predTph[[1]], AnfDf, 50, ptrain = 0.75, kfact = 10,
outMdl = "PineTph244_n.egm")
PineTphStats = readRDS("PineTPH244.egm")
getTrainOBBAIC(PineTphStats)
pTPH = predictEnsembleGam(PineTphStats[[1]], AnfDf)
AnfDf$r_pTPH = as.vector(pTPH[[1]] ^ 2 - AnfDf$Pine1TPH)
(mt = moran.test(AnfDf$r_pTPH, nb2listw(nb)))

AnfDfRaw$sqrt_Pine1TPH = sqrt(AnfDfRaw$Pine1TPH)
(resp = "sqrt_Pine1TPH")
predRaw = getGamSigFldNames(AnfDfRaw, resp, allImagePredRaw, 0.05, fam = gaussian())
PineTphStatsRaw = runSim(resp, predRaw[[1]], AnfDfRaw, 50, ptrain = 0.75, kfact =
10, outMdl = "PineTph244Raw_n.egm")
PineTphStatsRaw = readRDS("PineTPH244Raw_n.egm")
getTrainOBBAIC(PineTphStatsRaw)
pTPHraw = predictEnsembleGam(PineTphStatsRaw[[1]], AnfDfRaw)
AnfDfRaw$r_pTPH = as.vector(pTPHraw[[1]] ^ 2 - AnfDfRaw$Pine1TPH)
(mt = moran.test(AnfDfRaw$r_pTPH, nb2listw(nb)))

#getOtherModels
#OtherBAH
AnfDf$sqrt_Other1BAH = sqrt(AnfDf$Other1BAH)
(resp = "sqrt_Other1BAH")
pred = getGamSigFldNames(AnfDf, resp, allImagePred, 0.05, fam = gaussian())
OtherBahStats = runSim(resp, pred[[1]], AnfDf, 50, ptrain = 0.75, kfact = 10, outMdl
= "OtherBah244_n.egm")
OtherBahStats = readRDS("OtherBah244_n.egm")
getTrainOBBAIC(OtherBahStats)
oBAH = predictEnsembleGam(OtherBahStats[[1]], AnfDf)
AnfDf$r_oBAH = as.vector(oBAH[[1]] ^ 2 - AnfDf$Other1BAH)
(mt = moran.test(AnfDf$r_oBAH, nb2listw(nb)))

```



```

AnfDfRaw$sqrt_Other1BAH = sqrt(AnfDfRaw$Other1BAH)
(resp = "sqrt_Other1BAH")
predRaw = getGamSigFldNames(AnfDfRaw, resp, allImagePredRaw, 0.05, fam =
gaussian()[[1]])
OtherBahStatsRaw = runSim(resp, predRaw, AnfDfRaw, 50, ptrain = 0.75, kfact = 10,
outMdl = "OtherBahRaw244_n.egm")
OtherBahStatsRaw = readRDS("OtherBahRaw244_n.egm")
getTrainOBBAIC(OtherBahStatsRaw)
oBAHraw = predictEnsembleGam(OtherBahStatsRaw[[1]], AnfDfRaw)
AnfDfRaw$r_oBAH = as.vector(oBAHraw[[1]] ^ 2 - AnfDfRaw$Other1BAH)
(mt = moran.test(AnfDfRaw$r_oBAH, nb2listw(nb)))

#OtherTPH
AnfDf$sqrt_Other1TPH = sqrt(AnfDf$Other1TPH)
(resp = "sqrt_Other1TPH")
pred = getGamSigFldNames(AnfDf, resp, allImagePred, 0.05, fam = gaussian()[[1]])
OtherTphStats = runSim(resp, pred, AnfDf, 50, ptrain = 0.75, kfact = 10, outMdl =
"OtherTph244_n.egm")
OtherTphStats = readRDS("OtherTph244_n.egm")
getTrainOBBAIC(OtherTphStats)
oTPH = predictEnsembleGam(OtherTphStats[[1]], AnfDf)
AnfDf$r_oTPH = as.vector(oTPH[[1]] ^ 2 - AnfDf$Other1TPH)
(mt = moran.test(AnfDf$r_oTPH, nb2listw(nb)))

AnfDfRaw$sqrt_Other1TPH = sqrt(AnfDfRaw$Other1TPH)
(resp = "sqrt_Other1TPH")
predRaw = getGamSigFldNames(AnfDfRaw, resp, allImagePredRaw, 0.05, fam =
gaussian()[[1]])
OtherTphStatsRaw = runSim(resp, predRaw, AnfDfRaw, 50, ptrain = 0.75, kfact = 10,
outMdl = "OtherTphRaw244_n.egm")
OtherTphStatsRaw = readRDS("OtherTphRaw244_n.egm")
getTrainOBBAIC(OtherTphStatsRaw)
oTPHraw = predictEnsembleGam(OtherTphStatsRaw[[1]], AnfDfRaw)
AnfDfRaw$r_oTPH = as.vector(oTPHraw[[1]] ^ 2 - AnfDfRaw$Other1TPH)
(mt = moran.test(AnfDfRaw$r_oTPH, nb2listw(nb)))
library(rgdal)
library(raster)
library(mgcv)

createDomain <- function(Pred, rndDf, plotDf, outName, k = 100) {
  rndLocSub = subset(rndDf, select = names(Pred))
  AnfDfSub = subset(plotDf, select = names(Pred))

```

```

km = kmeans(rndLocSub, k)
ucls = unique(km$cluster)
v1 = predict(km, AnfDfSub)
uv1 = unique(v1)
msv1 = ucls[!(ucls %in% uv1)]
msdf = data.frame(id = msv1, v = rep(1, length(msv1)))
beginCluster(8)
krs = clusterR(Pred, predict, args = list(model = km, fun = predict.kmeans),
progress = 'text')
drs = clusterR(krs, subs, args = list(msdf), filename = outName, format =
"GTiff", datatype = "INT1U", NAFlag = 255, progress = 'text', overwrite = TRUE)
endCluster()
return(drs)
}

predict.kmeans <- function(x, newdata) {
  return(apply(newdata, 1, function(r) which.min(colSums((t(x$centers) - r) ^
2))))
}

predictEnsembleGam <- function(bGamMdl, data, trunc = 0) {
  fm = bGamMdl[[1]]$family
  m = NULL
  s = NULL
  mdl1 = length(bGamMdl)
  n = nrow(data)
  if (fm$family == "multinom") {
    nlp = fm$nlp
    sm = matrix(rep(0, n * (nlp + 1)), nrow = n, ncol = nlp + 1)
    s2m = sm
    for (i in seq(mdl1)) {
      mdl = bGamMdl[[i]]
      p = predict(mdl, data, type = "response")
      sm = sm + p
      s2m = s2m + p ^ 2
    }
    m = sm / mdl1
    s = sqrt((s2m - ((sm ^ 2) / mdl1)) / (mdl1 - 1))
    return(cbind(m,s))
  }
  else {
    sV = vector(mode = "double", length = n)

```

```

s2V = vector(mode = "double", length = n)
for (i in seq(mdls)) {
  mdl = bGamMdl[[i]]
  p = (predict(mdl, data, type = "response"))
  if (fm$family != "binomial") {
    p = p ^ 2
  }
  p[p < trunc] = trunc
  sV = sV + p
  s2V = s2V + p ^ 2
}
m = sV / mdls
s = sqrt((s2V - ((sV ^ 2) / mdls)) / (mdls - 1))
return(cbind(m,s))
}

}

baseDir <- "C:\\Users\\jshogland\\Documents\\John\\projects\\RESTORE\\Outputs3"
setwd(baseDir)
rasterOptions(tmpdir = paste(getwd(), "tmp", sep = "/"))
tmpDir(create = TRUE)

# read rasters
rndLoc = readOGR("RndLoc10000.shp", "RndLoc10000")
rndDf = rndLoc@data
plotLoc = readOGR("PlotsShifted244.shp", "PlotsShifted_244")
AnfDf = plotLoc@data
AOI = shapefile("AOI.shp")
ext = extent(AOI)
LSP =
brick("C:\\Users\\jshogland\\Documents\\John\\projects\\RESTORE\\Predictors\\LS.tif"
)
xmin = 882300
xmax = xmin + 30 * ncol(LSP)
ymin = 731820
ymax = ymin + 30 * nrow(LSP)
newExt = extent(c(xmin, xmax, ymin, ymax))
LSP2 = crop(setExtent(LSP, newExt), ext)
SNP =
crop(brick("C:\\Users\\jshogland\\Documents\\John\\projects\\RESTORE\\Predictors\\SN
.tif"),ext)

```

```

ARP =
crop(brick("C:\\Users\\jshogland\\Documents\\John\\projects\\RESTORE\\Predictors\\Aerial.tif"),ext)

#subset and combine layers

#DFT
#beginCluster(8)
#DftStats = readRDS("Dft244.egm")
P1 = LSP2[[c(9)]]
P2 = SNP[[c(1, 4, 6, 10)]]
P3 = LSP2[[c(2, 10)]]
P4 = ARP[[c(1)]]
Pred = stack(P1,P2,P3,P4)
names(Pred) <- c("LS_Band16", "SN_Band2", "SN_Band5", "SN_Band7", "SN_Band21",
"LS_Band3", "LS_Band17", "NP_Band1")
#Dft = clusterR(Pred, predict, args = list(model = DftStats[[1]], fun =
predictEnsembleGam, index = 1:6), filename = "DFT.tif", format = "GTiff", verbose =
TRUE, datatype = "FLT4S",NAFlag=-9999, progress='text')
#endCluster()
DFT_D = createDomain(Pred,rndDf = rndDf,plotDf=AnfDf,"DFT_D.tif")

#LPP
#beginCluster(8)
#LppStats = readRDS("LPP244.egm")
P1 = LSP2[[c(1)]]
P2 = SNP[[c(2,3)]]
P3 = LSP2[[c(10)]]
P4 = ARP[[c(2,3)]]
Pred = stack(P1, P2, P3,P4)
names(Pred) <- c("LS_Band2", "SN_Band3", "SN_Band4", "LS_Band17", "NP_Band2",
"NP_Band5")
#LPP = clusterR(Pred, predict, args = list(model = LppStats[[1]], fun =
predictEnsembleGam, index = 1:2), filename = "LPP.tif", format = "GTiff", verbose =
TRUE, datatype = "FLT4S", NAFlag = -9999, progress = 'text')
#endCluster()
LPP_D = createDomain(Pred, rndDf = rndDf, plotDf = AnfDf, "LPP_D.tif")

#baseDir <- "E:\\Projects\\RESTORE\\Outputs2"
#setwd(baseDir)
#rasterOptions(tmpdir = paste(getwd(), "tmp", sep = "/"))
#tmpDir(create = TRUE)

```

```

#PBAH
#beginCluster(8)
(P1 = LSP2[[c(1, 4, 6, 7, 9, 10, 11, 12)]])
(P2 = ARP[[c(2, 4)]])
(Pred = stack(P1, P2))
names(Pred)=c("LS_Band2", "LS_Band8", "LS_Band10", "LS_Band11", "LS_Band16", "LS_Band17",
"LS_Band18", "LS_Band22", "NP_Band2", "NP_Band8")
#PineBahStats = readRDS("PineBah244_n.egm")
#PBAH = clusterR(Pred, predict, args = list(model = PineBahStats[[1]], fun =
predictEnsembleGam, index = 1:2), filename =
"pBAH.tif", format="GTiff", verbose=TRUE, datatype="FLT4S", NAFlag = -9999, progress =
'text')
#endCluster()
PBAH_D = createDomain(Pred, rndDf = rndDf, plotDf = AnfDf, "PBAH_D.tif")

#PTPH
#beginCluster(8)
#PineTphStats = readRDS("PineTph244.egm")
(P1 = LSP2[[c(2,3,4,6,7,10,13)]])
(P2 = SNP[[c(3, 8)]])
(P3 = ARP[[c(2,3)]])
(Pred = stack(P1, P2, P3))
names(Pred) = c("LS_Band3", "LS_Band5", "LS_Band8", "LS_Band10", "LS_Band11",
"LS_Band17", "LS_Band28", "SN_Band4", "SN_Band11", "NP_Band2", "NP_Band5")
#PTPH = clusterR(Pred, predict, args = list(model = PineTphStats[[1]], fun =
predictEnsembleGam, index = 1:2), filename = "pTPH.tif", format = "GTiff", verbose =
TRUE, datatype = "FLT4S", NAFlag = -9999, progress = 'text')
#endCluster()
PTPH_D = createDomain(Pred, rndDf = rndDf, plotDf = AnfDf, "PTPH_D.tif")

#OBAH
#beginCluster(8)
#OtherBahStats = readRDS("OtherBah244_n.egm")
(P1 = LSP2[[c(2, 3, 7)]])
(P2 = SNP[[c(6,7,10,11)]])
(P3 = ARP[[c(2, 3)]])
(Pred = stack(P1, P2, P3))
names(Pred) = c("LS_Band3", "LS_Band5", "LS_Band11", "SN_Band7", "SN_Band9",
"SN_Band21", "SN_Band23", "NP_Band2", "NP_Band5")

```

```

#OBAH = clusterR(Pred, predict, args = list(model = OtherBahStats[[1]], fun =
predictEnsembleGam, index = 1:2), filename = "oBAH.tif", format = "GTiff", verbose =
TRUE, datatype = "FLT4S", NAFlag = -9999, progress = 'text')
#endCluster()
OBAH_D=createDomain(Pred, rndDf = rndDf, plotDf = AnfDf, "OBAH_D.tif")

#OTPH
#beginCluster(8)
#OtherTphStats = readRDS("OtherTph244_n.egm")
(P1 = LSP2[[c(5,8)]])
(P2 = SNP[[c(1,5,6,9)]])
(P3 = ARP[[c(1)]])
(Pred = stack(P1, P2, P3))
names(Pred) = c("LS_Band9", "LS_Band13", "SN_Band2", "SN_Band6", "SN_Band7",
"SN_Band20", "NP_Band1")
#OTPH = clusterR(Pred, predict, args = list(model = OtherTphStats[[1]], fun =
predictEnsembleGam, index = 1:2), filename = "oTPH.tif", format = "GTiff", verbose =
TRUE, datatype = "FLT4S", NAFlag = -9999, progress = 'text')
#endCluster()
OTPH_D=createDomain(Pred, rndDf = rndDf, plotDf = AnfDf, "OTPH_D.tif")

#Create feature domain mask

removeTmpFiles(h = 0)
library(ggplot2)
library(grid)
library(gridExtra)
library(mgcv)

MeanSeAA <- function(df,resp, egam) {
  #df = df244
  #resp = "LPP"
  #egam = LppStats[[1]]
  dfb = df
  unqCls = unique(dfb[resp][[1]])
  clsCnt = length(unqCls)
  fm = egam[[1]]$family$family
  mldCnt = length(egam)
  porMat = matrix(ncol = clsCnt ^ 2, nrow = mldCnt)
  for (m in 1:mldCnt) {
    mdl = egam[[m]]
    tp = predict(mdl, df, type="response")
  }
}

```

```

    if (fm == "multinom") {
      dfb$pred = apply(tp, 1, function(x) which(max(x) == x)) - 1
    }
    else {
      dfb$pred = as.integer(tp > 0.5)
    }
    rvect = vector(mode = "numeric", length = clsCnt ^ 2)
    cCnt = 1
    for (l1 in unqCls) {
      clTot = sum(as.character(dfb[resp][[1]]) == l1)
      for (l2 in unqCls) {
        rvect[cCnt] = sum(as.character(dfb[resp][[1]]) == l1 &
as.character(dfb$pred) == l2)/clTot
        cCnt = cCnt + 1
      }
    }
    porMat[m,] <- rvect
  }
  return(porMat)
}

```

```

getBootMeanCL <- function(x, alpha = 0.05) {
  k = length(x)
  vs = sort(x)
  a = alpha / 2
  ab = round(a * k)
  ae = round((1 - a) * k)
  return(c(mean(vs), vs[ab], vs[ae]))
}

```

```

predictEnsembleGam <- function(bGamMdl, df, trunc = 0) {
  fm = bGamMdl[[1]]$family
  m = NULL
  s = NULL
  mdls = length(bGamMdl)
  n = nrow(df)
  if (fm$family == "multinom") {
    nlp = fm$nlp
    sm = matrix(rep(0, n * (nlp + 1)), nrow = n, ncol = nlp + 1)
    s2m = sm
    for (i in seq(mdls)) {
      mdl = bGamMdl[[i]]
      p = predict(mdl, df, type = "response")
    }
  }
}

```

```

        sm = sm + p
        s2m = s2m + p ^ 2
    }
    m = sm / mdl$
    s = sqrt((s2m - ((sm ^ 2) / mdl$)) / (mdl$ - 1))
    return(list(m, s))
}
else {
    sV = vector(mode = "double", length = n)
    s2V = vector(mode = "double", length = n)
    for (i in seq(mdl$)) {
        mdl = bGamMdl[[i]]
        p = predict(mdl, df, type = "response")
        p[p < trunc] = trunc
        sV = sV + p
        s2V = s2V + p ^ 2
    }
    m = sV / mdl$
    s = sqrt((s2V - ((sV ^ 2) / mdl$)) / (mdl$ - 1))
    return(list(m, s))
}
}
createGraph <- function(obs, est, graphName, axisTitleBlank = TRUE, categorical =
FALSE, pcol = "blue") {
    xTitle = "Observed"
    if (categorical == TRUE) {
        xTitle = "Comp_1"
    }
    yTitle = "Predicted"
    if (axisTitleBlank == TRUE) {
        xTitle = ""
        yTitle = ""
    }
    uVal = max(est)
    cr = cor(est, obs)
    rmse = sqrt(mean((obs - est) ^ 2))
    sbt = paste("Correlation = ", round(cr, digits = 2), ": RMSE = ", round(rmse,
digits = 2), sep = "")
    p4 = ggplot(data.frame(Observed = obs, Predicted = est), aes(x = Observed, y =
Predicted)) +
        theme_bw() +

```



```

    geom_smooth(size = 1, linetype = "dotted", col = "black") +
    geom_point(col = pcol, pch = 1) +
    geom_abline(slope = 1, intercept = 0, col = "red", size = 0.5) +
    theme(legend.position = "none", plot.subtitle = element_text(size = 15),
plot.title = element_text(size = 18), axis.text = element_text(size = 15),
axis.title = element_text(size = 17)) +
    coord_cartesian(ylim = c(0, uVal), xlim = c(0, uVal)) +
    xlab(xTitle) +
    ylab(yTitle) +

    labs(title = graphName, subtitle = sbt)
  return(p4)
}

```

```

getTrainOBBAIC <- function(EGAM) {
  n = length(EGAM[[1]])
  sv1 = 0
  sv12 = 0
  for (m in 1:n) {
    mdl = EGAM[[1]][[m]]
    sv1 = sv1 + mdl$aic
    sv12 = sv12 + mdl$null.deviance
  }
  return(c(mean(EGAM[[3]]), mean(EGAM[[2]]), sv1 / n, sv12 / n))
}

```

### #KS test results

```

baseDir <-
"C:\\Users\\jshogland\\Documents\\John\\projects\\UMGradSchool\\Project\\Papers\\Dis
sertation\\chapter3\\data"
setwd(baseDir)

```

### #Figure 6

```

createTable <- function(df) {
  mAg <- aggregate(df[, 2:3], list(df$GRP), mean)
  sAg <- aggregate(df[, 2:3], list(df$GRP), sd)
  minAg <- aggregate(df[, 2:3], list(df$GRP), min)
  maxAg <- aggregate(df[, 2:3], list(df$GRP), max)
  outTbl <- merge(mAg, sAg, by = "Group.1")
  outTbl <- merge(outTbl, minAg, by = "Group.1")
  outTbl <- merge(outTbl, maxAg, by = "Group.1")
}

```

```

names(outTbl) <- c("Group", "Mean BAH", "Mean TPH", "SD BAH", "SD TPH", "Min
BAH", "Min TPH", "Max BAH", "Max TPH")
return(outTbl)
}

df244 = read.csv("ANFplots244.csv")
dfVis <- data.frame(GRP = rep("Pine", nrow(df244)), BAH = df244$Pine1BAH, TPH =
df244$Pine1TPH)
dfVis <- rbind(dfVis, data.frame(GRP = rep("Other", nrow(df244)), BAH =
df244$Other1BAH, TPH = df244$Other1TPH))
cbPalette <- c("#E69F00", "#56B4E9", "#009E73", "#F0E442")

tbl <- createTable(dfVis)
p1 <- ggplot(data = dfVis, aes(x = BAH, fill = GRP)) +
  geom_histogram(aes(), color = "gray", position = "dodge", alpha = 0.90) +
  theme_bw() +
  theme(legend.position = "none", axis.text = element_text(size = 15), axis.title
= element_text(size = 17), axis.title.y = element_blank()) +
  coord_cartesian(ylim = c(0, 120), xlim = c(0, 80)) +
  xlab("BAH") +
  ylab("Frequency") +
  annotation_custom(tableGrob(format(subset(tbl, select = c(1, 2, 4, 6, 8)),
digits = 2, nsmall = 1), rows = NULL), xmin = 10, ymin = 10) +
  scale_fill_manual(values = cbPalette)

p2 <- ggplot(data = dfVis, aes(x = TPH, fill = GRP)) +
  geom_histogram(aes(), color = "gray", position = "dodge", alpha = 0.90) +
  theme_bw() +
  theme(legend.position = "none", axis.text = element_text(size = 15), axis.title
= element_text(size = 17), axis.title.y = element_blank()) +
  coord_cartesian(ylim = c(0, 120), xlim = c(0, 6000)) +
  xlab("TPH") +
  ylab("Frequency") +
  annotation_custom(tableGrob(format(subset(tbl, select = c(1, 3, 5, 7, 9)),
digits = 2, nsmall = 1), rows = NULL), xmin = 750, ymin = 10) +
  scale_fill_manual(values = cbPalette)

png("fig_plotDistBAH_TPH.png", width = 1200, height = 250, res = 100)
grid.arrange(p1, p2, ncol=2, nrow=1)
dev.off()
plot.new()

```

```

df244$DFT = 2
df244[(df244$Pine1BAH + df244$Other1BAH) < 2, "DFT"] = 0
df244[df244$Other1BAH > df244$Pine1BAH & df244$DFT != 0, "DFT"] = 1
df244$LPP = 0
df244[df244$LP1BAH >= 2, "LPP"] = 1

sum(df244$DFT == 0) / nrow(df244)
sum(df244$DFT == 1) / nrow(df244)
sum(df244$DFT == 2) / nrow(df244)
sum(df244$LPP) / nrow(df244)

```

### #Figure 7

```

eanr = read.csv("EANR_summary.csv")
cbPalette <- c("#56B4E9", "#009E73", "#E69F00")
pl <- ggplot(eanr, aes(x = Source, y = R2, col = as.factor(Source))) +
  geom_boxplot() +
  theme_light() +
  theme(legend.position = "bottom", legend.text = element_text(size = 17, margin =
margin(0, 5, 0, 5)), legend.title = element_blank(), axis.title.x = element_blank(),
axis.title.y = element_blank(), axis.title = element_text(size = 17), axis.text =
element_text(size = 15), axis.text.x = element_blank(), axis.ticks.x =
element_blank()) +
  scale_colour_manual(values = cbPalette) +
  facet_grid(cols = vars(Season))
png("fig_EANR_boxplot.png", width = 600, height = 250, res = 100)
plot(pl)
dev.off()
plot.new()

```

### #Figure 8

```

rnd10000 = read.csv("RndLoc10000.csv")
allPred = names(rnd10000)[3:70]
frm = formula(paste("~", paste(allPred, collapse = "+"), sep = ""))
pca = princomp(frm, data = rnd10000, cor = TRUE)
png("fig_pca.png", width = 600, height = 250, res = 100)
plot(pca)
dev.off()
plot.new()

```

### #Figure 9

```

ksRnd244 = read.csv("KSfplot_244_10000_2.csv")
cbPalette <- c("#E69F00", "#56B4E9", "#009E73", "#F0E442")

```

```

png("fig_KS_244.png", width = 600, height = 400, res = 100)
(p1 = ggplot(subset(ksRnd244, Series == "Sample" | Series == "Population"), aes(x =
XValues, y = YValues, fill = Series, col = Series)) +
  geom_col(position = position_dodge2(preserve = "total")) +
  theme_bw() +
  theme(legend.position = c(0.1, 0.9), legend.title = element_blank(),
plot.subtitle = element_text(size = 15), plot.title = element_text(size = 18),
axis.text = element_text(size = 15), axis.title = element_text(size = 17)) +
  scale_colour_manual(values = cbPalette) +
  scale_fill_manual(values = alpha(cbPalette, 1)) +
  xlab("Bin") +
  ylab("Proportion")
)
dev.off()
plot.new()

```

### #Table 3

```

DftStats = readRDS("Dft244.egm")
getTrainOBBAIC(DftStats)
LppStats = readRDS("LPP244.egm")
getTrainOBBAIC(LppStats)
PineBahStats = readRDS("PineBah244_n.egm")
getTrainOBBAIC(PineBahStats)
PineTphStats = readRDS("PineTph244.egm")
getTrainOBBAIC(PineTphStats)
OtherBahStats = readRDS("OtherBah244_n.egm")
getTrainOBBAIC(OtherBahStats)
OtherTphStats = readRDS("OtherTph244_n.egm")
getTrainOBBAIC(OtherTphStats)

DftStatsRaw = readRDS("DftRaw244.egm")
getTrainOBBAIC(DftStatsRaw)
LppStatsRaw = readRDS("LPPRaw244.egm")
getTrainOBBAIC(LppStatsRaw)
PineBahStatsRaw = readRDS("PineBah244raw_n.egm")
getTrainOBBAIC(PineBahStatsRaw)
PineTphStatsRaw = readRDS("PineTph244raw.egm")
getTrainOBBAIC(PineTphStatsRaw)
OtherBahStatsRaw = readRDS("OtherBahRaw244_n.egm")
getTrainOBBAIC(OtherBahStatsRaw)
OtherTphStatsRaw = readRDS("OtherTphRaw244_n.egm")
getTrainOBBAIC(OtherTphStatsRaw)

```

```

#Figure 10
df244 = read.csv("ANFplots244.csv")
df244$DFT = 2
df244[(df244$Pine1BAH + df244$Other1BAH) < 2, "DFT"] = 0
df244[df244$Other1BAH > df244$Pine1BAH & df244$DFT != 0, "DFT"] = 1
df244$LPP = 0
df244[df244$LP1BAH >= 2, "LPP"] = 1

DftStats = readRDS("Dft244.egm")
dftBootVls = MeanSeAA(df244, "DFT", DftStats[[1]])
(aa = getBootMeanCL(dftBootVls[, 1], 0.05)) #11
(ab = getBootMeanCL(dftBootVls[, 2], 0.05)) #10
(ac = getBootMeanCL(dftBootVls[, 3], 0.05)) #12
(ba = getBootMeanCL(dftBootVls[, 4], 0.05)) #01
(bb = getBootMeanCL(dftBootVls[, 5], 0.05)) #00
(bc = getBootMeanCL(dftBootVls[, 6], 0.05)) #02
(ca = getBootMeanCL(dftBootVls[, 7], 0.05)) #21
(cb = getBootMeanCL(dftBootVls[, 8], 0.05)) #20
(cc = getBootMeanCL(dftBootVls[, 9], 0.05)) #22

sum(df244$DFT == 1)

LppStats = readRDS("LPP244.egm")
lppBootVls = MeanSeAA(df244, "LPP", LppStats[[1]])
(aa = getBootMeanCL(lppBootVls[, 1], 0.05)) #00
(ab = getBootMeanCL(lppBootVls[, 2], 0.05)) #01
(ba = getBootMeanCL(lppBootVls[, 3], 0.05)) #11
(bb = getBootMeanCL(lppBootVls[, 4], 0.05)) #10

sum(df244$LPP)

#Figure 11
AnfDf = read.csv("ANFplots244.csv")
PineBahStats = readRDS("PineBah244_n.egm")
pineBAH = predictEnsembleGam(PineBahStats[[1]], AnfDf)
p1 = createGraph(AnfDf$Pine1BAH, pineBAH[[1]] ^ 2, "Pine BAH")

PineTphStats = readRDS("PineTph244.egm")
pineTPH = predictEnsembleGam(PineTphStats[[1]], AnfDf)
p3=createGraph(AnfDf$Pine1TPH, pineTPH[[1]] ^ 2, "Pine TPH")

```

```

otherBahStats = readRDS("OtherBah244_n.egm")
otherBAH = predictEnsembleGam(otherBahStats[[1]], AnfDf)
p2=createGraph(AnfDf$Other1BAH, otherBAH[[1]] ^ 2, "Other BAH")

otherTphStats = readRDS("OtherTph244_n.egm")
otherTPH = predictEnsembleGam(otherTphStats[[1]], AnfDf)
p4 = createGraph(AnfDf$Other1TPH, otherTPH[[1]] ^ 2, "Other TPH")

png("fig_PredvsObs.png", width = 900, height = 900, res = 100)
grid.arrange(p1, p2, p3, p4, nrow = 2, ncol = 2)
dev.off()
plot.new()

pTph1000 = data.frame(obsTPH = AnfDf$Pine1TPH, obsBAH = AnfDf$Pine1BAH, predTPH =
pineTPH[[1]] ^ 2, predBAH = pineBAH[[1]])
pTph1000sub = subset(pTph1000, obsTPH < 1000)
(p5 = createGraph(pTph1000sub$obsTPH, pTph1000sub$predTPH, "Pine TPH (TPH < 1000)"))
(p6 = createGraph(pTph1000sub$obsBAH, pTph1000sub$predBAH, "Pine BAH (TPH < 1000)"))

oTph1000 = data.frame(obsTPH = AnfDf$Other1TPH, obsBAH = AnfDf$Other1BAH, predTPH =
otherTPH[[1]] ^ 2, predBAH = otherBAH[[1]]^2)
oTph1000sub = subset(oTph1000, obsTPH < 1000)
(o5 = createGraph(oTph1000sub$obsTPH, oTph1000sub$predTPH, "Other TPH (TPH <
1000)"))
(o6 = createGraph(oTph1000sub$obsBAH, oTph1000sub$predBAH, "Other BAH (TPH < 1000)"))

#Table 4
library(rgdal)
library(sp)
library(spdep)
plts = readOGR("PlotsShifted244.shp")
knn = knearneigh(plts@coords, k = 1)
nb = knn2nb(knn)
AnfDf = plts@data
AnfDf$DFT = 2
AnfDf[(AnfDf$Pine1BAH + AnfDf$Other1BAH) < 2, "DFT"] = 0
AnfDf[AnfDf$Other1BAH > AnfDf$Pine1BAH & AnfDf$DFT != 0, "DFT"] = 1
AnfDf$LPP = 0
AnfDf[AnfDf$LP1BAH >= 2, "LPP"] = 1
crd = plts@coords
AnfDf$X = crd[, 1]
AnfDf$Y = crd[, 2]
DftStats = readRDS("Dft244.egm")

```

```

pDFT = predictEnsembleGam(DftStats[[1]], AnfDf)
AnfDf$r_pDFT0 = as.integer(AnfDf$DFT == 0) - pDFT[[1]][, 1]
AnfDf$r_pDFT1 = as.integer(AnfDf$DFT == 1) - pDFT[[1]][, 2]
AnfDf$r_pDFT2 = as.integer(AnfDf$DFT == 2) - pDFT[[1]][, 3]
plts@data = AnfDf
(mt = moran.test(AnfDf$r_pDFT0, nb2listw(nb)))
(mt = moran.test(AnfDf$r_pDFT1, nb2listw(nb)))
(mt = moran.test(AnfDf$r_pDFT2, nb2listw(nb)))

LppStats = readRDS("LPP244.egm")
pLPP = predictEnsembleGam(LppStats[[1]], AnfDf)[[1]]
AnfDf$r_pLPP0 = as.vector((1 - pLPP) - as.integer(AnfDf$LPP == 0))
AnfDf$r_pLPP1 = as.vector(pLPP - as.integer(AnfDf$LPP == 1))
plts@data = AnfDf
(mt = moran.test(AnfDf$r_pLPP0, nb2listw(nb)))
(mt = moran.test(AnfDf$r_pLPP1, nb2listw(nb)))

PineBahStats = readRDS("PineBah244_n.egm")
pBAH = predictEnsembleGam(PineBahStats[[1]], AnfDf)
AnfDf$r_pBAH = as.vector(pBAH[[1]]^2 - AnfDf$Pine1BAH)
plts@data = AnfDf
(mt = moran.test(AnfDf$r_pBAH, nb2listw(nb)))

PineTphStats = readRDS("PineTph244.egm")
pTPH = predictEnsembleGam(PineTphStats[[1]], AnfDf)
AnfDf$r_pTPH = as.vector(pTPH[[1]]^2 - AnfDf$Pine1TPH)
plts@data = AnfDf
(mt = moran.test(AnfDf$r_pTPH, nb2listw(nb)))

OtherBahStats = readRDS("OtherBah244_n.egm")
oBAH = predictEnsembleGam(OtherBahStats[[1]], AnfDf)
AnfDf$r_oBAH = as.vector(oBAH[[1]]^2 - AnfDf$Other1BAH)
plts@data = AnfDf
(mt = moran.test(AnfDf$r_oBAH, nb2listw(nb)))

OtherTphStats = readRDS("OtherTph244_n.egm")
oTPH = predictEnsembleGam(OtherTphStats[[1]], AnfDf)
AnfDf$r_oTPH = as.vector(oTPH[[1]]^2 - AnfDf$Other1TPH)
plts@data = AnfDf
(mt = moran.test(AnfDf$r_oTPH, nb2listw(nb)))

writeOGR(plts, "testing", layer="testing", driver = "ESRI Shapefile")

```

## Centering Plot Locations (Python)



```

import os, requests, urlparse

def downloadFile(url,outPath):
    success = True
    uName = 'jshogland'
    pWord = 'Sentinelhoggs1!'
    r = requests.get(url,auth=(uName,pWord))
    if (r.status_code == 200):
        try:
            with open(outPath,'wb') as out:
                for bits in r.iter_content():
                    out.write(bits)
        except exc:
            print(str(exc))
            success = False
    return success

fl = open(mPath,'r')
lns = fl.readlines();
urlDic = {}
for l in lns:
    larr = l.split('<')
    for p in larr:
        parr = p.split('/>')
        nm = ''
        url = ''
        eInd = 0
        for e in parr:
            if e == "name": nm = parr[eInd + 1]
            if e == "url" : url = parr[eInd + 1]
            eInd = eInd+1
        urlDic[nm]=url

import arcpy, os, math
wksPath=r'C:\Users\jshogland\Documents\John\projects\UMGradSchool\Project\Papers\Dis
sertation\chapter4\data'
rsDbPath=r'C:\Users\jshogland\Documents\John\projects\RESTORE\RESTORE.gdb'
pltPath = rsDbPath+"\\Plots"
gpsPath = rsDbPath+"\\GPS"
subPath = rsDbPath+"\\Subplots"
treesPath= rsDbPath+"\\Trees"
grpPath=wksPath+"\\spGrpPath.grp"
arcpy.env.workspace = wksPath

```

```

arcpy.env.scratchWorkspace = wksPath + "\\tmp"
arcpy.env.overwriteOutput = True

###functions
def getSpeciesGroup(path="all"):
    outDic = {}
    if(path.lower()!="all"):
        fl = open(path,'r')
        lns = fl.readlines()
        for l in lns:
            llst=l.split(":")
            l2lst=llst[1].split(",")
            for l2 in l2lst:
                outDic[l2] = llst[0]
    return outDic

###update GPS coordinates
pltShift = arcpy.CopyFeatures_management(pltPath, "PlotsShift")
gpsDic={}

with arcpy.da.SearchCursor(gpsPath,["plotUid","x","y"]) as scur:
    for rw in scur:
        id = rw[0]
        x = rw[1]
        y = rw[2]
        if(gpsDic.has_key(id)):
            lsVl=gpsDic[id]
            x= x+lsVl[0]
            y= y+lsVl[1]
            cnt = 1+lsVl[2]
        else:
            cnt = 1
        gpsDic[id]=[x,y,cnt]

with arcpy.da.UpdateCursor(pltShift,["SHAPE@", "UID"]) as ucur:
    for rw in ucur:
        geo = rw[0]
        id = rw[1]
        for pnt in geo:
            if(gpsDic.has_key(id)):
                gpsVlLst = gpsDic[id]

```

```

        pnt.X=gpsV1Lst[0]/gpsV1Lst[2]-9
        pnt.Y=9+gpsV1Lst[1]/gpsV1Lst[2]
    else:
        pnt.X = pnt.X -9
        pnt.Y = pnt.Y + 9
    rw[0]=pnt
    ucur.updateRow(rw)

###create group field and populate
gpDic = getSpeciesGroup()
arcpy.AddField_management(treesPath, "grp", "TEXT")
arcpy.AddField_management(treesPath, "sDBH", "FLOAT")
arcpy.AddField_management(treesPath, "sBA", "FLOAT")
weStart = True
ed=arcpy.da.Editor(rsDbPath)
if(not ed.isEditing):
    ed.startEditing(False, False)
    weStart=False
ed.startOperation()
with arcpy.da.UpdateCursor(treesPath, ["uid", "sp", "cnt", "dbh", "grp", "sDBH", "sBA"]) as
scur:
    for rw in scur:
        id = rw[0]
        sp = rw[1]
        grp = "all"
        if(gpDic.has_key(sp)):
            grp = gpDic[sp]
        cnt = rw[2]
        dbh = rw[3]
        dbhm = dbh*0.0254 #inches to meters
        bam = (dbhm/2)**2 * math.pi #ba in meters
        rw[4] = grp
        rw[5] = dbhm*cnt
        rw[6] = bam*cnt
        scur.updateRow(rw)
ed.stopOperation()
if(weStart):
    ed.stopEditing(True)

```

# **Enhanced Aggregate No-Change Regression Library (C#)**

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using ESRI.ArcGIS.esriSystem;
using ESRI.ArcGIS.DataSourcesRaster;
using ESRI.ArcGIS.Geodatabase;
using ESRI.ArcGIS.Geometry;
using esriUtil;
using Accord.Statistics.Analysis;

namespace esriUtil
{
    public class sampleANR
    {
        public sampleANR(IFunctionRasterDataset referenceRs, IFunctionRasterDataset
transformRs, string mdlPath=null, int percentChange=20, IFeatureClass mask=null,int
sampleSize= 1000, int blockSize = 50, rasterUtil rasterUtility = null, string
storeXY=null)
        {
            refRs = referenceRs;
            transRs = transformRs;
            pct = percentChange/200d;
            mskFtrCls = mask;
            rsUtil = rasterUtility;
            n = sampleSize;
            rndPnts = new IPoint[sampleSize];
            refArray = new double[refRs.RasterInfo.BandCount][][];
            tranArray = new double[transRs.RasterInfo.BandCount][][];
            useArray = new bool[transRs.RasterInfo.BandCount][][];
            minArrayRef = new double[tranArray.Length];
            maxArrayRef = new double[tranArray.Length];
            minArrayTran = new double[tranArray.Length];
            maxArrayTran = new double[tranArray.Length];
            setArrayValues();
            outPth = mdlPath;
            bSize = blockSize;
            stXY = storeXY;
        }
    }
}

```

```

private void setArrayValues()
{
    for (int i = 0; i < minArrayRef.Length; i++)
    {
        minArrayRef[i] = double.MaxValue;
        minArrayTran[i] = double.MaxValue;
        maxArrayRef[i] = double.MinValue;
        maxArrayTran[i] = double.MinValue;
    }
}

private rasterUtil rsUtil = null;
private IFunctionRasterDataset refRs = null;
private IFunctionRasterDataset transRs = null;
private double pct = 0.1;
private IFeatureClass mskFtrCls = null;
private IGeometry geo = null;
private int n = 1000;
private IPoint[] rndPnts = null;
private double[] minArrayRef = null;
private double[] maxArrayRef = null;
private double[] minArrayTran = null;
private double[] maxArrayTran = null;
private double[][] coef = null;
private string outPth = null;
private string stXY = null;

public IFunctionRasterDataset normalize()
{
    //Console.WriteLine("creating points");
    checkSR();
    if(stXY!=null)
    {
        writeXY();
    }
    //Console.WriteLine("extracting array values");
    getValues();
    //Console.WriteLine("scaling and getting unchanged pixels");
    getUnchangedCells();
    //Console.WriteLine("getting coefficients");
    coef = getCoef();//intercept, slope, sse, r2 by band
    if (outPth != null)

```

```

    {
        writeCoef();
    }
    //Console.WriteLine("transforming values");
    IFunctionRasterDataset outRs = transform();

    return outRs;
}

private void writeXY()
{
    IRaster2 rs2 = (IRaster2)rsUtil.createRaster(transRs);
    double conv = (refRs.RasterInfo.CellSize.X /
transRs.RasterInfo.CellSize.X);
    int offSetCells = System.Convert.ToInt32(conv * bSize / 2);
    using (System.IO.StreamWriter sw = new System.IO.StreamWriter(stXY))
    {
        sw.WriteLine("X,Y");
        foreach (IPoint p in rndPnts)
        {
            sw.WriteLine(p.X.ToString() + "," + p.Y.ToString());
            int c, r;
            rs2.MapToPixel(p.X, p.Y, out c, out r);
            double nx, ny;
            IPnt tl = new PntClass();
            tl.X = c - offSetCells;
            tl.Y = r - offSetCells;
            rs2.PixelToMap(System.Convert.ToInt32(tl.X),
System.Convert.ToInt32(tl.Y), out nx, out ny);
            sw.WriteLine(nx.ToString() + "," + ny.ToString());
            IPnt br = new PntClass();
            br.X = c + offSetCells;
            br.Y = r + offSetCells;
            rs2.PixelToMap(System.Convert.ToInt32(br.X),
System.Convert.ToInt32(br.Y), out nx, out ny);
            sw.WriteLine(nx.ToString() + "," + ny.ToString());
            IPnt tr = new PntClass();
            tr.X = c - offSetCells;
            tr.Y = r + offSetCells;
            rs2.PixelToMap(System.Convert.ToInt32(tr.X),
System.Convert.ToInt32(tr.Y), out nx, out ny);
            sw.WriteLine(nx.ToString() + "," + ny.ToString());
            IPnt bl = new PntClass();

```

```

        bl.X = c + offSetCells;
        bl.Y = r - offSetCells;
        rs2.PixelToMap(System.Convert.ToInt32(bl.X),
System.Convert.ToInt32(bl.Y), out nx, out ny);
        sw.WriteLine(nx.ToString() + "," + ny.ToString());
    }
    sw.Close();
}
}

private void writeCoef()
{
    using (System.IO.StreamWriter sw = new System.IO.StreamWriter(outPth))
    {
        sw.WriteLine("Normalize");
        sw.WriteLine(pct.ToString());
        sw.WriteLine(refRs.RasterInfo.PixelType.ToString());
        sw.WriteLine(coef.Length.ToString());
        for (int i = 0; i < coef.Length; i++)
        {
            sw.WriteLine(String.Join(",", (from double d in coef[i] select
d.ToString()).ToArray()));
        }
        sw.Close();
    }
}

public IFunctionRasterDataset normalize(string mdlPath)
{
    readCoef(mdlPath);
    IFunctionRasterDataset outRs = transform();
    return outRs;
}

private void readCoef(string mdlPath)
{
    using (System.IO.StreamReader sr = new System.IO.StreamReader(outPth))
    {
        sr.ReadLine();
        double pct = System.Convert.ToSingle(sr.ReadLine());
        sr.ReadLine();
        int bnds = System.Convert.ToInt32(sr.ReadLine());
        coef = new double[bnds][];
    }
}

```



```

        for (int i = 0; i < coef.Length; i++)
        {
            string[] cArr = sr.ReadLine().Split(new char[',']);
            coef[i] = (from string s in cArr select
System.Convert.ToDouble(s)).ToArray();
        }
        sr.Close();
    }
}

private IFunctionRasterDataset transform()
{
    IFunctionRasterDataset outRs = null;
    IRasterBandCollection rsBc = new RasterClass();
    for (int i = 0; i < coef.Length; i++)
    {
        double[] c = coef[i];
        double intercept = c[0];
        double slope = c[1];
        IFunctionRasterDataset tRs = rsUtil.getBand(transRs, i);
        IFunctionRasterDataset pRs = rsUtil.calcArithmeticFunction(tRs,
slope, esriRasterArithmeticOperation.esriRasterMultiply);
        IFunctionRasterDataset fRs = rsUtil.calcArithmeticFunction(pRs,
intercept, esriRasterArithmeticOperation.esriRasterPlus);
        IFunctionRasterDataset bRs = rsUtil.convertToDiffFormatFunction(fRs,
refRs.RasterInfo.PixelType);
        rsBc.AppendBand(((IRasterBandCollection)bRs).Item(0));
    }
    outRs = rsUtil.compositeBandFunction(rsBc);
    return outRs;
}

private double[][] getCoef()
{
    double[][] outCoef = new double[useArray.Length][];
    int cellCntCheck = bSize * bSize / 4;
    for (int b = 0; b < useArray.Length; b++)
    {
        double[] xVls = new double[rndPnts.Length];
        double[] yVls = new double[rndPnts.Length];
        int[] cntVls = new int[rndPnts.Length];
        for (int r = 0; r < useArray[b].Length; r++)

```

```

{
    for (int c = 0; c < useArray[b][r].Length; c++)
    {

        if (useArray[b][r][c])
        {
            xVls[r] = xVls[r] + tranArray[b][r][c];
            yVls[r] = yVls[r] + refArray[b][r][c];
            cntVls[r] = cntVls[r] + 1;
        }

    }
}
List<double> xVlsLst = new List<double>();
List<double> yVlsLst = new List<double>();
List<int> cntVlsLst = new List<int>();
for (int v = 0; v < xVls.Length; v++)
{
    //Console.WriteLine(cntVls[v].ToString());
    //Console.WriteLine(xVls[v].ToString());
    //Console.WriteLine(yVls[v].ToString());
    int cnt = cntVls[v];
    if (cnt > cellCntCheck)
    {
        xVlsLst.Add(xVls[v] / cnt);
        yVlsLst.Add(yVls[v] / cnt);
        cntVlsLst.Add(cnt);
    }
}
xVls = xVlsLst.ToArray();
yVls = yVlsLst.ToArray();
cntVls = cntVlsLst.ToArray();
 Accord.Statistics.Models.Regression.Linear.SimpleLinearRegression
slr = new Accord.Statistics.Models.Regression.Linear.SimpleLinearRegression();
double sse = slr.Regress(xVls, yVls);
double r2 = slr.CoefficientOfDetermination(xVls, yVls);
outCoef[b] = new double[4]{ slr.Intercept, slr.Slope, sse,r2};
if (stXY != null)
{
    string pCoef = stXY.Replace(".csv", "_" + b.ToString() +
".txt");
    using (System.IO.StreamWriter sw = new
System.IO.StreamWriter(pCoef))

```

```

        {
            sw.WriteLine("X,Y,CNT");
            for (int v = 0; v < xVls.Length; v++)
            {
                sw.WriteLine(xVls[v].ToString() + "," +
yVls[v].ToString() + "," + cntVls[v].ToString());
            }
            sw.Close();
        }
    }

    return (outCoef);
}

private void getUnchangedCells()
{
    Random rnd = new Random();
    double[][][] difArr = new double[refArray.Length][][];
    for (int b = 0; b < refArray.Length; b++)
    {
        List<double> difLst = new List<double>();
        difArr[b] = new double[refArray[b].Length] [];
        double minVal = minArrayRef[b];
        double maxVal = maxArrayRef[b];
        double dif = maxVal - minVal;
        double tMinVal = minArrayTran[b];
        double tMaxVal = maxArrayTran[b];
        double tdif = tMaxVal - tMinVal;
        for (int r = 0; r < refArray[b].Length; r++)
        {
            difArr[b][r] = new double[refArray[b][r].Length];
            for (int c = 0; c < refArray[b][r].Length; c++)
            {
                double refOldV1 = refArray[b][r][c];
                double tranOldV1 = tranArray[b][r][c];
                if (!(refOldV1 == -9999 || tranOldV1 == -9999))
                {
                    double refNewV1 = (refOldV1 - minVal) / dif;
                    double tranNewV1 = (tranOldV1 - minVal) / tdif;
                    double sdif = refNewV1 - tranNewV1;
                }
            }
        }
    }
}

```

```

        difArr[b][r][c] = sdif;
        if (rnd.NextDouble() <= 0.1)
        {
            difLst.Add(sdif);
        }
    }
    else
    {
        difArr[b][r][c] = -9999;
    }
}
}
difLst.Sort();
int vlIndex = System.Convert.ToInt32(difLst.Count*pct);
double smallValue = difLst[vlIndex];
double largeValue = difLst[difLst.Count-vlIndex];
for (int r = 0; r < difArr[b].Length; r++)
{
    for (int c = 0; c < difArr[b][r].Length; c++)
    {
        double vl = difArr[b][r][c];
        if(vl<largeValue&&vl>smallValue)
        {
            useArray[b][r][c] = true;
        }
    }
}
}
}

```

```

private int bSize = 50;
private double[][][] refArray = null;
private double[][][] tranArray = null;
private bool[][][] useArray = null;
private void getValues()
{
    int bSizeCnt = bSize * bSize;
    for (int i = 0; i < refArray.Length; i++)
    {
        refArray[i] = new double[n][];
        tranArray[i] = new double[n][];
        useArray[i] = new bool[n][];
    }
}

```

```

    for (int j = 0; j < n; j++)
    {
        refArray[i][j] = new double[bSizeCnt];
        tranArray[i][j] = new double[bSizeCnt];
        useArray[i][j] = new bool[bSizeCnt];
    }
}
int offSetCells = bSize / 2;
IRaster2 rs2 = (IRaster2)rsUtil.createRaster(refRs);
IRaster rs = (IRaster)rs2;
int nIndex = 0;
foreach(IPoint pnt in rndPnts)
{
    fillTransValues(pnt,nIndex);
    IPnt t1Pnt = new PntClass();
    IPnt pbSize = new PntClass();
    pbSize.X = bSize;
    pbSize.Y = bSize;
    int c, r;
    rs2.MapToPixel(pnt.X, pnt.Y, out c, out r);
    c = c - offSetCells;
    r = r - offSetCells;
    t1Pnt.X = c;
    t1Pnt.Y = r;
    IPixelBlock pb = rs.CreatePixelBlock(pbSize);
    rs.Read(t1Pnt, pb);
    for (int b = 0; b < pb.Planes; b++)
    {
        System.Array v1Arr =
(System.Array)((IPixelBlock3)pb).get_PixelData(b);
        int vIndex = 0;
        for (int rw = 0; rw < pb.Height; rw++)
        {
            for (int cl = 0; cl < pb.Width; cl++)
            {
                object v1Obj = v1Arr.GetValue(rw, cl);
                double v1f = -9999;
                if(v1Obj!=null)
                {
                    v1f = System.Convert.ToDouble(v1Obj);
                    if (v1f < minArrayRef[b]) minArrayRef[b] = v1f;
                    if (v1f > maxArrayRef[b]) maxArrayRef[b] = v1f;
                }
            }
        }
    }
}

```

```

        refArray[b][nIndex][vIndex] = v1f;
        vIndex = vIndex + 1;
    }
}
}
nIndex = nIndex + 1;
}
}

private void fillTransValues(IPoint pnt, int nIndex)
{
    IRaster rs = rsUtil.createRaster(transRs);
    IRaster2 rs2 = (IRaster2)rs;
    double conv = (refRs.RasterInfo.CellSize.X /
transRs.RasterInfo.CellSize.X);
    int offSetCells = System.Convert.ToInt32(conv*bSize/2);
    IPnt tlPnt = new PntClass();
    IPnt pbSize = new PntClass();
    pbSize.X = System.Convert.ToInt32(bSize*conv);
    pbSize.Y = System.Convert.ToInt32(bSize*conv);
    int c, r;
    rs2.MapToPixel(pnt.X, pnt.Y, out c, out r);
    c = c - offSetCells;
    r = r - offSetCells;
    tlPnt.X = c;
    tlPnt.Y = r;
    IPixelBlock pb = rs.CreatePixelBlock(pbSize);
    rs.Read(tlPnt, pb);
    for (int b = 0; b < pb.Planes; b++)
    {
        System.Array v1Arr =
(System.Array)((IPixelBlock3)pb).get_PixelData(b);
        int vIndex = 0;
        double[] cellCnt = new double[bSize*bSize];
        for (int rw = 0; rw < pb.Height; rw++)
        {
            for (int cl = 0; cl < pb.Width; cl++)
            {
                object v1Obj = v1Arr.GetValue(rw, cl);
                if (v1Obj != null)
                {
                    int bIndex = (int)(rw/conv) * bSize + (int)(cl / conv);
                    double v1f = System.Convert.ToSingle(v1Obj);

```

```

        tranArray[b][nIndex][bIndex] = v1f +
tranArray[b][nIndex][bIndex];
        cellCnt[bIndex] = cellCnt[bIndex] + 1;
    }
    vIndex = vIndex + 1;
}
}
for(int i=0; i < cellCnt.Length;i++)
{
    double cellCntV1 = cellCnt[i];
    if (cellCntV1 > 0)
    {
        double meanValue = tranArray[b][nIndex][i] / cellCnt[i];
        tranArray[b][nIndex][i] = meanValue;
        if (meanValue < minArrayTran[b]) minArrayTran[b] =
meanValue;

        if (meanValue > maxArrayTran[b]) maxArrayTran[b] =
meanValue;

    }
    else
    {
        tranArray[b][nIndex][i] = -9999;
    }
}
}
}

```

```

private void checkSR()
{
    ISpatialReference srRef = refRs.RasterInfo.SpatialReference;
    ISpatialReference srTrans = transRs.RasterInfo.SpatialReference;
    ISpatialReference srFtrCls = null;
    ITopologicalOperator tp;
    if (mskFtrCls != null)
    {
        srFtrCls = ((IGeoDataset)mskFtrCls).SpatialReference;
        IGeometryCollection geoColl = new PolygonClass();
        object obj = Type.Missing;
        IFeatureCursor ftrCur = mskFtrCls.Search(null, true);
        IFeature ftr = ftrCur.NextFeature();
        geo = ftr.ShapeCopy;
    }
}

```

```

tp = (ITopologicalOperator)geo;
ftr = ftrCur.NextFeature();
while (ftr != null)
{
    IGeometry geo2 = ftr.ShapeCopy;
    geo = tp.Union(geo2);
    tp = (ITopologicalOperator)geo;
    ftr = ftrCur.NextFeature();
}
System.Runtime.InteropServices.Marshal.ReleaseComObject(ftrCur);
if(srRef.FactoryCode != srFtrCls.FactoryCode)
{
    geo.Project(srRef);
}
tp = (ITopologicalOperator)geo;
geo = tp.Intersect((IGeometry)refRs.RasterInfo.Extent,
esriGeometryDimension.esriGeometry2Dimension);
}
else
{
    IEnvelope env = refRs.RasterInfo.Extent;
    IGeometryBridge2 geoBr = new GeometryEnvironmentClass();
    IPointCollection4 pntCol = new PolygonClass();
    ((IGeometry)pntCol).SpatialReference =
refRs.RasterInfo.SpatialReference;
    object mis = Type.Missing;
    pntCol.AddPoint(env.UpperLeft);
    pntCol.AddPoint(env.UpperRight);
    pntCol.AddPoint(env.LowerRight);
    pntCol.AddPoint(env.LowerLeft);
    ((IPolygon)pntCol).Close();
    geo = (IGeometry)pntCol;
    //Console.WriteLine("Area = " + ((IArea)geo).Area.ToString());
}
if(srRef.FactoryCode!=srTrans.FactoryCode)
{
    transRs = rsUtil.reprojectRasterFunction(transRs, srRef);
}

//intersect with boundary of Trans Raster
tp = (ITopologicalOperator)geo;

```



```

        geo = tp.Intersect((IGeometry)transRs.RasterInfo.Extent,
esriGeometryDimension.esriGeometry2Dimension);
        //buffer inside raster half block size
        tp = (ITopologicalOperator)geo;
        geo = tp.Buffer(-1 * bSize * refRs.RasterInfo.CellSize.X / 2);
        IRaster2 rs2 = (IRaster2)rsUtil.createRaster(refRs);
        IPoint ulPnt = geo.Envelope.UpperLeft;
        IPoint lrPnt = geo.Envelope.LowerRight;
        int rStartClm, rStartRw;
        rs2.MapToPixel(ulPnt.X, ulPnt.Y, out rStartClm, out rStartRw);
        int endClm, endRw;
        rs2.MapToPixel(lrPnt.X, lrPnt.Y, out endClm, out endRw);
        int tCells = (endClm - rStartClm) * (endRw - rStartRw);
        double px, py;
        if(n>=(tCells*0.5)) //get all cells from ref Raster
        {
            int iCnt = 0;
            rndPnts = new IPoint[tCells];
            for (int c = rStartClm; c <= endClm; c++)
            {
                for (int r = rStartRw; r <= endRw ; r++)
                {
                    rs2.PixelToMap(c, r, out px, out py);
                    IPoint pnt = new PointClass();
                    pnt.PutCoords(px, py);
                    rndPnts[iCnt] = pnt;
                }
                iCnt = iCnt + 1;
            }
        }
        else //randomly chose cells from ref Raster up to n
        {
            Random rnd = new Random();
            HashSet<string> sCheck = new HashSet<string>();
            int iCnt = 0;
            rndPnts = new IPoint[n];
            while (sCheck.Count < n)
            {
                int c1m = rnd.Next(rStartClm, endClm);
                int rw = rnd.Next(rStartRw, endRw);
                string rwclm = rw.ToString() + "_" + c1m.ToString();
                if (!sCheck.Contains(rwclm))
                {

```

```
rs2.PixelToMap(c1m, rw, out px, out py);
IPoint pnt = new PointClass();
pnt.PutCoords(px, py);
IRelationalOperator ro = (IRelationalOperator)pnt;
if (ro.Within(geo))
{
    rndPnts[iCnt] = pnt;
    iCnt = iCnt + 1;
    sCheck.Add(rwclm);
}
}
}
n = rndPnts.Length;
}
}
}
```

## **ArcPad Library (Mobile Data Collection)**

## DataTable Library (VB Script)

Option Explicit

Class Table

```
    dim rows(), columns(), rwsCnt, clmCnt, columnTypes()
    Public Property Get RowCount()
        RowCount = rwsCnt + 1
    End Property
    Public Property Get ColumnCount()
        ColumnCount = clmCnt + 1
    End Property
    Public Sub createTable(rws,clms)
        Dim rw, i
        rwsCnt = rws - 1
        clmCnt = clms - 1
        redim rows(rwsCnt)
        redim columns(clmCnt)
        For i = 0 To (rwsCnt)
            Set rw = New Row
            rw.createRow(clms)
            Set rows(i) = rw
        Next
    End Sub
    Public Sub addRows(numRows)
        Dim rw, i
        rwsCnt = rwsCnt + numRows
        ReDim Preserve rows(rwsCnt)
        For i = (RowCount-numRows) To rwsCnt
            Set rw = New Row
            rw.createRow ColumnCount
            Set rows(i) = rw
        Next
    End Sub
    Public Sub subtractRow(rwInd)
        Dim trows(), tcnt, i
        If rwInd>rwsCnt Then
            MsgBox "rw Index must be between 0 and " & CStr(rwsCnt)
            Exit Sub
        End If
        If rwInd<0 Then
            MsgBox "rw Index must be between 0 and " & CStr(rwsCnt)
            Exit Sub
        End If
```

```

ReDim trows(rwsCnt-1)
tcnt=0
For i = 0 To rwsCnt
    If i <> rwInd Then
        Set trows(tcnt) = rows(i)
        tcnt = tcnt+1
    End if
Next
ReDim rows(tcnt-1)
rwsCnt = tcnt-1
For i = 0 To rwsCnt
    Set rows(i) = trows(i)
Next
Erase trows
End Sub
Public Function getRow(index)
    Set getRow = rows(index)
End Function
Public Property Get Records()
    Records = rows
End Property
Public Property Get FieldTypes()
    FieldTypes = columnTypes
End Property
Public Property Let FieldTypes(valueArr)
    Dim i
    ReDim columnTypes(UBound(valueArr))
    For i=0 To UBound(valueArr)
        columnTypes(i) = valueArr(i)
    Next
End Property
Public Property Get Fields()
    Fields = columns
End Property
Public Property Let Fields(valueArr)
    Dim i
    ReDim columns(UBound(valueArr))
    For i=0 To UBound(valueArr)
        columns(i) = valueArr(i)
    Next
End Property
Public Function findField(fldName)
    Dim outV1, tv11, tv12, i

```

```

outV1 = -1
tv11 = LCase(CStr(fldName))
For i = 0 To clmCnt
    tv12 = LCase(CStr(columns(i)))
    If tv11 = tv12 Then
        outV1 = i
        Exit For
    End If
Next
findField = outV1
End Function
Public Sub setCellValue(rw,clm,value)
    Dim ind,r,c
    If(Not IsNumeric(clm)) Then
        ind = findField(clm)
    Else
        ind = clm
    End If
    If ind=-1 Then
        MsgBox "Field " & clm & " does not exist"
        Exit Sub
    End if
    value = getCorrectValue(ind,value)
    Set r = getRow(rw)
    r.setCellValue ind, value
    Set r = nothing
End Sub
Public Function getCorrectValue(clm,value)
    Dim outv1,clmTyp
    outv1 = value
    If(Not IsEmpty(columnTypes)) Then
        clmTyp = CInt(columnTypes(clm))
        Select Case clmTyp
            Case 129
                outv1 = CStr(value)
            Case 5
                If IsNumeric(value) then
                    outv1 = CDBl(value)
                Else
                    outv1 = 0
                End if
            Case 7
                If IsDate(value) Then

```

```

        outv1 = CDate(value)
    Else
        outv1 = Now
    End if
Case 11
    outv1 = CBool(value)
End select
End if
getCorrectValue = outv1
End Function
Public Function getCellValue(rw,clm)
    Dim ind,r,outv1
    If(Not IsNumeric(clm)) Then
        ind = findField(clm)
    Else
        ind = clm
    End If
    Set r = getRow(rw)
    outv1 = r.getCellValue(ind)
    getCellValue = outv1
    Set r = Nothing
End Function
Public Function createGuid
    dim TypeLib, guid
    Set TypeLib = CreateObject("Scriptlet.TypeLib")
    guid = TypeLib.Guid
    guid = Left(guid,Len(guid)-2)
    createGuid = guid
    Set TypeLib = Nothing
End Function
Public Function findRowIndex(uidName,uidValue)
    Dim r, ind, outv1, v12
    outv1 = -1
    ind = findField(uidName)
    For r=0 to rwsCnt
        v12 = getCellValue(r,ind)
        If v12 = uidValue Then
            outv1 = r
            Exit For
        End If
    Next
    findRowIndex = outv1
End Function

```

```

Public Sub moveRecord(fromRow, toRow)
    Dim cnt, r, cr, nr
    cnt = 0
    ReDim tArr(rwsCnt)
    Set cr = rows(r)
    For r=0 To rwsCnt
        If(r=toRow) Then
            Set rows(cnt) = rows(fromRow)
            cnt = cnt + 1
        End If
        If(r=fromRow) Then
        Else
            Set nr = rows(cnt)
            Set rows(cnt) = cr
            Set cr = nr
            cnt = cnt + 1
        End if
    Next
    Set nr = Nothing
    Set cr = Nothing
End Sub
Private Sub Class_Terminate()
    Erase rows
    Erase columns
End Sub
End Class

```

Class Dictionary

```

Dim k(),v(), rc, ri
Public Property Get Count()
    Count = rc
End Property
Public Property Get Keys()
    Keys = k
End Property
Public Property Get Values()
    Values = v
End Property
Public sub Add(key,value)
    rc = rc + 1
    ReDim Preserve k(rc-1)
    ReDim Preserve v(rc-1)
    k(rc-1) = key

```



```

        v(rc-1) = value
    End sub
Public sub Remove(key)
    Dim ind, tk(), tv(), i, tcnt
    If(IsNumeric(key)) Then
        ind = CInt(key)
    Else
        ind = FindIndex(key)
    End If
    If ind = -1 Or ind > rc-1 Or rc = 0 Then
    Else
        ReDim tk(rc-2)
        ReDim tv(rc-2)
        tcnt = 0
        For i=0 To rc-1
            If(i=ind) then
            Else
                tk(tcnt) = k(i)
                tv(tcnt) = v(i)
                tcnt = tcnt+1
            End if
        Next
        ReDim k(tcnt-1)
        ReDim v(tcnt-1)
        rc = tcnt
        For i=0 To rc-1
            k(i)=tk(i)
            v(i)=tv(i)
        Next
        Erase tk
        Erase tv
    End if
End sub
Public Function FindValueIndex(value)
    Dim i, outV1
    outV1 = -1
    For i=0 To rc-1
        If(v(i)=value)Then
            outV1 = i
            Exit For
        End if
    Next
    FindValueIndex = outV1

```

```

End Function
Public Function FindIndex(key)
    Dim i, outV1
    outV1 = -1
    key = LCase(key)
    For i=0 To rc-1
        If(LCase(k(i))=key)Then
            outV1 = i
            Exit For
        End if
    Next
    FindIndex = outV1
End Function
Public Function GetValue(key)
    Dim ind,outV1
    outV1 = -1
    ind = FindIndex(key)
    If(ind>-1)Then
        outV1 = v(ind)
    End if
    GetValue = outV1
End Function
Public Function GetKey(value)
    Dim ind,outV1
    outV1 = -1
    ind = FindValueIndex(value)
    If(ind>-1)Then
        outV1 = k(ind)
    End if
    GetKey = outV1
End Function
Private Sub Class_Terminate()
    Erase k
    Erase v
End Sub
End Class

```

```

Class Row
    Dim rw(),clms()
    Public Property Get Cells()
        Cells = rw
    End Property
    Public Property Get Fields()

```

```

        Fields = clms
    End Property
    Public Property Let Fields(valueArr)
        ReDim clms(UBound(valueArr))
        For i=0 To UBound(valueArr)
            clms(i) = valueArr(i)
        Next
    End Property
    Public Sub createRow(numCells)
        Dim r
        ReDim rw(numCells-1)
        For r=0 To (numCells-1)
            Set rw(r) = New Cell
        Next
    End Sub
    Public Function getCellValue(index)
        getCellValue = rw(index).Value
    End Function
    Public Sub setCellValue(index,value)
        rw(index).Value = value
    End Sub
    Public Sub addCells(numCells)
        Dim r
        ReDim Preserve rw(UBound(rw)+numCells)
        For r=(UBound(rw)- numCells) To UBound(rw)
            Set rw(r)= New Cell
        Next
    End Sub
    Public Sub Class_Terminate()
        Erase rw
        Erase clms
    End Sub
End Class

```

```

Class Cell
    Public Value
End Class

```

## Plots Library

### (XML forms)

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<ArcPad>
  <LAYER name="Plots" transparency="1">
    <SYMBOLGY>
      <SIMPLELABELRENDERER visible="true" field="OBJECTID">
        <TEXTSYMBOL angle="0" fontcolor="255,0,0" font="Arial"
fontSize="8.25" horzalignment="left" vertalignment="center" rtl="false"
fontstyle="bold"/>
      </SIMPLELABELRENDERER>
      <VALUEMAPRENDERER lookupfield="VISITED">
        <EXACT value="0" label="NO">
          <SIMPLEMARKERSYMBOL color="Yellow" width="7"
outlinewidth="1"/>
        </EXACT>
        <OTHER label="YES">
          <SIMPLEMARKERSYMBOL color="Red" width="7"
outlinewidth="1"/>
        </OTHER>
      </VALUEMAPRENDERER>
    </SYMBOLGY>
    <FORMS>
      <FORM name="frmtrees" caption="TREES" width="130" height="130"
onok="Call updateDbfValues(&quot;frmtrees&quot;; dtTree)">
        <PAGE name="pgtrees" caption="TREE" sip="false"
onload="Call setTreeCombobox
Set dtTree = fillDataTable(&quot;frmtrees&quot;);
Call updateFormValues(&quot;frmtrees&quot;; dtTree)">
          <BUTTON name="btnprevious" x="3" width="13"
height="12" onclick="Call move_record(-1)
CommonDialog.ShowSIP(False)" caption="&lt;" tooltip="" tabstop="true" border="false"
alignment="center"/>
          <LABEL name="lbltree" x="17" width="17" height="9"
caption="1" tooltip="" group="true" border="false" alignment="center"/>
          <BUTTON name="btnnext" x="36" width="13"
height="12" onclick="Call move_record(1)
CommonDialog.ShowSIP(False)" caption="&gt;" tooltip="" tabstop="true" border="false"
alignment="center"/>
          <LABEL name="lblcount" x="50" width="43"
height="9" caption="of 10" tooltip="" group="true" border="false"/>
          <BUTTON name="btnadd" x="93" width="13"
height="12" onclick="Call addTreeRow
CommonDialog.ShowSIP(False)" caption="+" tooltip="" tabstop="true" border="false"
fontSize="9" fontstyle="bolditalic" alignment="center"/>
        </PAGE>
      </FORM>
    </FORMS>
  </LAYER>

```

```

        <BUTTON name="btndelete" x="107" width="13"
height="12" onclick="Call subtractTreeRow
CommonDialog.ShowSIP(False)" caption="-" tooltip="" tabstop="true" border="false"
fontsize="9" fontstyle="bolditalic" alignment="center"/>
        <COMBOBOX name="cmbbsp" x="14" y="40" width="111"
height="13" defaultvalue="" listtable="" listvaluefield="" listtextfield=""
tooltip="" tabstop="true" border="false" sip="false" limittolist="false"
sort="false"/>
        <LABEL name="lblspecies" y="42" width="10"
height="9" x="1" caption="SP" tooltip="" group="true" border="false"
alignment="right"/>
        <EDIT name="txtdbh" x="36" y="58" width="87"
height="12" defaultvalue="0" tooltip="" tabstop="true" border="true" sip="true"
minvalue="0" maxvalue="100"/>
        <LABEL name="lbldbh" x="15" y="59" width="17"
height="9" caption="DBH" tooltip="" group="true" border="false" alignment="right"/>
        <LABEL name="lblstatus" x="13" y="79" width="23"
height="9" caption="Status" tooltip="" group="true" border="false"/>
        <COMBOBOX name="cmbstatus" x="37" y="77"
width="87" height="13" defaultvalue="" listtable="" listvaluefield=""
listtextfield="" tooltip="" tabstop="true" border="false" sip="false"
limittolist="false" sort="false">
        </COMBOBOX>
        <LABEL name="lblCnt" x="11" y="95" width="24"
height="9" caption="Count" tooltip="" group="true" border="false"/>
        <EDIT name="txtcnt" x="37" y="95" width="86"
height="12" defaultvalue="1" tooltip="" tabstop="true" border="true" sip="true"
minvalue="0" maxvalue="100"/>
        <LABEL name="lblsubplot" x="3" y="18" width="103"
height="12" caption="Plot/Subplot" tooltip="" group="true" border="false"/>
        <EDIT name="txtcomment" x="36" y="112" width="88"
height="12" defaultvalue="" tooltip="" tabstop="true" border="true" sip="false"/>
        <LABEL name="lblcomment" x="1" y="114" width="35"
height="10" caption="Comment" tooltip="" group="true" border="false"/>
    </PAGE>
    <PAGE name="pgtreesview" caption="TREES VIEW" sip="false"
onsetactive="Call update_list_view">
        <LISTBOX name="lsttreesview" width="130"
height="120" defaultvalue="" listtable="" listvaluefield="" listtextfield="" y="9"
onselchange="Call view_select" tooltip="" tabstop="true" border="true" sort="false"
font="Courier New" fontsize="9"/>
        <LABEL name="lblsubplot" width="127" height="9"
caption="subplot" tooltip="" group="true" border="false"/>

```

```

        </PAGE>
    </FORM>
    <EDITFORM name="EDITFORM" caption="Plot" width="130"
height="130" picturepagevisible="false" attributespagevisible="false"
symbologypagevisible="false" geographypagevisible="false" required="false"
onload="Call LoadFormStartup
Call update_labels(&quot;EDITFORM&quot;);
Layer.Forms(&quot;EDITFORM&quot;).Pages(&quot;pgEDITFORM&quot;).Controls(&quot;txtut
ype&quot;).Value = 2
Layer.Forms(&quot;EDITFORM&quot;).Pages(&quot;pgEDITFORM&quot;).Controls(&quot;txtvi
sited&quot;).Value = 1
Layer.Forms(&quot;EDITFORM&quot;).Pages(&quot;pgEDITFORM&quot;).Controls(&quot;txtut
ype&quot;).Enabled = False
Layer.Forms(&quot;EDITFORM&quot;).Pages(&quot;pgEDITFORM&quot;).Controls(&quot;txtvi
sited&quot;).Enabled = False
Layer.Forms(&quot;EDITFORM&quot;).Pages(&quot;pgEDITFORM&quot;).Controls(&quot;txtut
ype&quot;).Visible = False
Layer.Forms(&quot;EDITFORM&quot;).Pages(&quot;pgEDITFORM&quot;).Controls(&quot;txtvi
sited&quot;).Visible = False" sip="false" onok="Map.Refresh" onunload="Call
plotCheck">
        <PAGE name="pgEDITFORM" caption="Subplot" sip="false">
            <BUTTON name="btn1" x="63" y="57" width="50"
height="34" onclick="Call setSubPlotUid(1)
Layer.Forms(&quot;frmSubplot&quot;).Show" caption="Subplot 1" tooltip=""
tabstop="true" border="false" alignment="center"/>
            <BUTTON name="btn2" x="63" y="22" width="50"
height="33" onclick="Call setSubPlotUid(2)
Layer.Forms(&quot;frmSubplot&quot;).Show" caption="Subplot 2" tooltip=""
tabstop="true" border="false" alignment="center"/>
            <BUTTON name="btn3" x="9" y="22" width="50"
height="33" onclick="Call setSubPlotUid(3)
Layer.Forms(&quot;frmSubplot&quot;).Show" caption="Subplot 3" tooltip=""
tabstop="true" border="false" alignment="center"/>
            <BUTTON name="btn4" x="9" y="58" width="50"
height="32" onclick="Call setSubPlotUid(4)
Layer.Forms(&quot;frmSubplot&quot;).Show" caption="Subplot 4" tooltip=""
tabstop="true" border="false" alignment="center"/>
            <BUTTON name="btnGPS" x="93" y="3" width="20"
height="15" onclick="Call collectGPS" caption="GPS" tooltip="" tabstop="true"
border="false" alignment="center"/>
            <BUTTON name="btnPicture" x="63" y="3" width="27"
height="15" onclick="Call takePicture" caption="Picture" tooltip="" tabstop="true"
border="false" alignment="center"/>

```

```

        <BUTTON name="btnSetup" x="9" y="3" width="30"
height="15" onclick="Layer.Forms(&quot;frmSetup&quot;).Show" caption="Setup"
tooltip="" tabstop="true" border="false" alignment="center"/>
        <EDIT name="txtutype" x="118" y="118" width="10"
height="10" defaultvalue="" tooltip="" tabstop="false" border="false" sip="false"
field="UTYPE"/>
        <EDIT name="txtvisited" x="117" y="102" width="10"
height="10" defaultvalue="" tooltip="" tabstop="false" border="false" sip="false"
field="VISITED"/>
        <COMBOBOX name="cmbnatc" x="27" y="95" width="100"
height="13" defaultvalue="" listtable="nccd.dbf" listvaluefield="CODE"
listtextfield="TEXT" tooltip="" tabstop="true" border="false" sip="false"
sort="false" field="NATC"/>
        <LABEL name="lblNatCom" x="11" y="97" width="10"
height="10" caption="NC" tooltip="" group="true" border="false"/>
        <LABEL name="lblcomment" x="1" y="112" width="33"
height="10" caption="Comment" tooltip="" group="true" border="false"/>
        <EDIT name="txtcomment" x="35" y="110" width="91"
height="12" defaultvalue="" tooltip="" tabstop="true" border="true" sip="false"
field="COMMENT"/>
    </PAGE>
</EDITFORM>
<IDENTIFYFORM name="IDENTIFYFORM" caption="IDENTIFY"
width="130" height="130" picturepagevisible="false" attributespagevisible="false"
symbologypagevisible="false" geographypagevisible="false" required="false">
    <PAGE name="pgidentify" caption="IDENTIFY" sip="false">
        <EDIT name="Edit1" x="47" y="12" width="80"
height="12" defaultvalue="" tooltip="" tabstop="true" border="true" readonly="true"
sip="false" field="OBJECTID"/>
        <EDIT name="Edit2" x="47" y="29" width="80"
height="12" defaultvalue="" tooltip="" tabstop="true" border="true" readonly="true"
sip="false" field="VISITED"/>
        <LABEL name="lblPlotId" x="28" y="14" width="18"
height="9" caption="Plot" tooltip="" group="true" border="false"/>
        <LABEL name="lblVisited" x="22" y="30" width="25"
height="9" caption="Visited" tooltip="" group="true" border="false"/>
        <EDIT name="Edit3" tooltip="" x="48" y="46"
width="80" height="12" tabstop="true" border="true" readonly="true" sip="false"
defaultvalue="" field="UID"/>
        <LABEL name="lbluid" caption="UID" tooltip=""
x="28" y="47" width="19" height="10" group="true" border="false"/>
    </PAGE>
</IDENTIFYFORM>

```

```

        <FORM name="frmSetup" caption="Setup" width="130" height="130"
onok="Call updateDbfValues(&quot;frmSetup&quot;;, dtSetup)" onload="Set dtSetup =
fillDataTable(&quot;frmSetup&quot;);
Call setCruiseCombobox
Call updateFormValues(&quot;frmSetup&quot;;, dtSetup)">
        <PAGE name="pgSetup" caption="Setup" sip="false">
                <COMBOBOX name="cmbctype" x="45" y="25" width="82"
height="13" defaultvalue="" listtable="" listvaluefield="" listtextfield=""
tooltip="" tabstop="true" border="false" sip="false" limittolist="false"
sort="false">

                </COMBOBOX>
                <LABEL name="lblcruiseType" x="5" y="27"
width="40" height="9" caption="Cruise Type" tooltip="" group="true" border="false"/>
                <LABEL name="lblValue" x="22" y="42" width="23"
height="9" caption="Value" tooltip="" group="true" border="false"/>
                <EDIT name="txtValue" x="46" y="42" width="80"
height="12" defaultvalue="4" tooltip="" tabstop="true" border="true" sip="true"
minvalue="1" maxvalue="100"/>
                <LABEL name="lblplot" x="7" y="3" width="53"
height="9" caption="Plot" tooltip="" group="true" border="false"/>
                <BUTTON
onclick="Layer.Forms(&quot;frmAddTree&quot;).Show" name="btnUTree" x="45" y="75"
width="81" height="14" caption="Update Tree List" tooltip="" tabstop="true"
border="false" alignment="center"/>
                <EDIT name="txtsubplots" x="46" y="57" width="80"
height="12" defaultvalue="" tooltip="" tabstop="true" border="true" sip="true"/>
                <LABEL name="lblsubplots" x="12" y="58" width="34"
height="10" caption="Sub Plots" tooltip="" group="true" border="false"/>
                <BUTTON
onclick="Application.ExecuteCommand(&quot;gpsoptions&quot;);
'Layer.Forms(&quot;frmgpssetup&quot;).Show" name="btnGpsSetup" x="45" y="90"
width="82" height="14" caption="Update GPS Setup" tooltip="" tabstop="true"
border="false" alignment="center"/>
        </PAGE>
</FORM>
        <FORM name="frmSubplot" caption="Subplot" width="130"
height="130" onload="Set dtSubplot = fillDataTable(&quot;frmSubplot&quot;);
Call SetSubplotComboboxes
Call updateFormValues(&quot;frmSubplot&quot;;, dtSubplot)" onok="Call
updateDbfValues(&quot;frmSubplot&quot;;, dtSubplot)">
        <PAGE name="pgSubplot" caption="Subplot" sip="false">
                <LABEL name="lblpine" x="7" y="114" width="33"
height="9" caption="% Pine" tooltip="" group="true" border="false"/>

```



```

        <LABEL name="lblbare" x="7" y="101" width="33"
height="9" caption="% Bare" tooltip="" group="true" border="false"/>
        <LABEL name="lblbroad" x="7" y="87" width="33"
height="9" caption="% Broad" tooltip="" group="true" border="false"/>
        <LABEL name="lblsaw" x="7" y="71" width="33"
height="9" caption="% Saw" tooltip="" group="true" border="false"/>
        <LABEL name="lblherb" x="7" y="58" width="33"
height="9" caption="% Herb" tooltip="" group="true" border="false"/>
        <LABEL name="lblpctcwd" x="7" y="42" width="33"
height="9" caption="% CWD" tooltip="" group="true" border="false"/>
        <LABEL name="lblburn" x="7" y="27" width="33"
height="9" caption="Last Burn" tooltip="" group="true" border="false"/>
        <LABEL name="lblsubplot" x="3" width="120"
height="9" caption="Plot/Subplot:" tooltip="" group="true" border="false"/>
        <BUTTON
onclick="Layer.Forms(&quot;frmTrees&quot;).Show" name="btnTree" x="3" y="9"
width="37" height="15" caption="Trees" tooltip="" tabstop="true" border="false"
alignment="center"/>
        <COMBOBOX name="cmbburn" x="47" y="26" width="80"
height="100" defaultvalue="" listtable="" listvaluefield="" listtextfield=""
tooltip="" tabstop="true" border="false" sip="false" limittolist="false"
sort="false"/>
        <COMBOBOX name="cmbpctcwd" x="47" y="41"
width="80" height="100" defaultvalue="" listtable="" listvaluefield=""
listtextfield="" tooltip="" tabstop="true" border="false" sip="false"
limittolist="false" sort="false"/>
        <COMBOBOX name="cmbpctherb" x="47" y="55"
width="80" height="100" defaultvalue="" listtable="" listvaluefield=""
listtextfield="" tooltip="" tabstop="true" border="false" sip="false"
limittolist="false" sort="false"/>
        <COMBOBOX name="cmbpctsaw" x="47" y="70"
width="80" height="100" defaultvalue="" listtable="" listvaluefield=""
listtextfield="" tooltip="" tabstop="true" border="false" sip="false"
limittolist="false" sort="false"/>
        <COMBOBOX name="cmbpctbroad" x="47" y="84"
width="80" height="100" defaultvalue="" listtable="" listvaluefield=""
listtextfield="" tooltip="" tabstop="true" border="false" sip="false"
limittolist="false" sort="false"/>
        <COMBOBOX name="cmbpctbare" x="47" y="98"
width="80" height="100" defaultvalue="" listtable="" listvaluefield=""
listtextfield="" tooltip="" tabstop="true" border="false" sip="false"
limittolist="false" sort="false"/>

```

```

        <COMBOBOX name="cmbpctpine" x="47" y="112"
width="80" height="100" defaultvalue="" listtable="" listvaluefield=""
listtextfield="" tooltip="" tabstop="true" border="false" sip="false"
limittolist="false" sort="false"/>
    </PAGE>
</FORM>
<FORM name="frmAddTree" caption="Tree List" width="130"
height="130" onunload="Set luSpcd =
fillDictionary(species,&quot;SPCD&quot;;&quot;COMMON_NAM&quot;)">
    <PAGE name="pgAddTree" caption="TreeList" sip="false"
onload="Call fillListViewAddTree">
        <LISTBOX name="lstTree" y="17" width="127"
height="110" defaultvalue="" listtable="" listvaluefield="" listtextfield="" x="1"
tooltip="" tabstop="true" border="true" sort="false" font="Courier New"
fontsize="9"/>
        <LABEL name="lblTreelist" x="1" width="59"
height="12" y="3" caption="Select to update" tooltip="" group="true"
border="false"/>
        <BUTTON onclick="spUId = &quot;&quot;;
Layer.Forms(&quot;frmSpecies&quot;).Show" name="btnAdd" x="101" width="12"
height="12" y="1" caption="+" tooltip="" tabstop="true" border="false"
alignment="center"/>
        <BUTTON name="btnminus" x="115" y="1" width="12"
height="12" onclick="Call removeAddTree" caption="-" tooltip="" tabstop="true"
border="false" alignment="center"/>
        <BUTTON onclick="Call selectAddTree"
name="btnswitch" x="88" y="1" width="12" height="12" caption="%" tooltip=""
tabstop="true" border="false" alignment="center"/>
    </PAGE>
</FORM>
<FORM name="frmSpecies" caption="Species" width="130"
height="130" onok="Call updateDbfValues(&quot;frmSpecies&quot;;dtSpecies)"
onunload="Call fillListViewAddTree">
    <PAGE name="pgSpecies" caption="Species" sip="false"
onload="Set dtSpecies = fillDataTable(&quot;frmSpecies&quot;);
Call updateFormValues(&quot;frmSpecies&quot;;dtSpecies)">
        <EDIT name="txtSPCD" x="41" y="6" width="80"
height="12" defaultvalue="" tooltip="" tabstop="true" border="true" sip="true"/>
        <LABEL name="lblSPCD" x="19" y="6" width="20"
height="10" caption="Code" tooltip="" group="true" border="false"/>
        <EDIT name="txtCOMMON_NAM" x="41" y="20"
width="80" height="12" defaultvalue="" tooltip="" tabstop="true" border="true"
sip="true"/>

```

```

        <LABEL name="lblCommon" x="9" y="22" width="30"
height="10" caption="Common" tooltip="" group="true" border="false"/>
        <EDIT name="txtGenus" x="41" y="35" width="80"
height="12" defaultvalue="" tooltip="" tabstop="true" border="true" sip="true"/>
        <LABEL name="lblGenus" x="15" y="36" width="24"
height="10" caption="Genus" tooltip="" group="true" border="false"/>
        <EDIT name="txtSPECIES" x="41" y="49" width="80"
height="12" defaultvalue="" tooltip="" tabstop="true" border="true" sip="true"/>
        <EDIT name="txtVARIETY" x="41" y="64" width="80"
height="12" defaultvalue="" tooltip="" tabstop="true" border="true" sip="true"/>
        <EDIT name="txtSUBSPECIES" x="41" y="79"
width="80" height="12" defaultvalue="" tooltip="" tabstop="true" border="true"
sip="true"/>
        <LABEL name="lblSpecies" x="11" y="50" width="29"
height="10" caption="Species" tooltip="" group="true" border="false"/>
        <LABEL name="lblvariety" x="15" y="65" width="24"
height="10" caption="Variety" tooltip="" group="true" border="false"/>
        <LABEL name="lblSub" y="80" width="40" height="10"
x="1" caption="Subspecies" tooltip="" group="true" border="false"/>
        <EDIT name="txtsort" x="41" y="94" width="80"
height="12" defaultvalue="" tooltip="" tabstop="true" border="true" sip="false"/>
        <LABEL name="lblsort" x="21" y="95" width="17"
height="9" caption="Sort" tooltip="" group="true" border="false"/>
    </PAGE>
</FORM>
</FORMS>
<SCRIPT src="Plots.vbs" language="VBScript"/>
<SYSTEMOBJECTS>
    <GPS onposition="Call updateGpsTable
" onaveragestop="collectData=False" onaveragestart="Call updateGpsTable"/>
</SYSTEMOBJECTS>
<FIND>
    <QUERYBUILDER>
        <QUERYFRAGMENT field="OBJECTID" operator="=" value="249"
fragmentoperator="AND"/>
    </QUERYBUILDER>
</FIND>
</LAYER>
</ArcPad>

```

## (VB Script)

```
option explicit
```

```
'Global Variables
```

```
Dim subPlotUid, plotUid, setupUid, projectUid, treeUid, spUid 'values used to link  
plots with subplots, and trees tables
```

```
Dim subplots, trees, setup, species, gps 'path to setup, subplot, and tree  
relational dbf tables
```

```
Dim dtProject, dtSetup, dtPlot, dtSubplot, dtTree, dtSpecies, dtGPS, dtGPSsetup  
'data tables used to update values
```

```
Dim luSpcd, luStcd, luCtype, luBrCd, luPct 'lookup dictionary used for comboboxes
```

```
Dim check, rcd, cruiseType, cruiseValue, plt, subplt, collectData, gpsPositionNumber
```

```
Sub Include(sInstFile)
```

```
    Dim s, oFSO
```

```
    Set oFSO = Application.CreateAppObject("file")
```

```
    On Error Resume Next
```

```
    If oFSO.Exists(sInstFile) Then
```

```
        'Application.MessageBox "file DataTable exists"
```

```
        oFSO.Open(sInstFile)
```

```
        do while not oFSO.EOF
```

```
            s = s & oFSO.ReadLine & vbCrLf
```

```
        loop
```

```
        oFSO.Close
```

```
        ExecuteGlobal s
```

```
    End If
```

```
    On Error Goto 0
```

```
    Set oFSO = Nothing
```

```
End Sub
```

```
function fillDictionary(path,cd,txt)
```

```
    Dim rcds, outDic, trcd, sql
```

```
    set outDic = new Dictionary
```

```
    set rcds = Application.CreateAppObject("RecordSet")
```

```
    sql = "[use] = 1 and [utype] < 3"
```

```
    rcds.Open path, 1
```

```
    trcd = rcds.Find(sql)
```

```
    do while trcd > 0
```

```
        outDic.Add rcds.Fields(cd).Value, rcds.Fields(txt).Value
```

```
        trcd = rcds.Find(sql,,trcd)
```

```
    loop
```

```
    rcds.Close
```

```

        set fillDictionary = outDic
        set rcds = nothing
        set outDic = nothing
End function

Function createGuid()
    createGuid = System.CreateGuid
End Function

'Application.MessageBox Application.Path & "\Applets\DataTable.vbs"
Include(Application.Path & "\Applets\DataTable.vbs")
'Application.MessageBox "Included DataTable"
set luSpcd = fillDictionary(Application.Path &
"\Applets\Tree_Data_Collection\spcd.dbf", "SPCD", "COMMON_NAM")
'Application.MessageBox "filled Dictionary Spcd"
set luStcd = fillDictionary(Application.Path &
"\Applets\Tree_Data_Collection\stcd.dbf", "CODE", "TEXT")
'Application.MessageBox "filled Dictionary Stcd"
set luCtype = fillDictionary(Application.Path &
"\Applets\Tree_Data_Collection\crcd.dbf", "CODE", "TEXT")
'Application.MessageBox "filled Dictionary crcd"
set luBrcd = fillDictionary(Application.Path &
"\Applets\Tree_Data_Collection\brcd.dbf", "CODE", "TEXT")
'Application.MessageBox "filled Dictionary Spcd"
set luPct = fillDictionary(Application.Path &
"\Applets\Tree_Data_Collection\pctcd.dbf", "CODE", "TEXT")

collectData=false
gpsPositionNumber = 0

Sub LoadFormStartup
'Called when edit form is opened 'If there are no shapefile layers in the map, then
exit
    call check_plot_layers 'sets the plotuid, setupuid, and projectuid
    if check = 3 then
        call setup_exists
        if check = 1 or plotUid = "" then
            if plotUid="" then
                Layer.Forms("EDITFORM").Close False
            end if
        else
            Layer.Forms("frmSetup").Show
        end if
    end if
end Sub

```

```

        end if
    else
        end if
End Sub

Function fillDataTable(frmName)
'Fills the data table based on uids. New values are created in the dbf file and
values are set in the data table
    Dim rcds, path, sql, outTbl, fcnt, fldArr, rcnt, fldName, fld, fldTypeArr,
guid
    sql = "[uid] = "
    set rcds = Application.CreateAppObject("recordset")
    Select Case frmName
        Case "frmSetup"
            path = setup
            rcds.Open path, 1
            sql = sql & """" & CStr(setupUid) & """"
        Case "frmSubplot"
            path = subplots
            rcds.Open path, 1
            sql = "[uid] = """" & subPlotUid & """"
        Case "frmtrees"
            path = trees
            rcds.Open path, 1
            sql = "[subplotUid] = """" & subPlotUid & """" AND [utype] < 3"
        Case "frmGPS"
            path = gps
            rcds.Open path, 1
            sql = "[plotUid] = """" & CStr(plotUid) & """" AND [utype] < 3"
        Case "frmSpecies"
            path = species
            rcds.Open path, 1
            sql = sql & """" & CStr(spUid) & """"
        Case else
    End Select
    set outTbl = new Table
    fcnt = rcds.Fields.Count
    outTbl.createTable 1, fcnt
    redim fldArr(fcnt-1)
    redim fldTypeArr(fcnt-1)
    fcnt = 0
    for each fld in rcds.Fields
        fldName = fld.Name

```

```

        fldArr(fcnt) = fldName
        fldTypeArr(fcnt) = fld.Type
        fcnt = fcnt + 1
Next
outTbl.Fields = fldArr
outTbl.FieldTypes = fldTypeArr
rcd = rcds.Find(sql)
if rcd = 0 then
    guid = createGuid()
    'rcds.AddNew
    outTbl.setCellValue 0,"uid",guid
    outTbl.setCellValue 0,"utype",1
    select case frmName
        Case "frmSubplot"
            outTbl.setCellValue 0,"plotUid",plotUid
            outTbl.setCellValue 0,"subplot",subplt
            outTbl.setCellValue 0,"pctpine",0
            outTbl.setCellValue 0,"pctbare",0
            outTbl.setCellValue 0,"burn",6
            outTbl.setCellValue 0,"pctherb",0
            outTbl.setCellValue 0,"pctbroad",0
            outTbl.setCellValue 0,"pctsaw",0
            outTbl.setCellValue 0,"pctcwd",0
            subPlotUid = guid
        Case "frmtrees"
            outTbl.setCellValue 0,"subplotUid",subPlotUid
            outTbl.setCellValue 0,"status",1
            outTbl.setCellValue 0,"sp",0
            outTbl.setCellValue 0,"cnt",1
            outTbl.setCellValue 0,"dbh",0
            treeUid = guid
        Case "frmSetup"
            outTbl.setCellValue 0,"projectUid",projectUid
            outTbl.setCellValue 0,"uid",setupUid
            'setupUid = guid
        Case "frmSpecies"
            outTbl.setCellValue 0,"use",1
            outTbl.setCellValue 0,"lbl","1 |0 |not set
|" & guid

            spUid = guid
        Case "frmGPS"
            outTbl.setCellValue 0,"plotUid",plotUid
        Case else

```

```

        end select
    else
        guid = rcds.Fields("uid").Value
        select case frmName
            Case "frmSubplot"
                subPlotUid = guid
            Case "frmtrees"
                treeUid = guid
            Case "frmSetup"
                setupUid = guid
                sql = sql & """" & CStr(setupUid) & """" AND [utype] < 3"
            Case "frmSpecies"
                spUid = guid
        end select
    end if
    rcnt = 0
    Do while rcd <> 0
        if rcnt = 0 then
            else
                'Application.MessageBox "adding Row rcd = " & Cstr(rcd) & ":" &
sql
                outTbl.addRow 1
            end if
            for each fld in rcds.Fields
                outTbl.setCellValue rcnt,fld.Name,fld.Value
            Next
            rcd = rcds.Find(sql, ,rcd)
            rcnt = rcnt + 1
        Loop
        set fillDataTable = outTbl
        rcds.Close
        set rcds = Nothing
    End Function

    sub setSubPlotUid(subPlotNumber)
        subplt = subPlotNumber
        subPlotUid = getUid(subplots,"[plotUid] = """" & plotUid & """" and [subplot] =
" & CStr(subplt),"uid")
    end sub

    sub createFile(flpath)
        dim myfile
        set myfile = Application.CreateAppObject("File")

```



```

    if myfile.Exists(flpath) then
    else
        myfile.Copy Application.Path & "\Applets\Tree_Data_Collection\" &
right(flpath,len(flpath)-InStrRev(flpath,"_")+1), flpath
    end if
    set myfile = nothing
end sub

```

```

Function getUid(path2, sql, fldName)
    Dim rcds2, outv1, weOpen
    set rcds2 = Application.CreateAppObject("recordset")
    weOpen = True
    createFile(setup)
    if (LCase(path2) = LCase(Layer.FilePath)) then
        weOpen = False
        set rcds2 = Layer.Records
    else
        rcds2.Open path2, 1
    end if
    outv1 = rcds2.Find(sql)
    if outv1 > 0 then
        outv1 = rcds2.Fields(fldName).Value
    else
        outv1 = -1
    end if
    if weOpen then
        rcds2.Close
    end if
    set rcds2 = nothing
    getUid = outv1
End Function

```

```

sub updateDbfValues(frmName,tbl)
'call this on ok button push
    Dim rcds, path, sql, uidIndex, uidValue, r, weOpen, fld
    call updateCurrentTableRecord(frmName,tbl)
    set rcds = Application.CreateAppObject("recordset")
    weOpen = true
    Select Case frmName
        Case "frmSetup"
            path = setup
        Case "frmSubplot"
            path = subplots
    End Select

```

```

    Case "frmtrees"
        path = trees
    Case "frmGPS"
        path = gps
        'Application.MessageBox "updated Current Record " & path
    Case "frmSpecies"
        path = species
    Case else
        exit sub
End Select
rcds.Open path, 2
uidIndex = tbl.findField("uid")
for r=0 to tbl.RowCount-1
    uidValue = tbl.getCellValue(r,uidIndex)
    sql = "[uid] = "" & uidValue & """"
    rcd = rcds.Find(sql)
    'Application.MessageBox Cstr(rcd)
    if rcd = 0 then
        'Application.MessageBox "adding record"
        rcds.AddNew
    end if
    for each fld in rcds.Fields
        'Application.MessageBox fld.Name & ": " &
CStr(tbl.getCellValue(r,fld.Name))
        fld.Value = tbl.getCellValue(r,fld.Name)
    Next
    rcds.Update
Next
if(weOpen) then
    rcds.Close
end if
set rcds = Nothing
end sub

sub updateCurrentTableRecord(frmName,tbl)
    Dim path, sql, uidValue, c, theControls, pg, cntName, cntType, cntValue,
    rwIndex
    Select Case frmName
        Case "frmSetup"
            pg = "pgSetup"
            uidValue = setupUid
        Case "frmSubplot"
            pg = "pgSubplot"

```

```

        path = subplots
        uidValue = subPlotUid
    Case "frmtrees"
        pg = "pgtrees"
        uidValue = treeUid
    Case "frmSpecies"
        pg = "pgSpecies"
        uidValue = spUid
    Case else
        'Application.MessageBox "Exit Current Record"
        exit sub
End Select
rwIndex = tbl.findRowIndex("uid",uidValue)
set theControls = Layer.Forms(frmName).Pages(pg).Controls
for each c in theControls
    cntName = lCase(Right(c.Name,len(c.Name)-3))
    cntType = LCase(c.Type)
    cntValue = c.Value
    if cntType = "edit" then
        cntValue = c.Value
        tbl.setCellValue rwIndex, cntName, cntValue
    elseif cntType = "combobox" then
        select case cntName
            Case "sp"
                cntValue = luSpcd.GetKey(c.Value)
                'Application.MessageBox "updating rowIndex (" &
Cstr(rwIndex) & ") sp = " & Cstr(cntValue)
            Case "status"
                cntValue = luStcd.GetKey(c.Value)
            Case "ctype"
                cntValue = luCtype.GetKey(c.Value)
            case "burn"
                cntValue = luBrcd.GetKey(c.Value)
            case else
                cntValue = luPct.GetKey(c.Value)
        end select
        tbl.setCellValue rwIndex, cntName, cntValue
    end if
Next
if(frmName = "frmSpecies") then
    Dim lblValue
    lblValue = formatStrValue(tbl.getCellValue(rwIndex,"use"),4) & "|" &
formatStrValue(tbl.getCellValue(rwIndex,"SPCD"),4) & "|" &

```

```

formatStrValue(tbl.getCellValue(rwIndex,"COMMON_NAM"),25) & "|" &
tbl.getCellValue(rwIndex,"uid")
tbl.setCellValue rwIndex, "lbl", lblValue
elseif(frmName = "EDITFORM") then
tbl.setCellValue rwIndex, "visited", 1
'Application.MessageBox Cstr(plotUid) & "visited = 1"
end if
call updateRow(tbl,rwIndex)
set theControls = Nothing
end sub

sub update_labels(form)
'Called on load event for forms or activate event on pages. Fills controls
with data table values
dim objTheForm, objTheControls, pg
if(form = "EDITFORM") then
pg = "pg" & form
elseif (form = "pgtreesview") then
pg = form
form = "frmtrees"
else
pg = "pg" & right(form,len(form)-3)
end if
Set objTheForm = Layer.Forms(form)
Set objTheControls = objTheForm.Pages(pg).Controls
select case pg
case "pgtrees"
objTheControls("lblsubplot").Value = "Plot " & CStr(plt) & "
Subplot " & CStr(subplt)
objTheControls("lblcount").Value = " of " &
CStr(dtTree.RowCount)
case "pgtreesview"
objTheControls("lblsubplot").Value = "Plot " & CStr(plt) & "
Subplot " & CStr(subplt)
case "pgSubplot"
objTheControls("lblsubplot").Value = "Plot " & CStr(plt) & "
Subplot " & CStr(subplt)
case "pgEDITFORM"
objTheForm.Pages(pg).Caption = "Plot " & CStr(plt)
case "pgSetup"
objTheControls("lblplot").Value = "Plot " & CStr(plt)
end select
set objTheForm = nothing

```

```

        set objTheControls = nothing
end sub

sub check_plot_layers
'Check to see if a record within a point layer is selected and if the point layer
has plotUid and setupUid. If so then gets the values associated with the selected
point
    trees = ""
    dim l, path, visited
    check = 0
    visited = 0
    dim lyr, rcds, flds, srcd
    set lyr = Map.SelectionLayer
    path = lyr.FilePath
    trees = left(path, len(path)-4) & "_trees.dbf"
    subplots = left(path, len(path)-4) & "_subplots.dbf"
    setup = left(path, len(path)-4) & "_setup.dbf"
    species = Application.Path & "\Applets\Tree_Data_Collection\spcd.dbf"
    gps = left(path, len(path)-4) & "_gps.dbf"
    set rcds = lyr.Records
    srcd = Map.SelectionBookmark
    rcds.Bookmark = srcd
    set flds = rcds.Fields
    for each l in flds
        if Ucase(l.Name) = "UID" or Ucase(l.Name) = "SETUPUID" or
Ucase(l.Name) = "OBJECTID" then
            check = check + 1
        elseif Ucase(l.Name) = "VISITED" then
            visited = 1
        else
            end if
    next
    if check = 3 then
        plotUid = flds.Item("uid").Value
        setupUid = flds.Item("setupUid").Value
        plt = flds.Item("OBJECTID").Value
        projectUid = getUid(setup, "[uid] = "" & setupUid & """,
"projectUid")
    else
        Application.MessageBox "Improperly Formated plot file. Please reload a
properly desinged plot file!", vbExclamation, "Improperly formated"
    end if
    set flds = nothing

```

```

        set rcds = nothing
        set lyr = nothing
    end sub

sub setup_exists
'checks to see if the related table setup exists. If so get the projectUid, cruise
type (BAF or Fixed) and cruise value (BAF value or fixed radius)
    dim myfile, myarray, i, setup_rcds, srcd, sql
    set myfile = Application.CreateAppObject("file")
    redim myarray(3)
    myarray(0) = trees
    myarray(1) = subplots
    myarray(2) = setup
    myarray(3) = gps
    for each i in myarray
        if myfile.Exists(i) then
        else
            myfile.Copy Application.Path & "\Applets\Tree_Data_Collection\"
& right(i,len(i)-InStrRev(i,"_")+1), i
        end if
    next
    set setup_rcds = Application.CreateAppObject("recordset")
    setup_rcds.Open setup, 1
    sql = "[uid]=" & CStr(setupUid) & ""
    srcd = setup_rcds.Find(sql)
    if srcd > 0 then
        check = 1
        cruiseType = setup_rcds.Fields("ctype").Value
        cruiseValue = setup_rcds.Fields("Value").Value
        projectUid = setup_rcds.Fields("projectUid").Value
    else
        check = 0
        cruiseType = "FIXED"
        cruiseValue = 4
    end if
    setup_rcds.Close
    set setup_rcds = nothing
    set myfile = nothing
    erase myarray
End sub

Sub setTreeCombobox
    dim controls,i, vl, tx

```

```

    set controls = Layer.Forms("frmTrees").Pages("pgTrees").Controls
    controls("cmbSp").Clear
    controls("cmbSp").AddItemsFromTable species,"COMMON_NAM","COMMON_NAM", "[use]
= 1 and [utype] < 3"
    controls("cmbStatus").Clear
    controls("cmbStatus").AddItemsFromTable Application.Path &
"\Applets\Tree_Data_Collection\stcd.dbf","TEXT","TEXT", "[use] = 1 and [utype] < 3" '
order [sort]"
    set controls = nothing
End Sub
Sub setCruiseCombobox
    dim cnt
    set cnt = Layer.Forms("frmSetup").Pages("pgSetup").Controls("cmbctype")
    cnt.Clear
    cnt.AddItemsFromTable Application.Path &
"\Applets\Tree_Data_Collection\crcd.dbf","TEXT","TEXT", "[use] = 1 and [utype] < 3"
    set cnt = nothing
End Sub

Sub setSubplotComboboxes
    dim cntrs, cnt
    set cntrs = Layer.Forms("frmSubplot").Pages("pgSubplot").Controls
    for each cnt in cntrs
        if LCase(cnt.Type) = "combobox" then
            cnt.Clear
            'Application.MessageBox cnt.Name
            select case cnt.Name
                Case "cmbburn"
                    cnt.AddItemsFromTable Application.Path &
"\Applets\Tree_Data_Collection\brcd.dbf","TEXT","TEXT", "[use] = 1 and [utype] < 3"
                    case else
                        cnt.AddItemsFromTable Application.Path &
"\Applets\Tree_Data_Collection\pctcd.dbf","TEXT","TEXT", "[use] = 1 and [utype] < 3"
            end select
        end if
    next
    set cnt = nothing
    set cntrs = nothing
End Sub

Sub updateFormValues(form, tbl)
'updates the form values based on field names and control names
    dim theControls, rwInd, c, uIndex

```

```

dim cntName, fldInd, cntType, form2, nv1
form2 = LCase(right(form,len(form)-3))
if form = "editform" then
    form2 = form
end if
set theControls =Layer.Forms("frm" & form2).Pages("pg" & form2).Controls
Select Case form2
    case "trees"
        uIndex = treeUid
    case "setup"
        uIndex = setupUid
    case "subplot"
        uIndex = subPlotUid
    case "project"
        uIndex = projectUid
    case "species"
        uindex = spUid
    case else
        exit sub
end select
rwInd = tbl.findRowIndex("uid",uIndex)
'Application.MessageBox Cstr(rwInd)
if rwInd < 0 then
    Application.MessageBox "Can't find related records. Setting to first
record."
    rwInd = 0
end if
for each c in theControls
    cntName = c.Name
    cntName = LCase(Right(cntName,len(cntName)-3))
    fldInd = tbl.findField(cntName)
    cntType = LCase(c.Type)
    if fldInd >-1 then
        if cntType = "edit" then
            c.Value = tbl.getCellValue(rwInd,fldInd)
        elseif cntType = "combobox" then
            select case cntName
                Case "sp"
                    'Application.MessageBox "table value = " &
Cstr(tbl.getCellValue(rwInd,fldInd))
                    nv1 =
luSpcd.GetValue(tbl.getCellValue(rwInd,fldInd))
                    c.Value = nv1

```



```

'Application.MessageBox Cstr("List count =
" & c.ListCount)

c.ListIndex =
luSpcd.findIndex(tbl.getCellValue(rwInd,fldInd))
'Application.MessageBox "lookup value = " &
CStr(luSpcd.Count) & " " & CStr(nv1) & " " & CStr(c.ListIndex)
Case "status"
nv1 =
luStcd.GetValue(tbl.getCellValue(rwInd,fldInd))
c.Value = nv1
c.ListIndex =
luStcd.findIndex(tbl.getCellValue(rwInd,fldInd))
Case "ctype"
nv1 =
luCtype.GetValue(tbl.getCellValue(rwInd,fldInd))
c.Value = nv1
c.ListIndex =
luCtype.findIndex(tbl.getCellValue(rwInd,fldInd))
Case "burn"
nv1 =
luBrcd.GetValue(tbl.getCellValue(rwInd,fldInd))
c.Value = nv1
c.ListIndex =
luBrcd.findIndex(tbl.getCellValue(rwInd,fldInd))
Case else
nv1 =
luPct.GetValue(tbl.getCellValue(rwInd,fldInd))
c.Value = nv1
c.ListIndex =
luPct.findIndex(tbl.getCellValue(rwInd,fldInd))
end select
end if
end if
Next
set theControls = nothing
call update_labels(form)
End Sub

Sub addTreeRow
dim outRwInd, guid, theControl,i
set theControl = ThisEvent.Object
call updateCurrentTableRecord("frmtrees",dtTree)
dtTree.AddRows(1)

```

```

guid = createGuid()
outRwInd = dtTree.RowCount - 1
dtTree.setCellValue outRwInd, "uid", guid
dtTree.setCellValue outRwInd, "utype", 1
dtTree.setCellValue outRwInd, "cnt", 1
dtTree.setCellValue outRwInd, "subplotUid", subplotUid
dtTree.setCellValue outRwInd, "status", 1
dtTree.setCellValue outRwInd, "sp", 0
treeUid = guid
call updateFormValues("frmtrees", dtTree)
theControl.Parent.Controls("lbltree").Value = CStr(dtTree.RowCount)
theControl.Parent.Controls("lblcount").Value = " of " & CStr(dtTree.RowCount)
set theControl = nothing
end Sub

```

```

Sub subtractTreeRow
    dim rcds, trcd, theControl, rwIndex
    set theControl = ThisEvent.Object
    rwIndex = dtTree.findRowIndex("uid", treeUid)
    if rwIndex = 0 then
        Application.MessageBox "Can't delete last record. If you want no tally
set tree count = 0"
        Exit Sub
    end if
    set rcds = Application.CreateAppObject("recordset")
    rcds.Open trees, 2
    trcd = rcds.Find("[uid] = "" & treeUid & """)
    if trcd > 0 then
        rcds.Fields("utype").Value = 3
        rcds.Update
    end if
    dtTree.subtractRow(rwIndex)
    rcds.Close
    set rcds = Nothing
    treeUid = dtTree.getCellValue((rwIndex-1), "uid")
    call updateFormValues("frmtrees", dtTree)
    theControl.Parent.Controls("lbltree").Value = CStr(rwIndex)
    theControl.Parent.Controls("lblcount").Value = " of " & CStr(dtTree.RowCount)
    set theControl = Nothing
End Sub

```

```

Sub updateRow(tbl, rwIndex)
    dim cvalue

```

```

        cvalue = tbl.getCellValue(rwIndex,"utype")
        if (cvalue > 0) and (cvalue < 4) then
        else
            tbl.setCellValue rwIndex, "utype", 2
        end if
    end Sub

sub move_record(spaces)
    dim theControl
    dim treenumber, maxrecords, rwIndex
    call updateCurrentTableRecord("frmtrees",dtTree)
    rwIndex = dtTree.findRowIndex("uid",treeUid)
    treenumber = rwIndex + spaces + 1
    maxrecords = dtTree.RowCount
    if treenumber > maxrecords then
        Application.MessageBox "You have reached the last record!" & vbnewline
& "If you want to add a new record press the '+' button"
        Exit Sub
    elseif treenumber < 1 then
        Application.MessageBox "You have reached the first record!" &
vbnewline & "If you want to delete a record press the '-' button"
        Exit Sub
    else
        treeUid = dtTree.getCellValue(rwIndex+spaces,"uid")
    end if
    Layer.Forms("frmtrees").Pages("pgtrees").Controls("lbltree").Value =
CStr(treenumber)
    call updateFormValues("frmtrees",dtTree)
end sub

sub view_select
    dim theViewControl
    dim form,treenumber
    set theViewControl = ThisEvent.Object
    if theViewControl.ListIndex > 0 then
        treenumber = theViewControl.ListIndex
        treeUid = dtTree.getCellValue(treenumber-1,"uid")
        theViewControl.Parent.Parent.Pages("pgtrees").Activate

        theViewControl.Parent.Parent.Pages("pgtrees").Controls("lbltree").Value =
treenumber

        call updateFormValues("frmtrees", dtTree)
        set theViewControl = nothing
    end if
end sub

```

```

        else
            set theViewControl = nothing
        Exit sub
    end if
end sub

sub update_list_view
    dim count, i, a, m, index_header, index_value
    dim theViewControls, myarray
    set theViewControls = Layer.Forms("frmTrees").Pages("pgTreesview").Controls
    call updateCurrentTableRecord("frmTrees",dtTree)
    index_header = "# | SP| DBH| STA| CNT|UTYP| UID
| SUBPLOTUID "
    theViewControls ("lstTreesview").Clear
    theViewControls ("lstTreesview").AddItem index_header,index_header
    for i=0 to dtTree.RowCount-1
        redim myarray(5)
        myarray(0) = CStr(i + 1)
        myarray(1) = CStr(dtTree.getCellValue(i,"sp"))
        myarray(2) = CStr(dtTree.getCellValue(i,"dbh"))
        myarray(3) = CStr(dtTree.getCellValue(i,"status"))
        myarray(4) = CStr(dtTree.getCellValue(i,"cnt"))
        myarray(5) = CStr(dtTree.getCellValue(i,"utype"))
        count = 0
        for each m in myarray
            if len(m) < 4 then
                for a = 1 to (4-len(m))
                    m = m & " "
                next
            end if
            myarray(count) = m
            count = count + 1
        next
        index_value = join(myarray,"|") & "|" & dtTree.getCellValue(i,"uid") &
        "|" & dtTree.getCellValue(i,"subplotUID")
        theViewControls("lstTreesview").AddItem index_value, index_value
    next
    Erase myarray
    set theViewcontrols = nothing
end sub

function formatStrValue(value,lng)
    Dim outvl,cLng

```

```

lng = lng + 1
clng = len(CStr(value))
outv1 = CStr(value)
if clng < lng then
    outv1 = outv1 & space(lng-clng)
elseif clng > lng then
    outv1 = left(outv1,lng)
end if
formatStrValue = outv1
end function

sub fillListViewAddTree
    dim control,i, vl, tx
    set control =
Layer.Forms("frmAddTree").Pages("pgAddTree").Controls("lstTree")
    control.Clear
    control.AddItemFromTable species,"lbl","lbl", "[utype] < 3"
    set control = nothing
end sub

sub selectAddTree
    dim rcds, theControl, ind, use, ln, ovl,ary, svl,code,common,ut
    set theControl = ThisEvent.Object.Parent("lstTree")
    ind = theControl.ListIndex
    ln = theControl.Value
    ary = split(ln,"|")
    spUId = ary(ubound(ary))
    use = CInt(ary(0))
    code = CInt(ary(1))
    common = Trim(ary(2))
    if(use=1) then
        ovl=0
    else
        ovl = 1
    end if
    svl = formatStrValue(ovl,len(ary(0))-1)
    ary(0) = svl
    ln = join(ary,"|")
    set rcds = Application.CreateAppObject("recordset")
    rcds.Open species, 2
    rcds.Find "[uid] = "" & spUId & """"
    rcds.Fields("use").Value = ovl
    rcds.Fields("lbl").Value = ln

```

```

    ut = Cint(rcds.Fields("utype").Value)
    if(ut = 1 or ut = 3) then
    else
        rcds.Fields("utype").Value = 2
    end if
    rcds.Update
    rcds.Close
    fillListViewAddTree
    theControl.ListIndex = ind
    set rcds = nothing
    set theControl = nothing
    Erase ary
end sub

sub removeAddTree
    dim theControl, rcds, sql, uid, ind, ary
    set theControl = ThisEvent.Object.Parent("lstTree")
    set rcds = Application.CreateAppObject("recordset")
    ind = theControl.ListIndex
    if ind >= (theControl.ListCount - 1) then
        ind = ind-1
    elseif ind = 0 then
        ind = -1
    end if
    ary = split(theControl.Value, "|")
    uid = ary(3)
    sql = "[uid] = "" & uid & """"
    rcds.Open species, 2
    rcds.Find sql
    rcds.Fields("utype").Value = 3
    rcds.Fields("use").Value = 0
    rcds.Update
    rcds.Close
    fillListViewAddTree
    theControl.ListIndex = ind
    set rcds = nothing
    set theControl = nothing
    Erase ary
end sub

sub takePicture
    dim path, fl, rslt
    path = replace(Layer.FilePath, Layer.Name & ".shp", "") & "PlotPics"
    set fl = Application.CreateAppObject("file")

```

```

if(not fl.Exists(path)) then
    'Application.MessageBox "creating directory" & path
    fl.CreateDirectory(path)
end if
path = path & "\" & plotUid & ".jpg"
if(fl.Exists(path)) then
    rslt = Application.MessageBox("File exist! Do you want to replace?",
4, "File Exists")
    if( rslt = 6) then
        'Application.MessageBox "taking picture"
        MultiMedia.CaptureStill(path)
    else
        'show image
    end if
else
    MultiMedia.CaptureStill(path)
end if
set fl = nothing
end sub

sub collectGPS
    Dim mpos, i, rCnt, dif, rIndex, weOpen, ln, optArr
    dim view
    gpsPositionNumber = 0
    optArr = getAveragingOptions
    set dtGPS = fillDataTable("frmGPS")
    mpos = optArr(0)
    if(optArr(2)=False or mpos < 6) then
        Application.MessageBox "You must enable averaging and have more than 5
positions!", vbInformation, "GPS setup"
        Application.ExecuteCommand("gpsoptions")
        exit sub
    end if
    rCnt = dtGPS.RowCount
    'Application.MessageBox "Row count = " & Cstr(rCnt)
    dif = mpos-rCnt
    'Application.MessageBox "Diff = " & Cstr(dif)
    if dif > 0 then
        call dtGPS.AddRows(dif)
        'Application.MessageBox "Added " & Cstr(dif) & " new row count = " &
Cstr(dtGPS.RowCount)
        for i = rCnt to mpos-1
            rIndex = i

```

```

        dtGPS.setCellValue rIndex, "uid", createGuid()
        dtGPS.setCellValue rIndex, "plotUid", plotUid
        dtGPS.setCellValue rIndex, "utype", 1
        dtGPS.setCellValue rIndex, "position", i
        'Application.MessageBox "set value for rindex " & cStr(rIndex)
    next

    view = false
    elseif dif < 0 then
        if Application.MessageBox("Positions exist. Do you want to
replace?",4,"Replace") = 6 then
            view = false
        else
            view = true
        end if
        for i = mpos to rCnt-1
            rIndex = i
            'Application.MessageBox "Setting rowindex " & Cstr(rIndex) & "
to 3"

            dtGPS.setCellValue rIndex, "utype", 3
        next
    else
        if Application.MessageBox("Positions exist. Do you want to
replace?",4,"Replace") = 6 then
            view = false
        else
            view = true
        end if
    end if
    if(view) then
    else
        weOpen = false
        if(Application.GPS.IsOpen) then
        else
            weOpen = True
            Application.GPS.Open
            Application.MessageBox "Activated GPS",vbInformation,
"Activate"

        end if
        collectData = True
        Application.ExecuteCommand("movepointtogps")
        collectData = False
        if(weOpen) then

```



```

        if Application.MessageBox("Do you want to deactivate the GPS?",
vbYesNo, "Deactivate") = 6 then
            Application.GPS.Close
        end if
    end if
end if
call removeGpsZeros
call updateDbfValues("frmGPS",dtGPS)
end sub

```

```

sub removeGpsZeros
    Dim r, xvalue, sdel, xindex
    xindex = dtGPS.findField("x")
    sdel = -1
    for r=0 to dtGPS.RowCount-1
        xvalue = dtGPS.getCellValue(r,xindex)
        if xvalue=0 then
            sdel = r
            exit for
        end if
    next
    if sdel>-1 then
        for r=dtGPS.RowCount-1 to sdel Step -1
            dtGPS.SubtractRow(r)
        next
    end if
end sub

```

```

sub setupGpsForm
    Dim theControl, theLabel, i, rCnt, ln, xp, yp, h, s, sh
    set theControl = Layer.Forms("frmGPS").Pages("pgGPS").Controls("lstgps")
    set theLabel = Layer.Forms("frmGPS").Pages("pgGPS").Controls("lblgps")
    rCnt = dtGPS.RowCount
    theControl.Clear
    'Application.MessageBox "Row count = " & Cstr(rCnt)
    for i = 1 to dtGPS.RowCount
        sh = CInt(dtGPS.getCellValue(i-1,"utype"))
        if sh < 3 then
            xp = dtGPS.getCellValue(i-1,"x")
            yp = dtGPS.getCellValue(i-1,"y")
            h = dtGPS.getCellValue(i-1,"hdop")
            s = dtGPS.getCellValue(i-1,"sat")

```

```

        ln = formatStrValue(i,3) & "|" & formatStrValue(xp,10) & "|" &
formatStrValue(y,10) & "|" & formatStrValue(h,5) & "|" & formatStrValue(s,3)
        theControl.AddItem ln, ln
    end if
next
theLabel.Value = "Viewing Data"
set theControl = nothing
set theLabel = nothing
end sub

function getGpsValues
    dim myArray
    redim myArray(3)
    myArray(0) = Cdbl(Application.GPS.Properties("HDOP"))
    myArray(1) = Cint(Application.GPS.Satellites.Count)
    myArray(2) = Cdbl(Application.GPS.X)
    myArray(3) = Cdbl(Application.GPS.Y)
    getGpsValues = myArray
end function

sub updateGpsTable
    dim gpsArray, p
    if(collectData) then
        if gpsPositionNumber > dtGPS.RowCount-1 then
            gpsPositionNumber = 0
        end if
        gpsArray = getGpsValues
        p = gpsPositionNumber + 1
        dtGPS.setCellValue gpsPositionNumber, "x", gpsArray(2)
        dtGPS.setCellValue gpsPositionNumber, "y", gpsArray(3)
        dtGPS.setCellValue gpsPositionNumber, "position", p
        dtGPS.setCellValue gpsPositionNumber, "UTC",
Application.GPS.Properties("UTC")
        dtGPS.setCellValue gpsPositionNumber, "hdop", gpsArray(0)
        dtGPS.setCellValue gpsPositionNumber, "sat", gpsArray(1)
        gpsPositionNumber = p
    end if
end sub

function getAveragingOptions 'returns (points,verticies,enabled)
    dim f1, ln, pt, mArr, outArr, ar
    Redim outArr(2)
    outArr(0) = 0

```

```

outArr(1) = 0
outArr(2) = false
pt = Application.System.Properties("PersonalFolder") & "\\My
ArcPad\ArcPadPrefs.apx"
'Application.MessageBox pt
Set fl = Application.CreateAppObject("file")
if fl.Exists(pt) then
    fl.Open pt, apFileRead
    Do while not fl.EOF
        ln = Trim(fl.ReadLine)
        'Application.MessageBox ln
        if(InStr(1,ln,"<AVERAGING">0) > 0) then
            '<AVERAGING point="30" vertex="5" enabled="true"/>'
            mArr = Split(ln, " ")
            if ubound(mArr) > 0 then
                for each ar in mArr
                    ar = Replace(ar,"/>","")
                    if InStr(1,ar,"point") > 0 then
                        outArr(0) =
Cint(Replace(Split(ar,"=")(1),"",""))
                    elseif InStr(1,ar,"vertex") > 0 then
                        outArr(1) =
Cint(Replace(Split(ar,"=")(1),"",""))
                    elseif InStr(1,ar,"enabled") > 0 then
                        outArr(2) =
Cbool(Mid(Split(ar,"=")(1),2,4))
                    end if
                next
            end if
            erase mArr
        exit do
    end if
loop
fl.Close
else
    Application.MessageBox "Can't find file"
end if
getAveragingOptions = outArr
set fl = nothing
end function

sub setVisitedZero
    Dim rcds

```

```

set rcds = Layer.Records
rcds.Bookmark = Map.SelectionBookmark
rcds.Fields("visited").Value = 0
rcds.Fields("utype").Value = 0
rcds.Update
set rcds = nothing
end sub

sub plotCheck
Dim rcds, rcds2, sql, trcd, cnt, suid, sql2, slst, path, fl, xvalue
set rcds = Application.CreateAppObject("RecordSet")
set rcds2 = Application.CreateAppObject("RecordSet")
sql = "[plotUid] = "" & plotUid & "" and [utype] < 3"
cnt = 0
rcds.Open gps, 1
trcd = rcds.Find(sql)
do while trcd>0
    xvalue = rcds.Fields("x").Value
    if xvalue <> 0 then
        cnt = cnt+1
    end if
    trcd = rcds.Find(sql,,trcd)
    if(cnt>5)then
        exit do
    end if
loop
rcds.Close
if(cnt < 5)then
    Application.MessageBox "At least 5 GPS positions have not been
collected! Setting visited to zero!", vbInformation
    call setVisitedZero
    exit sub
end if
cnt = 0
redim slst(3)
rcds.Open subplots, 1
trcd = rcds.Find(sql)
do while trcd>0
    suid = rcds.Fields("uid").Value
    sql2 = "[subplotUid] = "" & suid & "" and [utype] < 3"
    rcds2.Open trees, 1
    if rcds2.Find(sql2) = 0 then

```

```

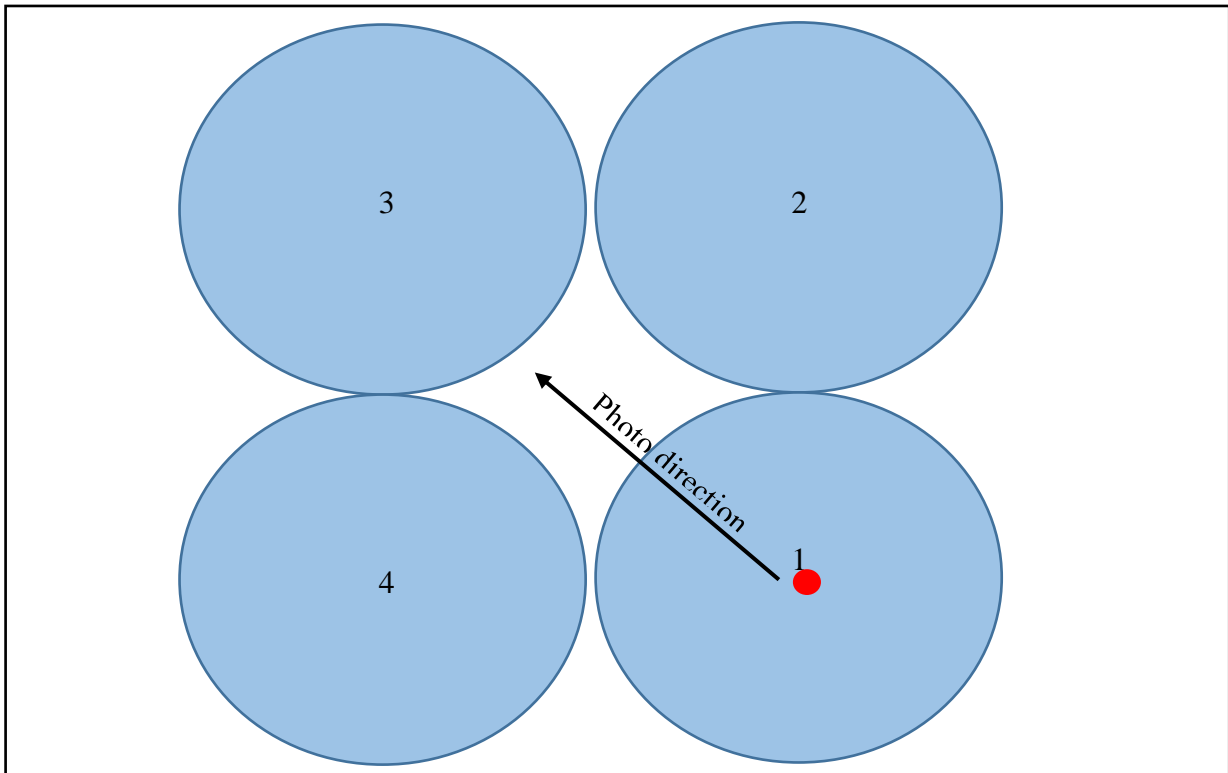
        Application.MessageBox "Subplot = " &
Cstr(rcds.Fields("subplot")) & " trees have not been collected! Setting visited to
zero!", vbInformation
        rcds.Close
        rcds2.Close
        call setVisitedZero
        exit sub
    end if
    slst(cnt) = rcds.Fields("subplot").Value
    cnt = cnt+1
    trcd = rcds.Find(sql,,trcd)
    if(cnt>=4)then
        exit do
    end if
loop
rcds.Close
if(cnt < 4)then
    Application.MessageBox "Subplot " & getSubPlotValue(slst) & " has not
been collected! Setting visited to zero!", vbInformation
    call setVisitedZero
    exit sub
end if
path = replace(Layer.FilePath,Layer.Name&".shp","",) & "PlotPics\" & plotUid &
".jpg"
set fl = Application.CreateAppObject("file")
if(not fl.Exists(path)) then
    Application.MessageBox "Picture has not been collected! Setting
visited to zero!", vbinformation
    call setVisitedZero
    exit sub
end if
set rcds = nothing
set rcds2 = nothing
end sub

function getSubPlotValue(mArr)
    dim outStr, i
    outStr = "1234"
    for each i in mArr
        outStr = Replace(outStr, Cstr(i),"")
    Next
    outStr = left(outStr,1)
    getSubPlotValue = outStr

```

```
end function
```

## **Field Plot Protocol**



**Figure 1. Plot/subplot configuration**

This project is trying to estimate forest structure and conditions throughout most of the eastern panhandle of Florida using field plots that represent the entire range of forested and non-forested conditions. FNAI is working with the US Forest Service to collect the field data needed to calibrate mathematical models that will be used for estimations.

We would like to visit location(s) on your property to record vegetation cover, tree species, and diameters within a small area (120 feet x 120 feet). This will probably take less than an hour and we will only collect observation data on forest structure. No markings (flags, etc.) will be left in the forest.

#### Field data to collect

Each plot is made up of 4 subplots (Figure 1). Plots should be collected in a counter clockwise manner starting at subplot 1. At subplot 1, GPS position and a plot photo will be taken as described below. Subplot percent cover will be estimated from an aerial perspective above the canopy as if the field technician was looking down from an airplane. All tree data should be measured and input into the tree data collection forms.

Plots that fall in nonforested areas and are obviously not tally plots such as a soccer field, baseball diamond, and water can be filled in using remotely sensed data such as the NAIP aerial photography. These plots do not need updated GPS locations, and should be collected as no tally subplots. Note, due to not collecting a GPS location or picture, the plot will not change to a red color on data recorders.

Tree combo box lists within the Tree data component can be updated within the application. However, this can be cumbersome and time consuming. Sorting and updating can be done more



efficiently using ArcDesktop and its sorting capabilities on the spcd.dbf table (see data forms section for more information).

Data managed will be an important component to collecting the field data. There are multiple ways to manage the data and they can vary depending on the number of technicians collecting data. One effective approach is to have each group using a data recording device to work in separate geographic areas and download their work from the data recorder to a directory named for that day on a laptop computer each day after returning to the field. In this case the entire project directory on the mobile device is copied to the laptop and the mobile device can be used the next day without any changes. If the mobile device starts to become slow due to the amount of data collected, field crews can remove the plot\_trees.dbf table and the plot\_gps.dbf table from the mobile device after they have moved those files to the laptop. This will effectively remove all the records from those two tables and will reduce the number of records that the mobile device needs to sort through. **Note all previous records need to be kept on the laptop to be merged at a later date.**

Below is a hierarchical outline of what needs to be collected at each plot.

- Plot (collect all percent cover and tree data for each subplot)
  - At subplot 1
    - Collect GPS positions
      - Ideally 20 positions (Averaging)
      - Hdop < 5
      - 3D mode
      - DGPS if possible
    - Take a Picture facing NW across the subplots
  - Subplot measurements
    - Percent cover is estimated from an aerial perspective above the tree canopy
    - Last Burn: (years since last burn, >6)
    - % CWD: (% duff { dead broad leaf and pine needles } and coarse woody debris cover in subplot; 0-100)
    - % Herb: (% herbaceous cover in subplot; 0-100)
    - % Saw: (% palmetto cover in subplot; 0-100)
    - % Broad: (% broad leaf cover in subplot; 0-100)
    - % Bare: (% mineral soil cover in subplot; 0-100)
    - % Pine: (% pine needle cover in subplot; 0-100)
      - Tree measurements  $\geq 2''$  at dbh
        - SP: (Species; drop down)
        - DBH: (DBH in inches; 2-100)
        - Status: (Status; drop down)
        - Count: (Number of trees that meet the above condition; 1-100)
      - Tree measurements  $< 2''$ 
        - SP: (Species; drop down)

- DBH: (DBH in inches; 2-100)
- Status: (Status; drop down)
- Count: (Number of trees that meet the above condition; 1-100)
- Subplot no tally
  - One tree, unknown species, DHB=0, Count=0
  - Status: (Status value=No Tally)

### Plot layout (Figure 1)

- Extent = 36m by 36m
- 4 subplots
  - Subplot radius = 9m

### Plot locations

- Based on Hogland plot allocation technique described in chapter 4 of this dissertation
- Navigate to each plot coordinate
- Each plot's coordinate represent center of subplot 1

### Data entry forms

All coding and tools work within ArcPad. Changes made to GPS and camera preferences impact how GPS data and images are collected within the forms. Related tables and images are stored in the same directory as the plot layer. Form designs (.apl) and vbscript files (.vbs) are stored in the same directory as the plot layer. Within the plot's .apl file there are 7 different forms and 3 built in dialogs (picture, move point, and GPS preferences). All forms allow users to change data without instantly updating records in a given table. Tapping the green OK button will make changes to the underlying data tables. Tapping the red cancel button or an exit button (X) allows a user to back out of a given form without changing the underlying data. Below is a short description of each form.

- Identify: used to identify plot information

IDENTIFY

IDENTIFY

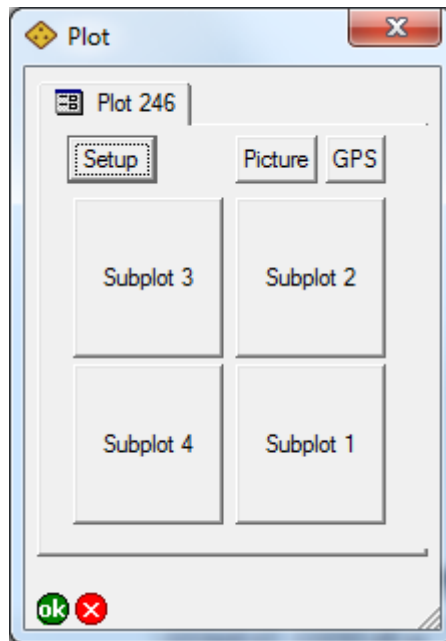
Plot 101

Visited 1

UID {36FE331E-57DB-4286

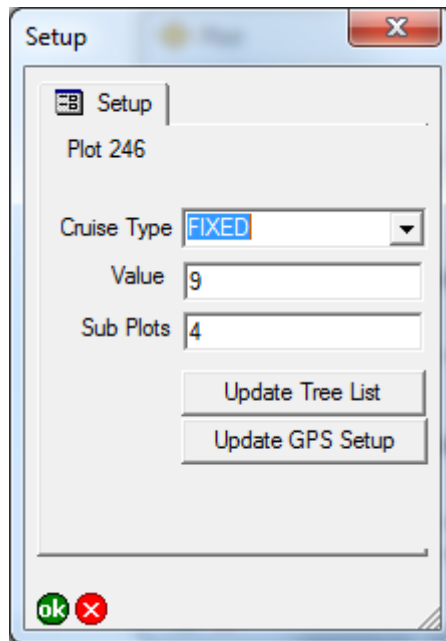
The Identify form can be accessed using ArcPad's Identify button and used to display the Plot's ObjectID, whether the plot has been visited (1=yes, 0=no), and the unique identifier (UID)

- Plot: used to collect plot information



The plot form is accessed by clicking the Edit Feature Properties button within ArcPad (note you must be editing and have a plot selected to use this button in ArcPad). This form is used to access the project setup, collect and update GPS position, capture a picture of the vegetative condition, and navigate between subplots. GPS positions and pictures should only be collected when standing at the center of Subplot 1 (existing plot locations identify the center of each plot's subplot 1). To access the project Setup form tap the Setup button. To capture a picture of the vegetative condition tap the Picture button. To collect GPS position tap the GPS button. To open a given Subplot form tap the appropriate subplot button. Tapping the green OK button will trigger checks to see if all the data have been collected. If criteria are met, the visited field within the plots layer will change from 0 to 1 and the plot display will change from yellow to red. If criteria are not met or the cancel button is tapped the plot visited value and display will not be changed.

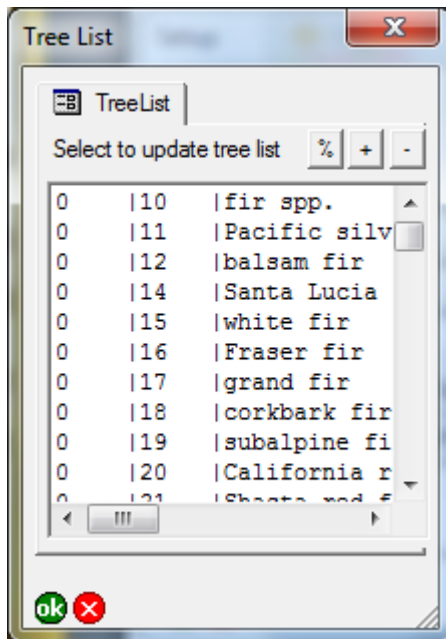
- Setup: used to define cruise parameters



The screenshot shows a mobile application window titled "Setup". At the top left, there is a menu icon and the word "Setup". Below that, it says "Plot 246". There are three input fields: "Cruise Type" is a dropdown menu with "FIXED" selected, "Value" is a text box containing "9", and "Sub Plots" is a text box containing "4". Below these fields are two buttons: "Update Tree List" and "Update GPS Setup". At the bottom left of the window, there are two buttons: a green "ok" button and a red "X" button.

The Setup form is accessed through the Plot form by tapping on the Setup button in the Plot form. This form is used to define the project type and project specifications. In addition, users can update which trees are available in the species drop down of the Tree form (Update Tree List) and set GPS preferences (Update GPS Setup). To store changes to the project setup, a user must tap the green OK button. To undo changes made in the form, users can tap the red X button.

- Update Tree List: used to add, subtract, and select which trees are visible in the Trees species dropdown



The Update Tree List form is accessed through the Setup form by tapping the Update Tree List button and is used to add, subtract, and select which tree species are available in the Trees Species (SP) combo box. Tree species available were extracted from the USFS FIA database. There are many tree species to choose from and selecting common species will make finding the correct species within the Trees SP combo box easier and quicker. Currently the species selected (value of 1 in the first column) were identified based on the species found within the FIA plots located within the Florida SGAs. When the form opens all available tree species, species codes, common names, unique identifiers, and whether the species is currently available in the Trees SP combo box (first column value of 1 or 0: yes or no) are shown in the form. To toggle on or off a tree species in the Trees SP combo box select a given species and tap the % button. This will change the first column of data from 0 to 1 or vice versa. Species that have a value of 1 are available in the Trees SP combo box. If a new species needs to be added to the potential species, users can click the + button which will open the Species form. If a species needs to be removed from the potential list, users can select that species and click the – button. To store changes to the potential tree species a user must tap the green OK button. To undo changes made in the form, users can tap the red X button. Note, to change the sort order of the Trees SP combo box users can use the sort field within the ~Applets\Tree\_Data\_Collection\spcd.dbf file and ArcMap’s sort function to replace the existing spcd.dbf file. The order of the data within the spcd.dbf file (determined by the row) determines the order in of the Trees SP combo box.

- Species: used to update potential tree species in the tree species list

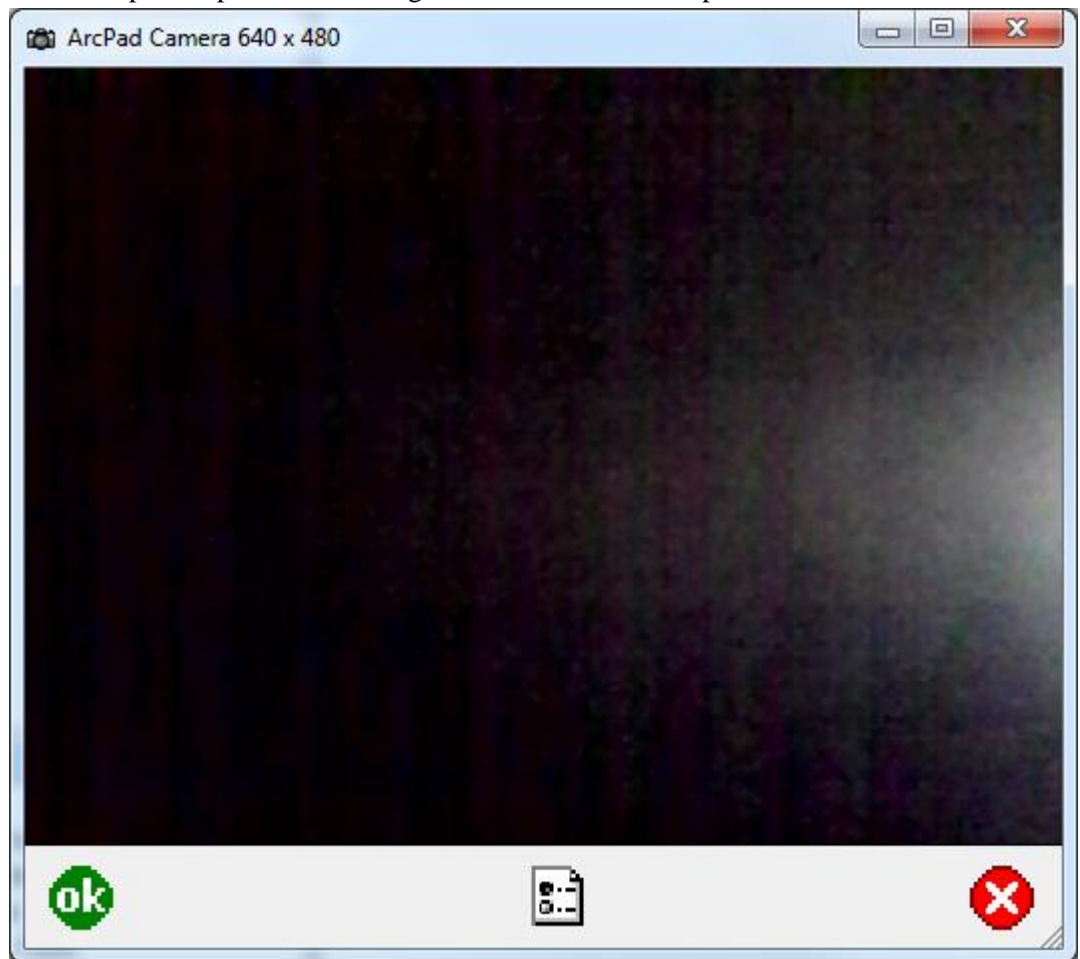
The Species form is accessed through the Tree List form by tapping on the + button. This form is used to add a new species to the potential species list. All species must have a Code (numeric value; 0-9999) and Common name specified (text). Ideally all fields would be given valid values but Code and Common name are the only two required fields. Be sure to use a unique code for each new tree species. To store changes to the potential tree species a user must tap the green OK button. To undo changes made in the form, users can tap the red X button.

- Update GPS Setup: used to update GPS preferences

The GPS Preferences dialog is accessed through ArcPad's GPS Preferences button or the Update GPS Setup button in the Setup form. This dialog can be used to modify GPS preferences as described in ArcPad's documentation. Make sure Enable Averaging is checked and that the number of positions to average is set to

20 on the Capture tab of the dialog (number of positions must be greater than 5 to collect GPS data). In the Quality tab of the dialog box make sure maximum HDOP is set to 5 and 3D Mode Only is checked (note you may want to turn off Alerts). To store changes to GPS Preferences a user must tap the green OK button. To undo changes made in the form, users can tap the red X button. Note, Arcpad and other software can have conflicts between one another based on comport settings. Similarly, external devices such as GPS receivers should be configured to always stay on and should be connected to Arcpad as described in Arcpad's documentation.

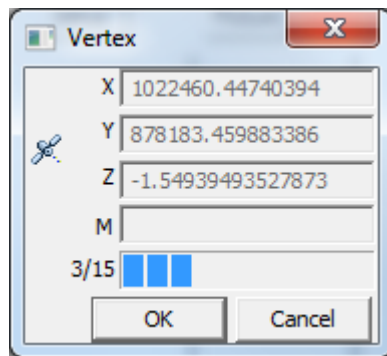
- Picture: used to capture a picture of the vegetative condition of the plot



The Capture Picture dialog is accessed through the Plot form by tapping the Picture button. This dialog can be used to capture a picture as described in ArcPad's documentation. All pictures will be stored in a subdirectory located within the same directory as the plots layer named PlotPics. Each image recorded will be named after the plot's unique identifier (GUID).

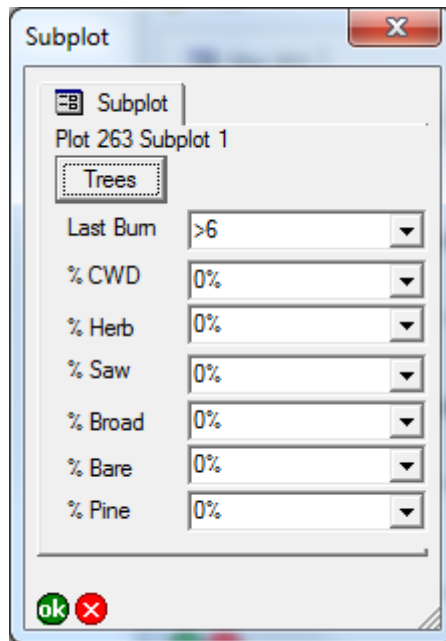


- GPS: used to collect and store plot GPS positions



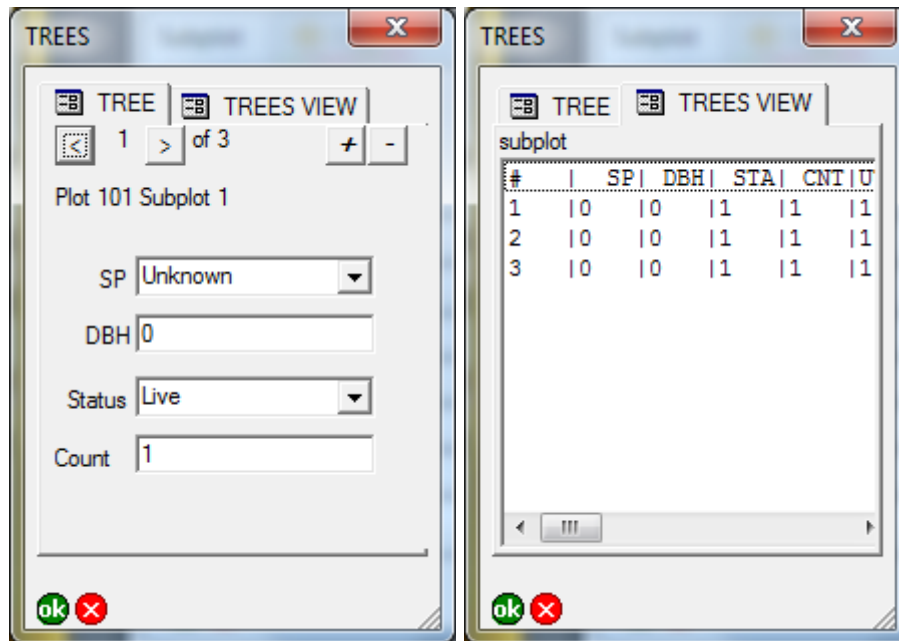
Upon tapping the GPS button a check will be performed to see if GPS positions already exist for the selected plot. If positions exist you will be prompted as to whether you want to replace existing positions or not. If you want to replace existing position or position have not been collected for that plot a check will be performed to see if the GPS is active. If the GPS is not active, it will be activated and a message box will appear letting you know that it has been activated. At this point the Move Point dialog (Vertex) will appear and begin collecting GPS positions based on the specified GPS Preferences. After collecting GPS positions the dialog will close. If you activated the GPS through the form, you will be prompted if you want to keep the GPS active.

- Subplot: used to collect subplot data

The image shows a screenshot of a mobile application window titled "Subplot". At the top right of the window is a red "X" button. Below the title bar, there is a tab labeled "Subplot" and a text field containing "Plot 263 Subplot 1". Underneath, there is a button labeled "Trees". Below this are seven dropdown menus, each with a label and a value: "Last Burn" with ">6", "% CWD" with "0%", "% Herb" with "0%", "% Saw" with "0%", "% Broad" with "0%", "% Bare" with "0%", and "% Pine" with "0%". At the bottom left of the window are two buttons: a green "ok" button and a red "X" button.

The Subplot form can be accessed through the Plot form by tapping on of the Subplot buttons. Tapping one of the buttons will open the Subplot form. Within the form users can select the number of years since last burn, % CWD cover, % Herbaceous cover, % Palmetto cover, % Broadleaf shrub cover, % Pine shrub cover. To store subplot data a user must tap the green OK button. To undo changes made in the form, users can tap the red X button. Note, tapping the red x button before a record has been saved will result in losing related tree values (it is safer to tap the green ok button and then reopen the subplot and change the values then to click the red x button before the green ok button has ever been tapped).

- Tree: used to collect tree data



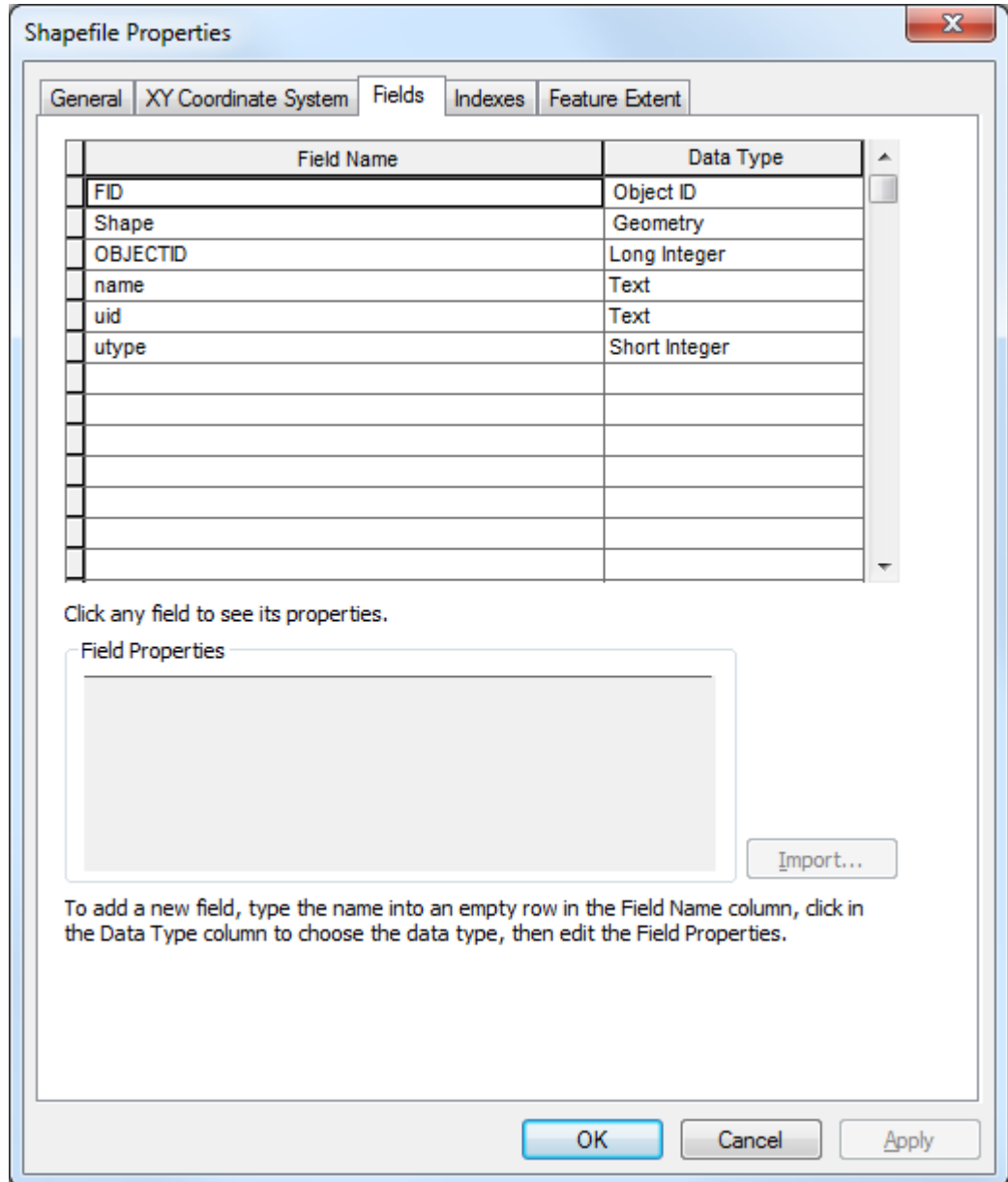
The Trees form can be accessed through the subplot form by tapping on the Trees button. Tapping the Tress button will open the Trees form which has two tabs; 1) Tree and 2) Trees View. The Tree tab allow the user to add, subtract, and navigate through the subplot’s tree list. Required values for a given tree are species (SP), DBH for trees greater than 2” in diameter (less than 2” does not need a DBH measured), Status, and Number of trees (count). To denote a no tally subplot, species must be set to Unknown, DBH must be 0, Status must be No Tally, and count must be set to 0. Values in the SP combo box can be modified using the Update Tree List form. The Trees View can be used to view all trees list data for a subplot. Click on a given record within the tree list will navigate the user to that given tree. To add a tree to a subplot’s tree list use the + button. To remove a tree from the tree list use the – button. To move to the next tree in the tree list use the > button. To move to the previous tree in the tree list use the < button. To store tree data a user must tap the green OK button. To undo changes made in the form, users can tap the red X button.

### Database Schema

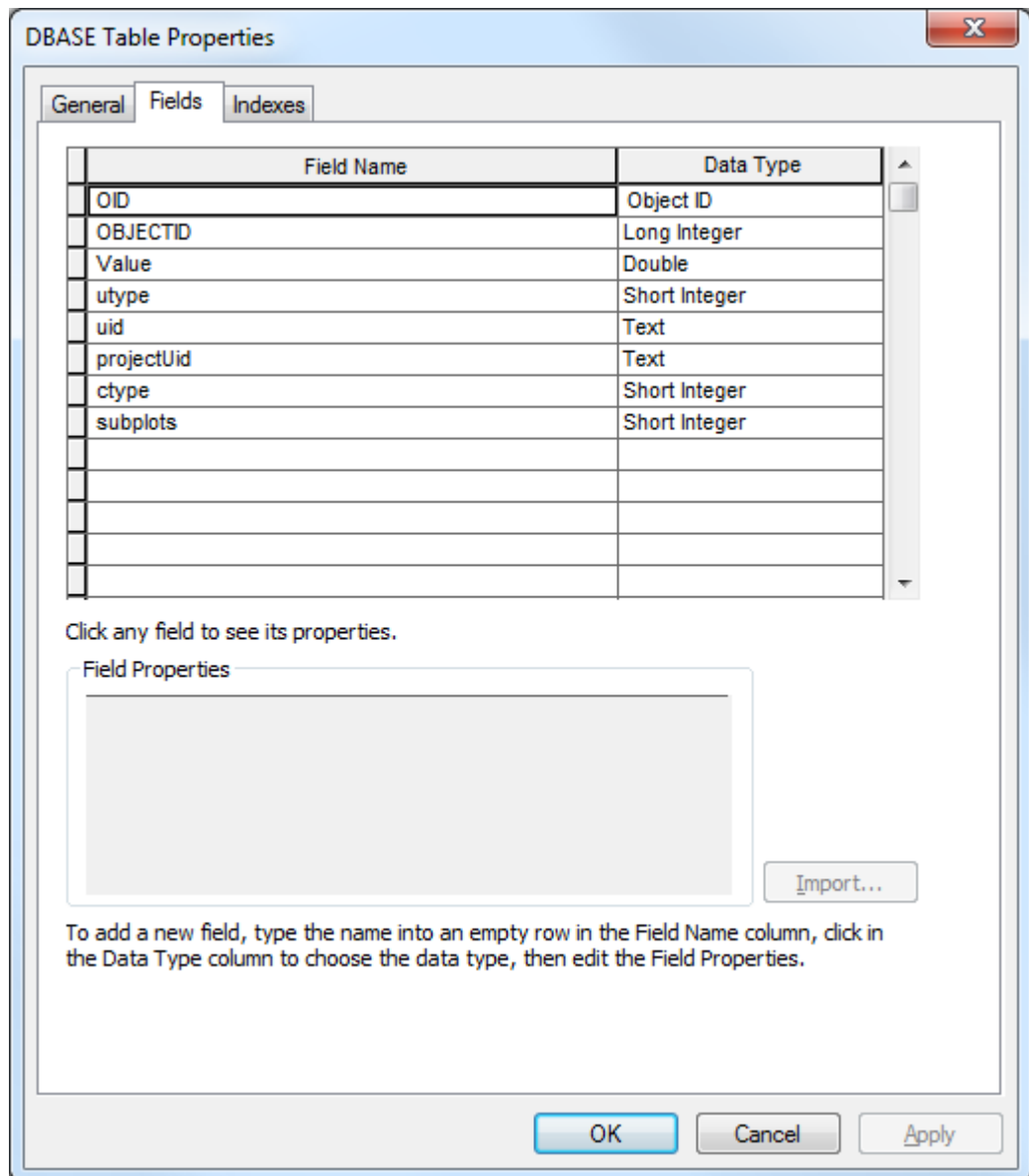
The data tables in the plots tool consist of: project.dbf, plots.dbf, plots\_gps.dbf, plots\_setup.dbf, plots\_subplots.dbf, and plot\_trees.dbf. Supporting data tables located in ~\Applets\Tree\_Data\_Collection are: \_gps.dbf, \_setup.dbf, \_subplot.dbf, \_trees.dbf, crcd.dbf, spcd.dbf, and stcd.dbf. All dbf tables have a unique identifier field (UID:string), an update type field (utype:integer;1=insert,2=update,3=delete). All children dbf tables have a link field to their parent named after the parent layer (e.g., subplotUID:string). UIDs are globally unique and are stored as strings. The hierarchical relationship between tables is described below

- project.dbf UID
  - plots\_setup.db -> projectUID
    - plots.dbf -> setupUID

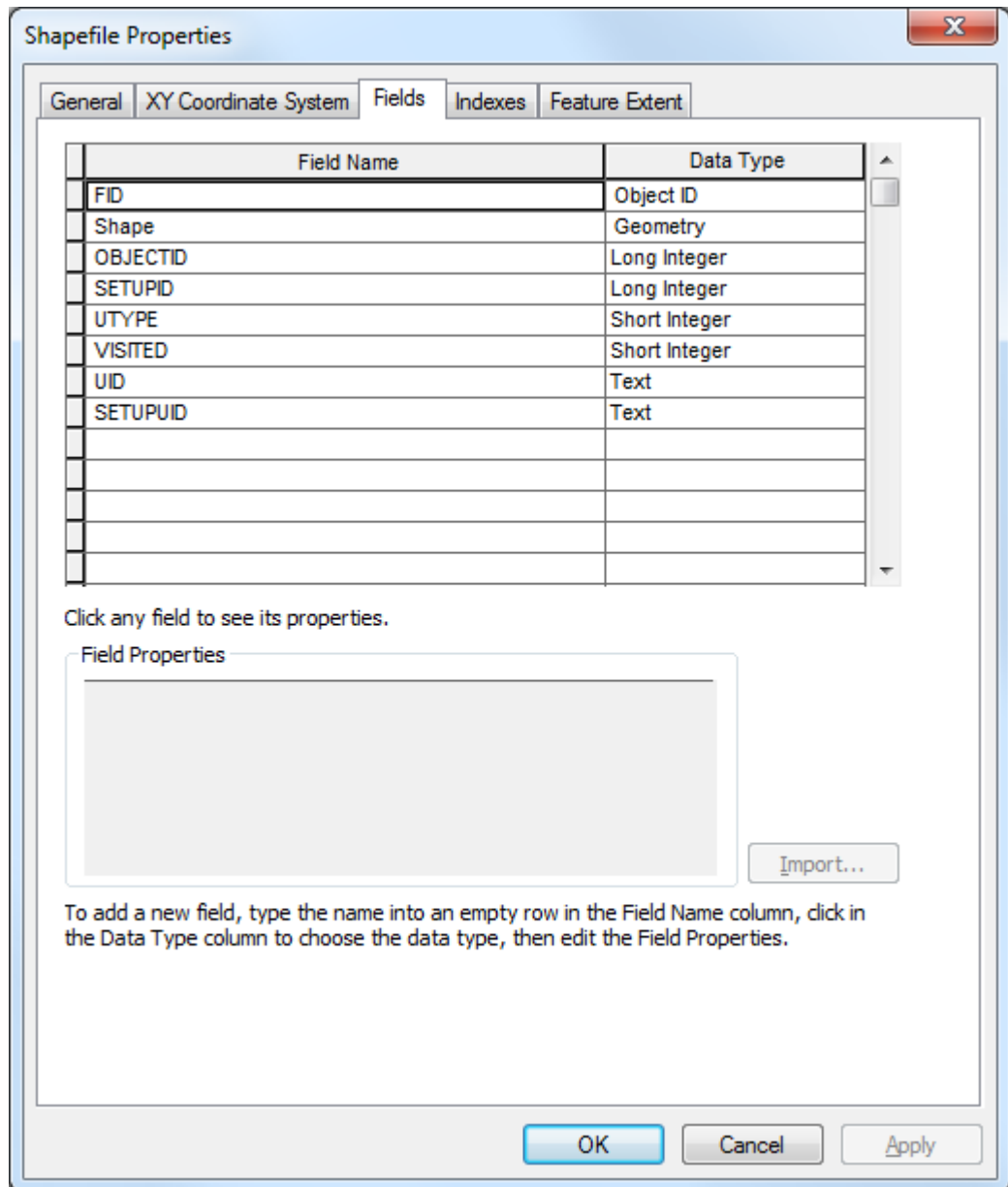
- plots\_subplots.dbf -> plotUID
    - plots\_trees.dbf -> subplotUID
  - plots\_gps.dbf -> plotUID
- Project Table: used to store project boundaries and names



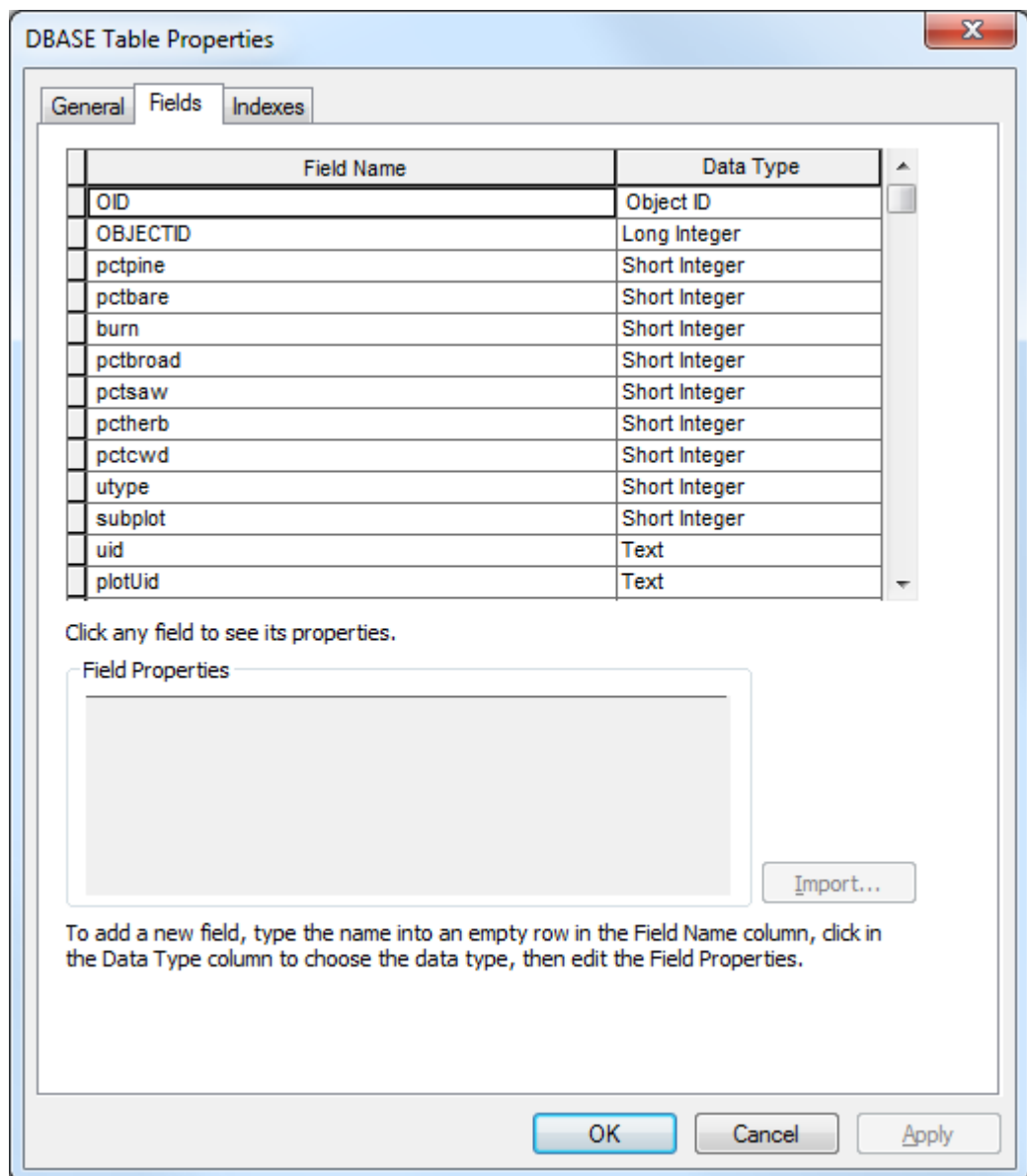
- Plots\_setup Table: used to describe the plot collect protocol



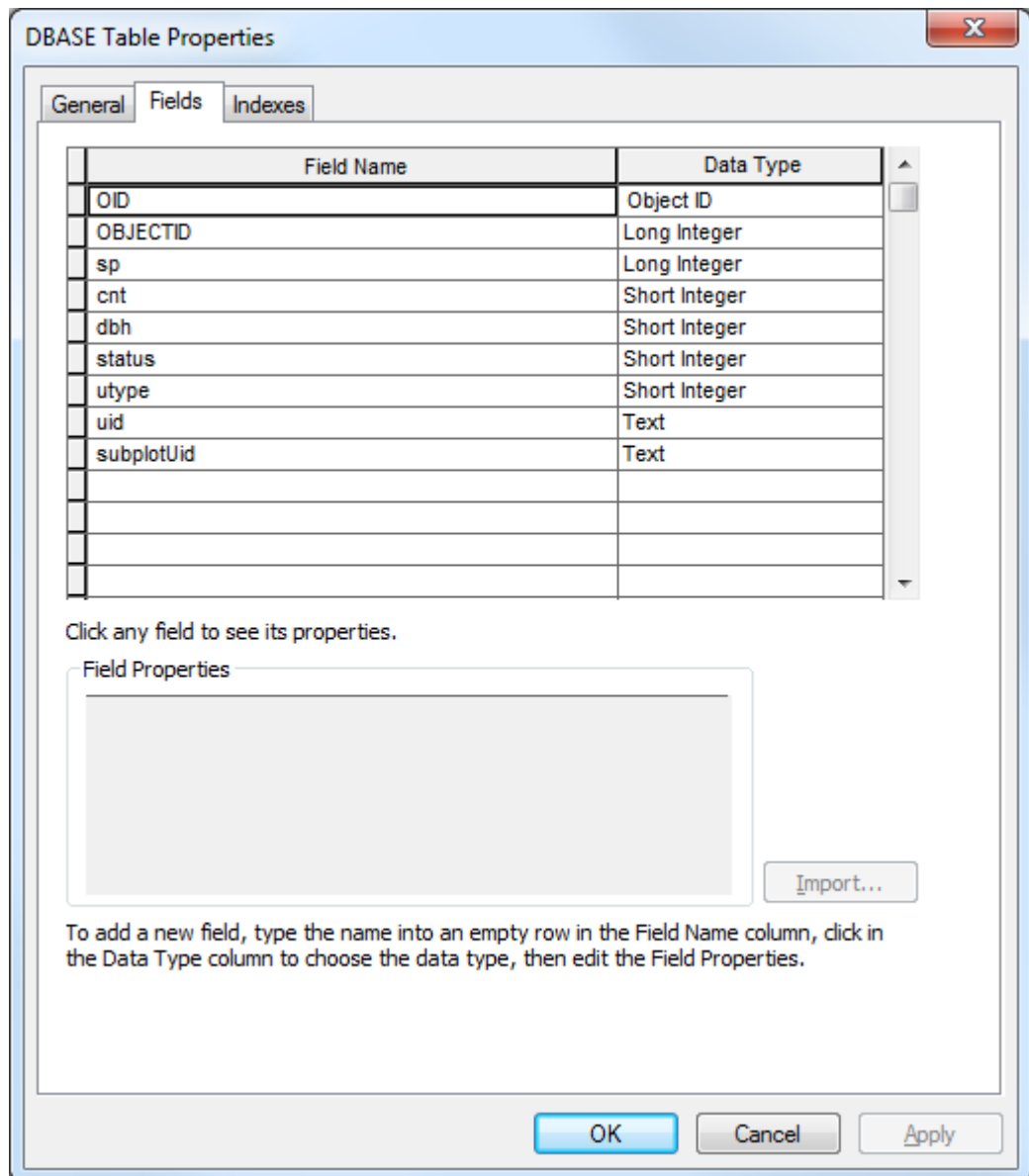
- Plots Table: used to store geometry of the plot and if the plot has been visited.



- Plots\_Subplots Table: used to store percent cover estimates

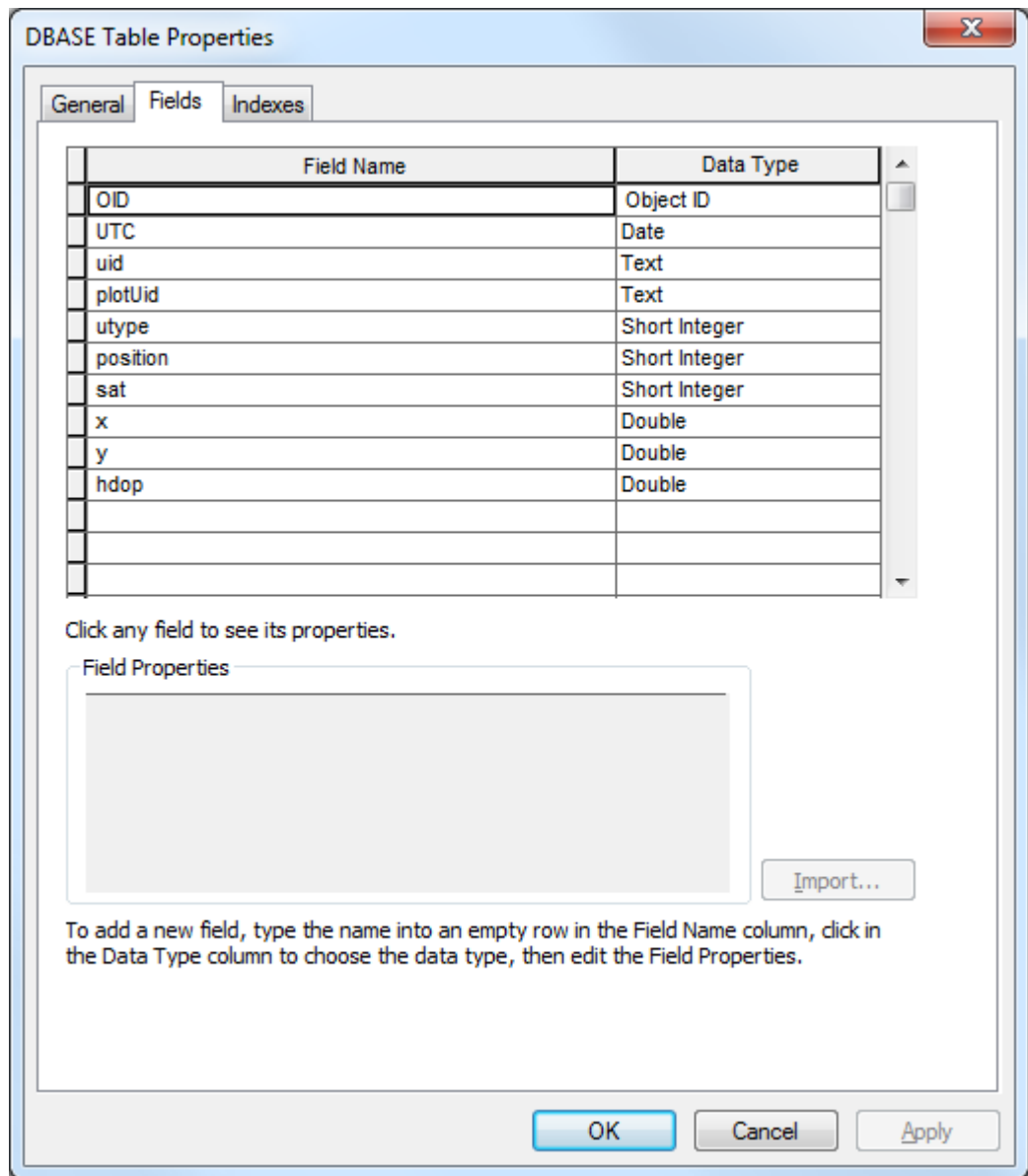


- Plots\_Trees Table: Used to store information about each tree within a subplot





- Plots\_Gps Table: used to store GPS information related to each plot



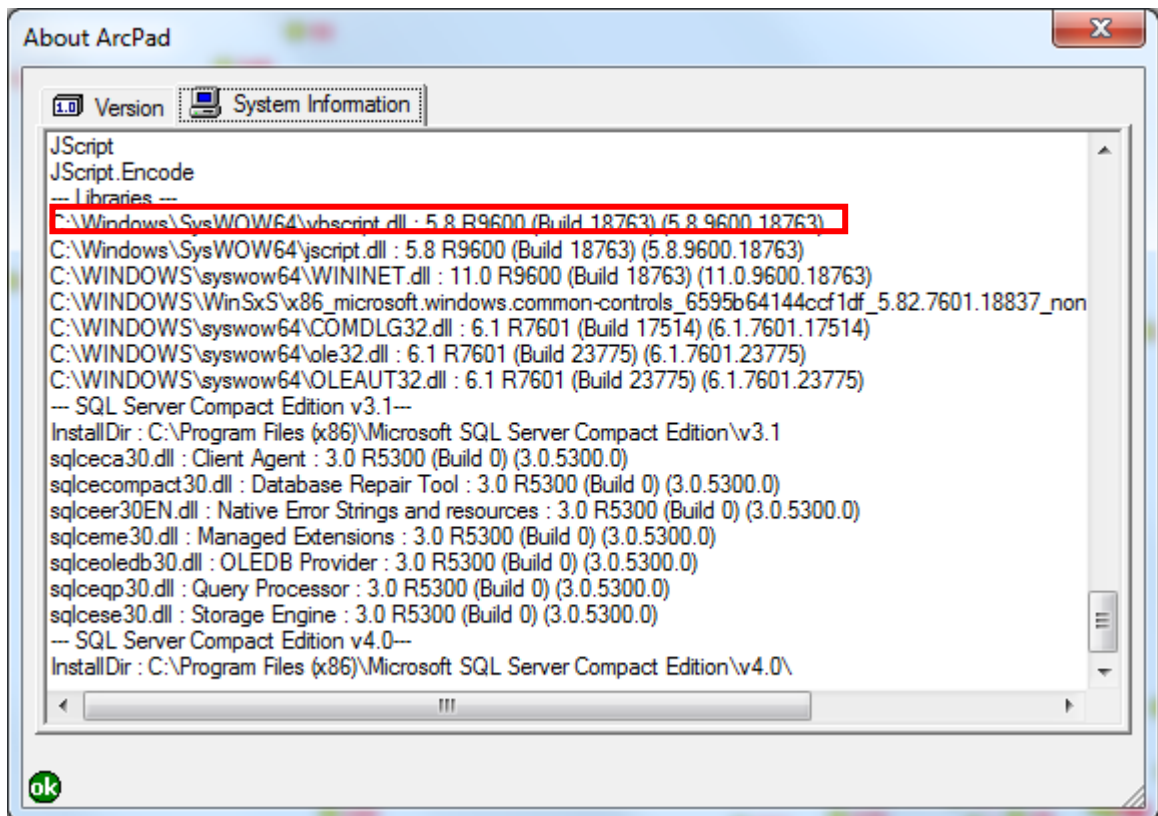
- PlotPics Directory: located within the same directory as the plot and used to store picture taken in the field (picture name after plot UID).

### Coding library

The coding library consists of three primary files: plots.vbs, DataTable.vbs, and plots.apl. The plots.apl file contains the design content of the plot forms (ArcPad's xml). The plots.vbs contains most of the functionality within the forms (vbscript). These two files must be located in the same directory as the plots shape file. The DataTable.vbs file is a class library that contains the logic used to temporarily store table values. This file must be placed in ArcPad's Applets directory along with the supporting dbf tables located in the Tree\_Data\_Collection directory.

## Application Installation

To install the plots data collection application, copy the project directory that contains the plots files to your mobile device. In addition, copy the Tree\_DataCollection folder and the DataTable.vbs file to the Applets directory within ArcPad. On your mobile device this is typically located at \Program File\ArcPad 10.2\Applets. On your desktop this is typically located at C:\Program Files (x86)\ArcGIS\ArcPad10.2\Applets. Note, this application requires vbscript runtime library 5.8 or greater. If you are currently running an older version of the runtime library the application will not work. To determine which version of the vbscript runtime library is installed tap on About ArcPad>System Information and scroll down until you find the libraries section. Look at the vbscript runtime version.



If your version is less than 5.8, you need to update the vbscript runtime library version 5.8 R9600.