Title

Methods for Comparative Model Selection and Parameter Estimation in Diverse Modeling Applications

Permalink

https://escholarship.org/uc/item/6f92k06d

Author

Fortmann-Roe, Scott

Publication Date

2014

Peer reviewed|Thesis/dissertation

Methods for Comparative Model Selection and Parameter Estimation in Diverse Modeling Applications

By Scott Fortmann-Roe

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Environmental Science, Policy and Management

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Wayne Getz, Chair

Professor Rodrigo Almeida

Professor Alan Hubbard

Spring 2014

Methods for Comparative Model Selection and Parameter Estimation

in Diverse Modeling Applications

By Scott Fortmann-Roe

Abstract

Methods for Comparative Model Selection and Parameter Estimation in Diverse Modeling Applications

by

Scott Fortmann-Roe

Doctor of Philosophy in Environmental Science Policy and Management
University of California, Berkeley

Professor Wayne Getz, Chair

Predictive accuracy of a model is of key importance in research and to a lay audience. Diverse modeling methods and parameter estimation methods exist, such that a wide range of techniques are available from which to select when approaching a modeling task. Given this, two questions naturally arise in relation to a modeling task: model selection and model parameter estimation. This dissertation is intended to advance the theory and practice of model selection and parameter estimation for the topics discussed here.

- In Chapter 2, I develop A3, a novel method for assessing predictive accuracy and enabling direct comparisons between competing models in an accessible framework. This method uses resampling techniques to "wrap" predictive modeling methods and estimate a standard set of error metrics for both the model as a whole and additionally for each explanatory variable utilized by the model. Two case studies in the chapter illustrate the applied utility of the method and how improved models may not only result in increased predictive accuracy, but also potentially alter inferences and conclusions about the effects of parameters in the model. An *R* package implementing the method is made available on CRAN.
- In Chapter 3, I develop ICE, a novel method of home range estimation. ICE uses a competitive method for estimating home ranges. Effectively, an estimator of estimators, ICE pits existing home range estimators against each other, each of which may be best suited for a given type of data. By selecting between different approaches, ICE can theoretically improve on the performance of any individual estimator across heterogeneous data sets.
- In Chapter 4, I develop Contingent Kernel Density Estimation, an extension to Kernel Density Estimation designed to account for the case when observations are measured with a specific form of error. Chapter 4 develops the method and derives contingent kernels for commonly-used kernels and sampling regimes. An application of the method is presented

to data collected from the social networking site, Twitter, to estimate the national distribution of a sample of Twitter users.

- The study in Chapter 5 analyzes a large data set collected from Twitter. This study is based on data from over four million Twitter users and estimates parameters of this population with a primary focus on color preference choices made by these users. Novel results are found in this "big data" analysis approach that may not have been able to be identified with earlier, traditional approaches of sampling and surveying the behavior of individuals.

Dedicated to my wife who supported me through this work and my parents who fostered my love of exploration.

Table of Contents

Acknowledgements

My research and academic studies at the University of California, Berkeley were funded by a Berkeley Fellowship and a National Science Graduate Research Fellowship.

I am immeasurably grateful to the numerous people who supported my research and graduate studies at Berkeley. First and foremost are my advisor Wayne Getz and the members of my graduate committee Rodrigo Almeida and Alan Hubbard who provided scholarly guidance and collegial support. Additionally, I am grateful to the many devoted members of the Berkeley faculty who I have been privileged to receive instruction from including Perry de Valpine, John Harte, Marco Carone and Sandrine Dudoit. Lastly, I would like to thank my fellow graduate students and the post-docs in Wayne Getz's lab who provided invaluable guidance and mentorship, most notably Andy Lyons.

Finally, I would like to thank others outside of Berkeley who contributed to my intellectual development. First, Włodzimierz Wójcik who advised me during my earlier studies in Poland and who has continued to provide mentorship. Second, my colleague Gene Bellinger who in addition to providing key insights that drove some of my work was an indefatigable source of encouragement and enthusiasm pushing me onwards. I would also like to thank Geoff McDonnell for his encouragement and our always-illuminating discussions.

1. Introduction

Models play an integral role in today's society; yet they are often distrusted. To assess the public's trust in models in general and the drivers of this trust, I commissioned two surveys of national scope each of approximately 1,500 people. The surveys were undertaken by Google Consumer Surveys (McDonald, Mohebbi, & Slatkin, 2012) from April to June 2013; for additional details, see Appendix A. Two survey questions were asked (each of a different sample), the first of which attempted to assess the general trust in scientific models of the United State's population of Internet users. Specifically, the following query was asked:

Scientific models have many uses including forecasting and estimation. In general, how much do you trust scientific models?

Respondents responded to this question on a 5-point scale, from 1 corresponding to the statement "Not at all" through 5 corresponding to the statement "I trust them completely". The survey results are summarized in Figure 1. In aggregate, the findings indicate the sample of Americans is slightly distrustful of models with a mean response of 2.8 (where a value of 3.0 would indicate neutrality). Some demographic data were also available as part of the survey (namely gender, income, location, and age). There was a statistically significant difference in model trust between genders with men (mean = 2.91) on average trusting models more than women (mean = 2.75) (*p* value = 0.018). There was also a difference in model trust between locations with urban respondents (mean = 2.92) on average trusting models more than suburban respondents (mean = 2.79) (*p* value = 0.040).

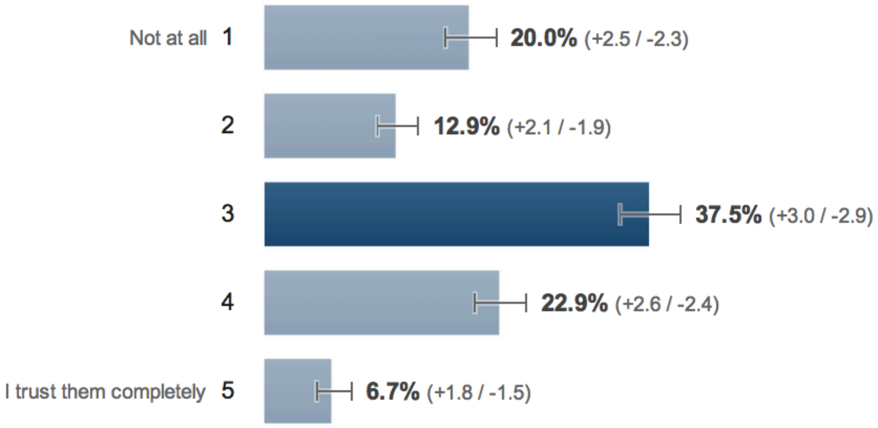


Figure 1. Distribution of responses to the question: "Scientific models have many uses including forecasting and estimation. In general, how much do you trust scientific models?" Results are weighted to mimic the population of United States Internet users (*n*=1,208).

In order to assess drivers of laypeople's confidence or lack of confidence in models, the following second question was asked: "To **trust** a model, what is **most**

important for you? Scientific models are used to understand things like the weather or the economy” (emphasis in the original question). Five categorical response options were available to the respondents: “Model predicts events accurately”, “Model is widely used”, “Model is developed and validated by experts”, “Model assumptions are understandable”, and “Model design mirrors reality.” The order of response options was randomized when presented to each respondent. The distribution of results is shown in Figure 2.

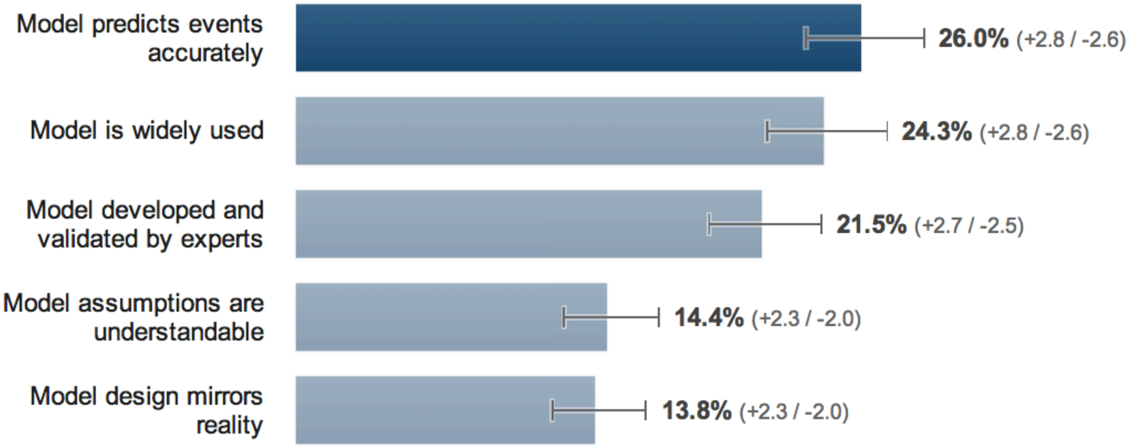


Figure 2. Distribution of responses to the question: “To trust **a model, what is** most important **for you? Scientific models are used to understand things like the weather or the economy.” Results are weighted to mimic the population of United States Internet users (*n*=1,171).**

Given the indication of predictive accuracy of being of primary importance in inspiring trust in a model[1] to a lay audience, two questions naturally arise in relation to a modeling task: model selection and model parameter estimation. A multitude of modeling techniques are available to be selected from and a range of conceptual frameworks are often used to classify them. For instance, models are often classified as deterministic or stochastic [e.g. (Allen & Burgin, 2000) or (Petrovskii, Malchow, Hilker, & Venturino, 2005) for works making this distinction]; as mechanistic or statistical [e.g. (Kendall et al., 1999) or (Ellner et al., 1998) as works making this distinction]; as aggregated or disaggregated [e.g. (Nam, 1997) for a discussion focusing on this in transportation]; as Bayesian or frequentist when it comes to a statistical model [e.g. (Ellison, 2004) for presenting Bayesian inference for Ecological applications with a specific contrast to frequentist inference]; and so on. There are of course many nuances to such broad distinctions and other categorizations that can be made between model implementations.

As an example of one general approach to modeling we can take the instance of Machine Learning. At the intersection between statistics and computer science, the machine learning has a strong focus on applied predictive modeling and this field is typified by texts such as *The Elements of Statistical Learning* (Hastie, Tibshirani, &

[1] Though admittedly not all the results are statistically significant.

Friedman, 2009). Many methods fall under the general machine learning umbrella, including random forests (Breiman, 2001), support vector machines (Cortes & Vapnik, 1995), RuleFit (Friedman & Popescu, 2008), and MARS (Friedman, 1991). In addition, there are many improvements to traditional and existing methods such as the application of deep neural networks (Larochelle, Bengio, Louradour, & Lamblin, 2009) or the development of efficient regularization methods for linear regressions (Friedman, Hastie, & Tibshirani, 2010). These techniques can often provide regressions that are more accurate than more traditionally used modeling techniques.

Much of modeling can be thought of as an attempt to estimate the relationship among multiple variables. It is rare that the true relationship between variables will be known and instead an approximate must be estimated from available data and sources of knowledge. Some approaches to modeling are primarily driven by expert knowledge. For instance, System dynamics is a modeling technique focused on the elicitation, description and improvement of "mental models:" internal models or knowledge individuals have about a system (Richardson, Andersen, Maxwell, & Stewart, 1994), (Capelo & Dias, 2009). This type of modeling technique is primarily driven by expert knowledge, beliefs and assumptions about a system. Other modeling techniques, such as the machine learning methods introduced above, are primarily data driven. In these cases, modeler's beliefs and assumptions can still play a role in shaping the model, but data is the ultimate driver. Whatever modeling approach is taken, there is a danger of model misspecification. Misspecification occurs when an incorrect model is used to model the relationships between variables. Since the true or correct model will almost never be available, models will almost invariable be misspecified in practice, to a greater or lesser extent.

If a misspecified model is used, then inferences and conclusions generated by the model may well be incorrect. For instance, if a linear model is applied to data that was generated in a non-linear way, misleading and incorrect conclusions may be obtained. In practice, these conclusions are often arrived at based on the estimated values of parameters in the applied model (and the variance of these estimates). Formally, a parameter indexes a family of probability distributions. In our linear regression example, the coefficients in the regression are parameters. The estimates of these parameters and the variability of these estimates allow us to draw conclusions such as answering the question about whether a given explanatory variable has a significant influence on the dependent variable.

Although the true model for a non-trivial empirical data set is unlikely to be arrived upon, models that better approximate the true model should generally lead to more accurate inferences and predictions. One conceptual approach to the issue of errors in models is the bias variance tradeoff (Fortmann-Roe, 2012). Roughly speaking, "Bias" in an estimate of a parameter represents persistent deviation in the estimate of that parameter while "Variance" represents the variability of that estimate between realizations of the model. In general there is a tradeoff between these two forms of errors with techniques to reduce one often increasing the other.

We can create a graphical visualization of bias and variance in regards to estimating a parameter using a bulls-eye diagram. Imagine that the center of the target is the true value of a parameter. As we move away from the bulls-eye, our estimates get worse and worse. Imagine we can repeat our entire model building process (including new data collection each time) to get a number of separate hits on the target. Each hit represents an individual realization of our model and estimate of the parameter, given the chance variability in the training data we gather. Sometimes we will get a good distribution of training data so we estimate the parameter very well and we are close to the bulls-eye, while other times our training data might be full of outliers or non-standard values resulting in poorer estimates. These different realizations result in a scatter of hits on the target. A linear regression with a few predictor variables is an example of a model that generally has low variance and high bias. A fully-grown classification and regression tree is a model which has potentially lower bias but higher variance.

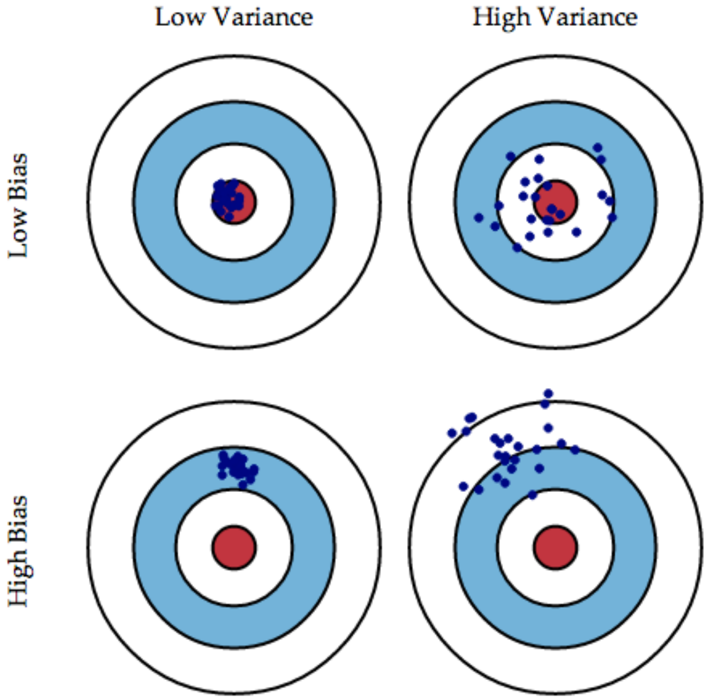


Figure 3. Graphical illustrations of the bias variance tradeoff[2].

A number of model selection techniques exist to identify the best model for a given data set from within a set of models. For instance, information theoretic approaches are popular within the ecological field (Burnham & Anderson, 2002) and here at UC Berkeley, Super Learner has been developed for selection with a predictive focus (Van Der Laan & Rose, 2011). Two chapters in this dissertation primarily address model selection. In Chapter 2, I use the A3 method to reanalyze a model and data set published in *Science*. The reader will see how moving from an

[2] This figure and description is directly taken from (Fortmann-Roe, 2012).

inaccurate linear regression model to a more accurate machine learning model not only increases the accuracy of our predictions, but, more important, alters our inferences and conclusions about the system under study in non-trivial ways. Similarly, in Chapter 3, I develop a novel home range estimator of estimators that selects among competing models.

Parameter estimation is of similar importance to model selection (and parameter estimation can be thought of as a form of model selection). Fundamentally, given a most likely unknown distribution from which a sample of data has been collected, how do we estimate parameters of that distribution? The parameters can range from the conceptually simple, such as the average value of a distribution, to the very much more complex such as parameters that are not scalar values. By way of an example of a complex parameter, consider the case of an animal's utilization of space. Imagine a hypothetical animal that sleeps in a given area and everyday explores or forages in the nearby regions. At a given point in time in the future, a distribution describes that animal's potential locations. For many animals we can represent this distribution as a bivariate distribution, where dense areas represent areas of frequent usage and less dense areas represent regions of infrequent usage.

Many parameters of this distribution may be of interest. The centroid (multi-dimensional mean) is one such parameter that could be of interest: what is the "average" position of the animal. A more complex, and potentially more useful, parameter might be the smallest region in which you will find the animal 90% of the time. This is known as the 90% isopleth and is also a parameter of the animal's utilization of space. Many techniques exist for the estimation of parameters in parametric models with Maximum Likelihood Estimation being likely the most frequently employed analytical tool. Maximum Likelihood Estimation is not always suitable. However, such as in cases where a likelihood function for the data does not exist. Such is, as we see in Chapter 3, the case for the LoCoH family of methods. In cases like these, alternate parameter estimation methods may be utilized.

One last point should be noted before I present a roadmap for the remainder of this dissertation. In applied work or in industry, a model may be the ultimate product of modeling effort. For example, when a person applies for a credit card, the credit card company may run a predictive model to score the potential customer's risk of defaulting [See (Thomas, 2000) for a historical overview of this specific modeling challenge]. In academic research, on the other hand, models are often what I term "middleware". Middleware is used here to refer to the models that are an internal, often implicit part of the data analysis processes. For instance, when comparing groups, a scientist may often use some form of ANOVA analysis. By doing so, they implicitly choose a model of their data – which may or may not accurately reflect nature of their data. In such cases, the model middleware may never be documented and exposed to the model users who will instead just see simplified results and inferences (e.g. "Treatment A outperformed Treatments B and C, an effect significant at the 0.05% level."). When more complex models are developed in academic research, they often also serve as middleware in effect machines generating *p* values for the significances of different relationships.

The chapters in this dissertation relate both to these middleware models and to models that are end products in and of themselves. In this work, I present four chapters on applied parameter estimation and model selection. Each chapter addresses these questions from a different perspective:

In Chapter 2, I develop A3, a novel method for assessing predictive accuracy and enabling direct comparisons between competing models in an accessible framework. A3 uses resampling techniques to "wrap" predictive modeling methods and estimate a standard set of error metrics for both the model as a whole and additionally for each explanatory variable utilized by the model. These error metrics are standard across predictive models allowing apples to apples comparisons between models and facilitating the comparison of typical models, such as linear regressions, to more exotic models, such as random forests. The method is focused on accuracy, adaptability between methods and accessibility to users leading to the moniker A3, pronounced "A Cubed" (Accuracy, Adaptability and Accessibility. Two case studies are included in the chapter illustrating the applied utility of the method and how improved models may not only result in increased predictive accuracy, but also potentially alter significances employed in a Null Hypothesis Significance Testing framework thereby leading to altered conclusions.

In Chapter 3, I develop Isopleth Cross Validation (ICE), a novel method of home range estimation. In keeping with the general approach presented by A3 in Chapter 2, ICE uses a competitive approach to estimating home ranges. Effectively, an estimator of estimators, ICE pits existing home range estimators against each other. For instance, ICE may be used to select between Kernel Density Estimation and the LoCoH family of home range estimators to select the estimator that is best for a given data set. In general, Kernel Density Estimation may outperform LoCoH for utilization distributions governed by gradual, smooth changes in densities. LoCoH, on the other hand, should generally outperform Kernel Density Estimation for utilization distributions governed by sharp boundary (as created by, say, a river or a fence). By selecting between different approaches, ICE can theoretically improve on the performance of any individual estimator across heterogeneous data sets.

In Chapter 4, I develop Contingent Kernel Density Estimation, an extension to Kernel Density Estimation to account for the case when observations are measured with a specific form of error. Although a general method, like ICE, Contingent Kernel Density Estimation is strongly motivated by the needs of home range estimation and the analysis of spatial data. This chapter develops the method and derives contingent kernels for commonly used kernels and sampling regimes. An application of the method is presented to data collected from the social networking site, Twitter, to estimate the distribution of a sample of Twitter users. This data set is suitable for the Contingent Kernel method as different errors are associated with the observation of each user in the data set, with some user positions being measured with GPS accuracy while other users are measured with resolutions as coarse as the radius of a country.

In chapter 5, I analyze a large data set collected from Twitter. This study is based on data from over four million Twitter users and estimates parameters of this population with a primary focus on color preference choices made by these users. Novel results are found which may not have been able to be identified with earlier, traditional approaches of sampling and surveying the behavior of individuals. These "big data" approaches to analysis open up many avenues to ask and answer novel and important questions but also pose new challenges in terms of data management of analysis of truly massive data sets whose sizes may be measured in many terabytes. Interesting work on developing tools to analyze these data sets is both being done in academia and within industry such as the development of Google's high performing Dremel database engine (Melnik et al., 2010).

I hope that through the developments in these chapters, I have advanced the theory and practice of model selection and parameter estimation primarily in the ecological sphere for the topics discussed here.

References

Allen, L., & Burgin, A. M. (2000). Comparison of deterministic and stochastic SIS and SIR models in discrete time. *Mathematical Biosciences*, *163*(1), 1–33. doi:10.1016/S0025-5564(99)00047-4

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32.

Burnham, K. P., & Anderson, D. R. (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer-Verlag.

Capelo, C., & Dias, J. F. (2009). A system dynamics-based simulation experiment for testing mental model and performance effects of using the balanced scorecard. *System Dynamics Review*, *25*(1), 1–34. doi:10.1002/sdr.413

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*(3), 273–297.

Ellison, A. M. (2004). Bayesian inference in ecology. *Ecology Letters*, *7*(6), 509–520. doi:10.1111/j.1461-0248.2004.00603.x

Ellner, S. P., Bailey, B. A., Bobashev, G. V., Gallant, A. R., Grenfell, B. T., & Nychka, D. W. (1998). Noise and Nonlinearity in Measles Epidemics: Combining Mechanistic and Statistical Approaches to Population Modeling. *American Naturalist, 151*(5), 425–440. doi:10.1086/286130

Fortmann-Roe, S. (2012, June). Understanding the Bias-Variance Tradeoff. Retrieved January 3, 2014, from http://scott.fortmann-roe.com/docs/BiasVariance.html

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics*. doi:10.2307/2241837

Friedman, J. H., & Popescu, B. E. (2008). Predictive Learning via Rule Ensembles. *The Annals of Applied Statistics*, *2*(3), 916–954. doi:10.1214/07-AOAS148

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *27*(6), 957–

968. doi:10.1109/TPAMI.2005.127

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. New York, NY: Springer New York. doi:10.1007/978-0-387-84858-7

Kendall, B. E., Briggs, C. J., Murdoch, W. W., Turchin, P., Ellner, S. P., McCauley, E., et al. (1999). Why do populations cycle? A synthesis of statistical and mechanistic modeling approaches. *Ecology*, *80*(6), 1789–1805.

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring Strategies for Training Deep Neural Networks. *The Journal of Machine Learning Research*, *10*, 1–40.

McDonald, P., Mohebbi, M., & Slatkin, B. (2012). Comparing Google consumer surveys to existing probability and non-probability based Internet surveys.

Melnik, S., Gubarev, A., Long, J. J., Romer, G., Shivakumar, S., Tolton, M., & Vassilakis, T. (2010). Dremel: interactive analysis of web-scale datasets. *Proceedings of the VLDB Endowment*, *3*(1-2), 330–339.

Nam, K. C. (1997). A study on the estimation and aggregation of disaggregate models of mode choice for freight transport. *Transportation Research Part E-Logistics and Transportation Review*, *33*(3), 223–231. doi:10.1016/S1366-5545(97)00011-2

Petrovskii, S. V., Malchow, H., Hilker, F. M., & Venturino, E. (2005). Patterns of Patchy Spread in Deterministic and Stochastic Models of Biological Invasion and Biological Control. *Biological Invasions*, *7*(5), 771–793. doi:10.1007/s10530-005-5217-7

Richardson, G. P., Andersen, D. F., Maxwell, T. A., & Stewart, T. R. (1994). Foundations of mental model research, 181–192.

Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International Journal of Forecasting*, *16*(2), 149–172. doi:10.1016/S0169-2070(00)00034-0

Van Der Laan, M., & Rose, S. (2011). *Targeted Learning: Causal Inference for Observational and Experimental Data*. New York: Springer.

2. Assessing Predictive Model Accuracy for Comparative Purposes: A3

Introduction

In the first research chapter of this dissertation, A3, a platform and assessment framework for developing predictive models, is presented. A wide variety of metrics have been developed to assess model results for prediction or other inferences. Such metrics include the classic coefficient of determination and *p* value, risk functions such as MSE, and newer approaches such as information theoretic techniques like AIC or BIC. Although many approaches can be used to assess such error metrics, it can be argued that three basic criteria should be employed to assess metrics for predictive models:

- **Accuracy** Does the method accurately report error or goodness of fit?
- **Accessibility** Is the manner in which the results are reported understandable to end-users?
- **Adaptability** Can the method be applied to a variety of different modeling approaches?

The importance of accuracy is self-evident. If the method of reporting model fit itself contains significant inaccuracies, it may be of little value. The widely reported metric R^2 is perhaps the best example of a commonly used, yet potentially highly inaccurate measure of error. The issue with the coefficient of determination is, of course, that it will always increase as more variables are added to the model. Even if the added variables are independent from the desired prediction target, R^2 will still increase. The commonly used adjusted R^2 measure attempts to account for this issue, but even that may be slightly biased when the model is overfit.

Accessibility is of equal importance to accuracy. A metric may objectively be highly accurate, but if practitioners "on the ground" are unable to properly apply it in practice, then it has little utility. An example of a widely discussed measure ((Falk & Greenbaum, 1995; Loftus, 1993; Nickerson, 2000) and others) that has repeatedly been shown to be inaccessible is the *p* value. Although the subject of basic statistics, *p* values have been demonstrated to not only be misunderstood by laypeople, but also be misunderstood by university statistics instructors. In one study of 30 instructors, 80% made at least one error when answering six basic true/false questions about the interpretation of *p* values[3] (Haller & Krauss, 2002).

[3] Such as, "[Given a *p* value] you know, if you decide to reject the null hypothesis, the probability that you are making the wrong decision."

Adaptability is of almost as much importance as accuracy and accessibility. This criterion indicates how flexible the metric is in regards to being applied to different forms of model construction algorithms. Some metrics – such as AIC or BIC, for instance – require statistical models that generate likelihoods. Many types of models naturally generate likelihoods, but other types of model construction algorithms (such as CART (Breiman, Friedman, Olshen, & Stone, 1984)) may not naturally generate likelihoods. By utilizing a metric that is limited to a certain subspace of model construction algorithms, we limit ourselves in our ability to obtain the best-suited models for a given data-fitting problem.

The A3 method (pronounced A^3) and the **A3** package implementation of it for *R* (R Core Team, 2012) targets these three criteria. It is designed to be accessible, accurate and adaptable[4]. In terms of accessibility, the method is constructed to use familiar concepts. It is based around the R^2 metric and *p* values. Because these metrics are familiar, most practitioners will not have to learn new concepts to understand the A3 output. The A3 method utilizes a derivative of R^2 – the added R^2 – in order to assess variable importance and the contribution of each feature to the success of the overall model. Unlike likelihood-based approaches – such as AIC-based approaches – or more application specific approaches – such as Gini importance for tree-base methods– this technique is quite general and so can be applied wherever A3 itself may be utilized.

The added R^2 metric, analogous to semi partial correlation in linear regression, indicates how much the model improves when a given variable is added to the model. This change indicates the practical importance of the added variables which in practice can be much more meaningful than statistical significance. The A3 method also provides a slope metric, analogous to the slope in linear regression, to indicate the effect of a variable on the outcome. In summary, the method calculates the following three items for each feature in the data set:

- **Slope**: How a change in the feature affects the dependent variable. A distribution of values is calculated for each feature in addition to a single, summarizing average.
- **Added R^2:** The predictive utility of the feature that is unique from all other features in the model.
- ***p* Value:** The chance of seeing the observed level of predictive utility for a feature assuming the null hypothesis that the feature in fact has no predictive utility.

Muñoz & van der Laan (2012) introduced a parameter and derived asymptotically linear, semiparametric estimators of said parameter to quantify the effect of displaced covariates on an outcome. If this parameter were to be normalized by the displacement amount, it would be similar to the slope calculated by the A3 package. In the A3 package, the slope parameter for a feature is estimated

[4] It is from the acronym of these three adjectives that the method is named.

by approximating the slope at each data point using simple displacement and then averaging these results. While this is a simpler approach than Munoz and Van der Laan's approach, it has unknown and almost certainly worse variance and convergence properties. Possibly the Munoz and Van der Laan approach may be adopted in a future version of the A3 package.

In terms of accuracy, the A3 method uses robust, resampling-based methods for the calculation of *p* values and R^2 metrics. R^2 is calculated using cross validation which correctly accounts for overfitting and does well in matching the true R^2 value[5]. *p* values are calculated using an exact test. Full details about the methods are available in Appendix B. These methods may be computationally intensive for complex models, but they require no parametric assumptions other than independence between observations (a constraint which itself may be violated, see the dealing with correlations section of this paper). The use of these methods make the results reported by the A3 method much more robust to user misuse and abuse than do standard parametric model results (where parametric assumptions are frequently not tested for in practice). Thus the results of the A3 method are both objectively accurate, but also likely to remain accurate even when used by individuals with limited statistical expertise.

Last, the A3 method is highly adaptable to different modeling techniques. Technically, the A3 method is defined as a wrapper function that encapsulates a predictive modeling algorithm. Thus the method can be applied to any predictive modeling algorithm. The same principle is used in the **A3** package where the primary package functions can take an arbitrary predictive modeling function. Different modeling methods can be utilized by passing different functions (e.g., *lm* for Linear Regression models, *glm* for Logistic Regression models, *rpart* for CART models) to the package's *a3* function. The A3 method can seamlessly encapsulate different techniques and generate a consistent output for them that facilitates the direct comparison of these different methods using the same criteria.

The following sections of this chapter will first describe the **A3** package in general. Next, two applications of the method are developed for predictive and inferential tasks. Finally, a discussion of using the package to deal with correlated data will be presented followed by general conclusions. This chapter was written for submission to the *Journal of Statistical Software*. That journal prefers articles that contain significant examples and applications of the presented techniques. This chapter is written in that style and it consequently contains numerous code samples and illustrations of the direct usage of the **A3** package.

One important note should be made at this point. In general, I would make the claim that mathematical models are constructed for three purposes: prediction,

[5] R^2 may be defined as the fraction of the squared error explained by a model compared to the null model. This is the definition used in the A3 method's calculation of R^2.

inference other than prediction, and conceptual/narrative applications. The **A3** package can be applied both to predictive and other inferential usage cases. It is based in a predictive framework, but it can also assess the statistical significance of variables within this framework. The key to doing this is by making a small leap in our thinking about inference in order to reframe it in a predictive manner. As an example, take the question "Does *X* significantly affect *Y*?" which implicitly implies a probability model. We can rephrase the same question in a predictive framework as "Does knowledge of *X* help us to predict *Y*?" The two questions are equivalent, and the latter is answered by the **A3** package allowing inferences based solely on the predictive accuracy of models.

Usage Overview

The primary function provided by the **A3** package is *a3* which takes three principle arguments:

- **formula**: A regression formula object

- **data**: A data frame containing the data for the regressions

- **model.fn**: A function object which generates a regression that has a corresponding *predict* generic method

As example of the usage of the *a3* function, we may use *R*'s built in *lm* function with the *a3* function to generate the **A3** results for a Linear Regression model of *R*'s built-in *attitude* data set. The output is an S3 object with a generic *print* function that displays an A3 results table. This table contains four columns: the features in the model, the average slope for each feature (analogous to the coefficient in a Linear Regression, and directly equivalent to it when used for a Linear Regression model), the cross-validated R^2 for the whole model and the added R^2 for each feature (a measure of the practical significance of that feature), and the *p* values for the model itself and for each feature.

```
R> a3(formula = rating ~ ., data = attitude, model.fn =
lm)

             Average Slope   CV R^2 p value
-Full Model-                 47.8 %  < 0.01
(Intercept)    10.78707639 -  8.5 %    0.97
complaints      0.61318761 + 10.5 %  < 0.01
privileges     -0.07305014 -  5.7 %    0.87
learning        0.32033212 +  7.8 %    0.05
raises          0.08173213 -  6.1 %    0.82
critical        0.03838145 -  8.3 %    0.98
advance        -0.21705668 -  3.6 %    0.59
```

R comes with numerous predictive modeling algorithms in its core distribution and default packages. An even larger set of modeling techniques are available in the wider ecosystem of user contributed and maintained packaged. The *a3* method can support most of these techniques (see Appendix C for details). For instance, the **e1071** package (Meyer, Dimitriadou, Hornik, Weingessel, & Leisch, 2012) provides Support Vector Machine regressions (Cortes & Vapnik, 1995) using the function *svm*. We can use the *svm* function in place of the *lm* to obtain the A3 results for Support Vector Machines[6]:

```
R> library("e1071")
R> a3(rating ~ . + 0, attitude, svm)

             Average Slope   CV R^2 p value
-Full Model-                 42.3 %  < 0.01
complaints      0.37018519 + 25.3 %  < 0.01
privileges     0.079660425 -  3.5 %    0.44
learning       0.120305835 +  0.9 %    0.14
raises          0.07477541 +  4.0 %    0.11
critical      -0.138117725 - 11.5 %    0.92
advance         -0.1573689 +  5.4 %    0.06
```

As another example, we can use the **randomForest** package (Liaw & Wiener, 2002) to generate Random Forest regressions (Breiman, 2001). In order to speed calculation we can set the desired accuracy for the calculation of *p* values (*p.acc*) to a coarser value than the default 0.01. Setting *p.acc* to *NULL* would disable the calculation of *p* values completely.

```
R> library("randomForest")
R> out.rf <- a3(rating ~ . + 0, attitude, randomForest,
p.acc = 0.05)
R> out.rf

             Average Slope   CV R^2 p value
-Full Model-                 44.2 %  < 0.05
complaints          0.0443 + 27.0 %  < 0.05
privileges               0 +  3.4 %    1.00
learning          0.050075 + 14.3 %    0.05
raises         0.050816665 + 11.3 %    0.15
critical           -0.0052 +  0.9 %    1.00
advance       -0.019408335 +  7.9 %    0.50
```

[6] Please note the use of "+0" in the formula object. This removes the constant term generated in the model matrix which is unnecessary (and can cause errors) for the *svm* function.

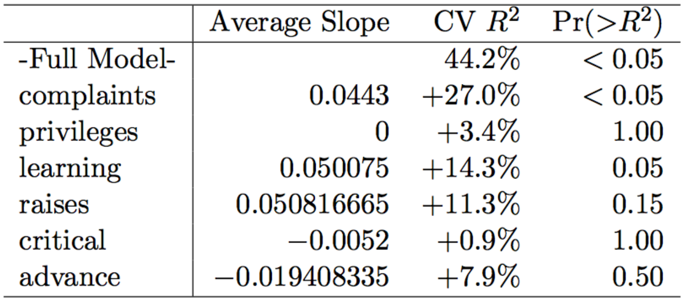
Arguments passed to the *a3* function are not passed on to the *model.fn* (in this case *randomForest*). To pass additional arguments to the *model.fn*, you can pass a list of arguments to *model.args*. For instance, the following code sets the *ntree* argument of *randomForest* to 1,000.

```
R> a3(rating ~ . + 0, attitude, randomForest, model.args
= list(ntree = 1000), + p.acc = 0.05)

             Average Slope   CV R^2 p value
-Full Model-                 44.3 %  < 0.05
complaints      0.039670835 + 24.2 %  < 0.05
privileges                0 +  2.5 %    0.90
learning         0.04915417 + 14.2 %    0.05
raises            0.0318375 +  8.2 %    0.30
critical        -0.00262083 +  0.5 %    0.95
advance        -0.020308335 +  2.5 %    0.90
```

In addition to the general *a3* function, we can also use the specialized *a3.lm* function specifically for linear models which removes the need for the argument *model.fn* (it is set automatically to *glm*). Both *a3* and *a3.lm* return an S3 *A3* object. The default *print.A3* function prints an ASCII results table. You can also use the **xtable** package (Dahl, 2013) to create a nicely formatted output table from this object .

```
R> xtable(out.rf, caption = "\\LaTeX \\, formatted A3
output", label = "xtableFormat")
```



	Average Slope	CV R^2	Pr(>R^2)
-Full Model-		44.2%	< 0.05
complaints	0.0443	+27.0%	< 0.05
privileges	0	+3.4%	1.00
learning	0.050075	+14.3%	0.05
raises	0.050816665	+11.3%	0.15
critical	−0.0052	+0.9%	1.00
advance	−0.019408335	+7.9%	0.50

Table 1: LaTeX formatted A3 output

Several plotting functions are included in the **A3** package to display results. *plotPredictions* is important in assessing the overall predictive accuracy of the model. For each observation in the data set, it plots the predicted and original values along with an optional line marking perfect results. In Figure 4 we can see that our Random Forest model for the attitude data appears to tend to overestimate ratings for low values and underestimate them for high ratings.

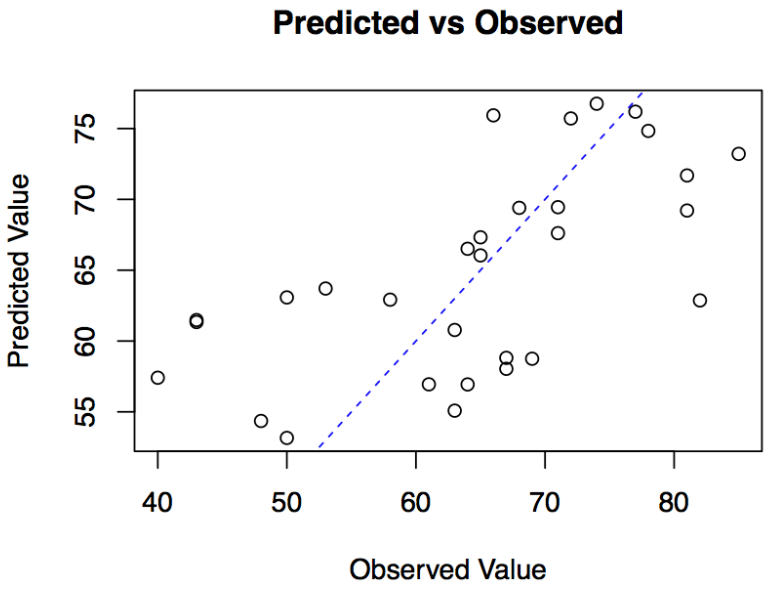


Figure 4. A plot of predicted versus observed values for Random Forest model of the attitude data.

In the results table, the average of the slopes at each observation are reported which are analogous to the coefficients of Linear Regressions. These slopes indicate how the prediction for an observation will change as the values of the observation's features change (or, more simply, whether a given feature has a positive or negative effect on the outcome). For some models, such as Linear Regressions, this slope will be constant between observations. However, for other models, such as Random Forests, the slope may change at each point as the model's behavior may differ between regions of the parameter space. The *plotSlopes* function may be used to plot the distribution of slopes for each feature.

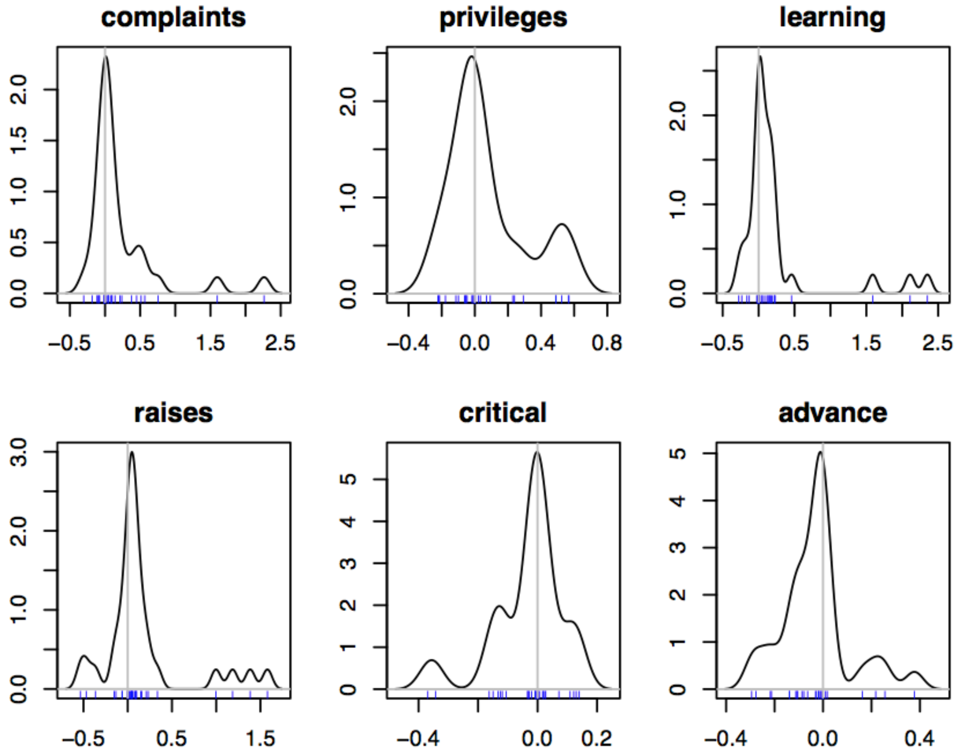


Figure 5. Slope distributions for the Random Forest model of the attitude data.

The generic *plot.A3* method may be used to plot both the predictions and slopes at once.

Worked Applications

The most effective way to illustrate the use of the **A3** package and its utility is through applied case studies. Two example applications will be used to illustrate the method. The first comes from an attempt to predict housing prices, the second is focused on drawing inferences in an ecological application.

Housing Application

This application is based on a data set that includes information on the prices of houses from the Boston area. It originally was developed by (Harrison & Rubinfeld, 1978) and the copy used in this analysis was provided by (Frank & Asuncion, 2010) .The data set is included in the **A3** package in the data variable *housing*. The following are some of the key features in the data set (based on the summary of (Frank & Asuncion, 2010).

- **NOX** Nitrogen oxides pollutant concentration (parts per 10 million)

- **ROOMS** Average number of rooms per dwelling

- **AGE** Proportion of owner-occupied units built prior to 1940

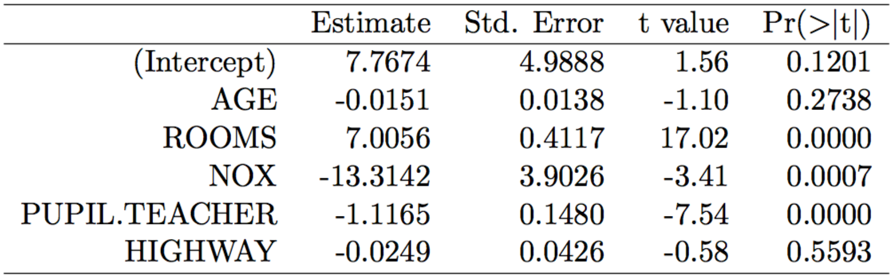
- **HIGHWAY** Index of accessibility to radial highways

- **PUPIL.TEACHER** Pupil-teacher ratio by town

- **MED.VALUE** Median value of owner-occupied homes in $1000's

A typical approach to analyzing this data set, either to build a predictive model or for inference, might be to apply a Linear Regression to the data set[7]. The results of a Linear Regression are shown in Table 1. Although these specific results are generated by *R*, the selection of displayed data is very similar to that of other packages. These results allow us to clearly see that *ROOMS*, *NOX*, and *PUPIL.TEACHER* are statistically significant variables as the 5% level. However, they do not provide information on the practical importance of these variables in regards to prediction accuracy.

Table 1. Standard Linear Regression model results for the housing data (R^2 = 0.6037, Adjusted R^2 = 0.5997).



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	7.7674	4.9888	1.56	0.1201
AGE	-0.0151	0.0138	-1.10	0.2738
ROOMS	7.0056	0.4117	17.02	0.0000
NOX	-13.3142	3.9026	-3.41	0.0007
PUPIL.TEACHER	-1.1165	0.1480	-7.54	0.0000
HIGHWAY	-0.0249	0.0426	-0.58	0.5593

In order to attempt to gain an understanding of the practical significance of the different variables, we can use the A3 results. Table 2 contains the A3 results for a Linear Regression of the housing data. The primary change in the physical construction of this table is the removal of the standard error and *t* value columns and the addition of the cross-validated R^2 column. This column first reports the R^2 for the whole model[8], and then the added R^2's for each feature.

Table 2. A3 results for a Linear Regression model for the housing data.

[7] It is important to note that, in this case, the authors of cited the paper carry out a very different analysis. What is shown in this section is simply illustrative of a typical approach.

[8] Which is slightly lower, as will generally be the case for overfit models, than the adjusted R^2.

	Average Slope	CV R^2	Pr(>R^2)
-Full Model-		59.3%	< 0.01
(Intercept)	7.76739265	−0.1%	0.26
AGE	−0.01509378	−0.1%	0.25
ROOMS	7.00564966	+23.2%	< 0.01
NOX	−13.31418184	+0.8%	< 0.01
PUPIL.TEACHER	−1.11645064	+4.5%	< 0.01
HIGHWAY	−0.02486812	−0.2%	0.96

From the added R^2 measures reported in the cross-validated R^2 column, it can be seen that the *ROOMS* variable explains 23% more of the squared error when it is added to the model while the *NOX* variable explains less than 1% of the squared error when it is added to the model. Although both these variables are highly statistically significant (p<0.001), the A3 output makes it very clear that *ROOMS* is a much more important predictor of housing prices than *NOX*. In applications, this information may have great practical importance such as helping to determine where to devote resources for additional data collection.

Furthermore, the **A3** package allows the straightforward comparison of results between different forms of predictive models. The A3 method is a wrapper that can theoretically be used on any predictive model (see Appendix C for more details). Table 3 and Table 4 show the results of the A3 method applied to, respectively, a Support Vector Machine model construction algorithm and a Random Forest model construction algorithm for the housing data set. There are two primary things to note from these results. The first is both of these algorithms were able to generate much better predictive models (10% to 15% higher R^2 values for the full models) than the Linear Regression.

Table 3. A3 results for the Support Vector Machine model for the housing data.

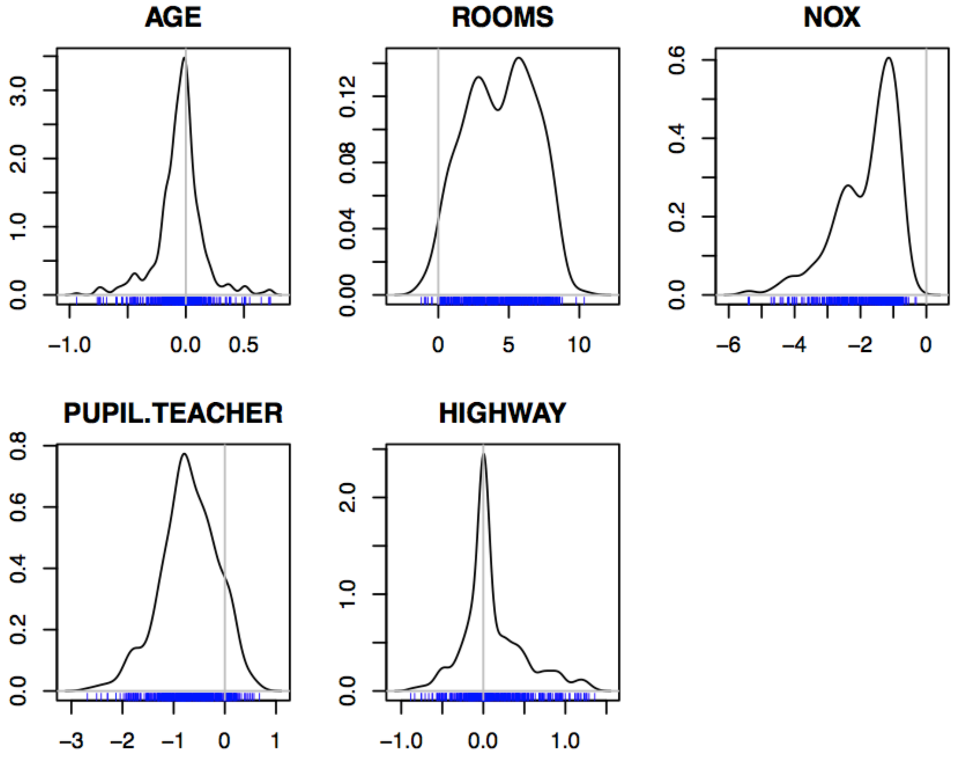
	Average Slope	CV R^2	Pr(>R^2)
-Full Model-		70.9%	< 0.01
AGE	−0.0694438	+0.9%	0.01
ROOMS	5.18863587	+34.7%	< 0.01
NOX	−0.00058093	+2.2%	< 0.01
PUPIL.TEACHER	−0.247276825	+1.6%	< 0.01
HIGHWAY	0.042275695	−0.3%	0.16

Table 4. A3 results for the Random Forest model for the housing data.

	Average Slope	CV R^2	Pr(>R^2)
-Full Model-		74.1%	< 0.01
AGE	−0.03457149	−1.1%	< 0.01
ROOMS	4.532621725	+20.3%	< 0.01
NOX	−1.477721415	+6.1%	< 0.01
PUPIL.TEACHER	−0.71400468	−1.1%	< 0.01
HIGHWAY	0	−2.2%	0.02

The second item of note, and which is of even greater importance, is that the inferences drawn from the data have changed in these latter models. In the Linear Regression model, the *AGE* variable is not at all significant. However in both the Support Vector Machine and Random Forest models, it is highly significant (p<0.01). Thus we can see that the data does in fact support a relationship between the age of a house and its price. It is not a trivial linear relationship, but it does exist. If we constrained ourselves to only explore linear models, as is often done in practice, we would have failed to identify this relationship and the significance of the *AGE* variable[9].

Figure 6. A plot of slopes for the Random Forest model of the housing data.



[9] It should be noted that if we were building a predictive model, it would still be best to exclude *AGE* from the model despite this statistical significance due to its low added R^2.

Lastly, it is beneficial to gain a fuller understanding of the meaning of the Average Slope column in these results. Unlike the case for the Linear Regression model, the slopes in the Support Vector Machine and Random Forest models may change between different regions of the parameter space. As such it can be useful to plot the distribution of slopes rather than simply relying on the median value as shown in the table. The results of this distribution are shown in Figure 6. From this figure it can be seen that the variable *NOX* always has a negative effect while a variable such as *AGE* has differing effects – sometimes negative and sometimes positive – depending on where a specific observation fits within the parameter space.

Ecological Application

This case study relates a dryland ecosystem's multifunctionality (roughly speaking, the magnitude of different services performed by an ecosystem) to a range of different environmental variables. It was collected by a large team of researchers and recently published in the journal Science along with a statistical analysis (Maestre et al., 2012).

A copy of the data set is provided by the **A3** package in the data variable *multifunctionality*. The following features are in the data set:

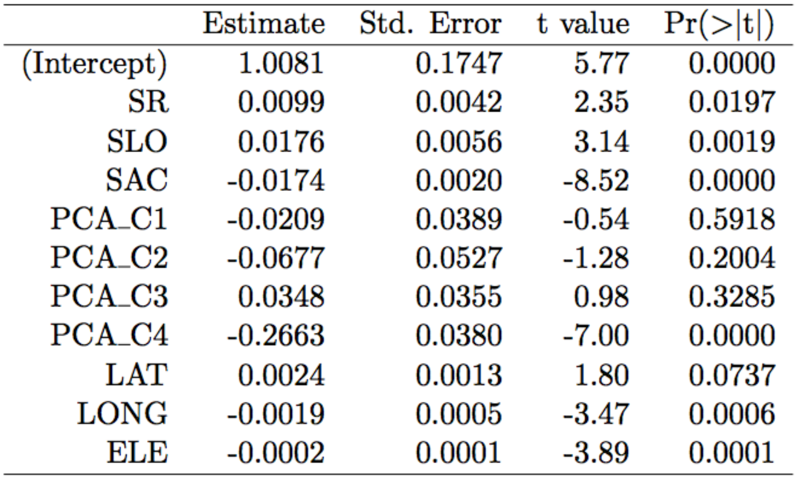
- **ELE:** Elevation of the site
- **LAT** and **LONG:** Location of the site
- **SLO:** Site slope
- **SAC:** Soil sand content
- **PCA_C1**, **PCA_C2**, **PCA_C3**, **PCA_4:** Principal components of a set of 21 climatic features
- **SR:** Species richness
- **MUL:** Multifunctionality

The original data were analyzed using a multi-model inference approach after (Burnham & Anderson, 2002). Only Linear Regressions were considered in this model selection[10]. The base Linear Regression explored in the original work is shown in Table 5 will all other Linear Regressions explored being subsets of this

[10] The authors also look at a simultaneously autoregression model, but that is not considered in their model selection and is not a focus of their work

original one. From this table, we can see that *SR*, *SLO*, *SAC*, *PCA_C4*, *LONG*, and *ELE* are all statistically significant at the 5% level.

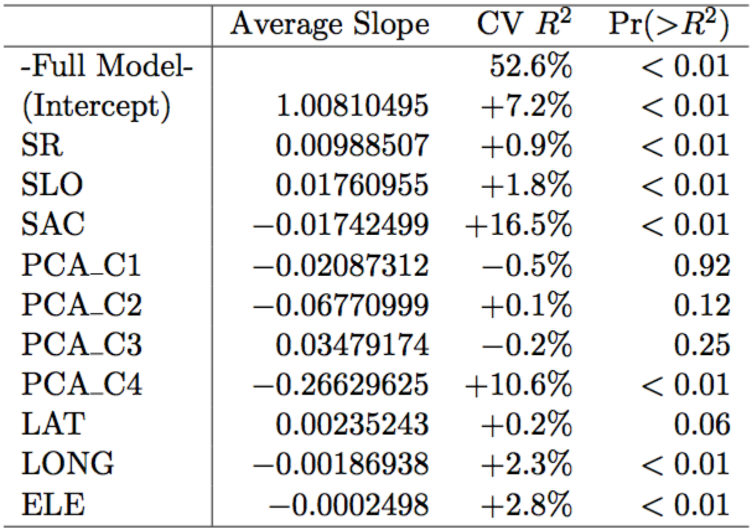
Table 5. Standard Linear Regression model results for the multifunctionality data (R^2 = 0.5645, Adjusted R^2 = 0.5441).



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.0081	0.1747	5.77	0.0000
SR	0.0099	0.0042	2.35	0.0197
SLO	0.0176	0.0056	3.14	0.0019
SAC	-0.0174	0.0020	-8.52	0.0000
PCA_C1	-0.0209	0.0389	-0.54	0.5918
PCA_C2	-0.0677	0.0527	-1.28	0.2004
PCA_C3	0.0348	0.0355	0.98	0.3285
PCA_C4	-0.2663	0.0380	-7.00	0.0000
LAT	0.0024	0.0013	1.80	0.0737
LONG	-0.0019	0.0005	-3.47	0.0006
ELE	-0.0002	0.0001	-3.89	0.0001

However, as with the housing application, we are again missing information on how important each of these variables are to the actual predictive accuracy of the model. This information is simply not available in the standard Linear Regression output table. The A3 method, however makes them very clear. As shown in Table 6, *SAC* explains the most additional squared error when added to the model, followed by *PCA_C4*. This basic conclusion agrees with what the original researchers found using their more complex information theoretic approach as they note "By this criterion [the sum of Akaike weights across models], the two most important predictors of multifunctionality were annual mean temperature (reflected in ... [*PCA_C4*]) and the sand content in the soil [*SAC*]."

Table 6. A3 Results for a Linear Regression model for the multifunctionality data.



	Average Slope	CV R^2	Pr(>R^2)
-Full Model-		52.6%	< 0.01
(Intercept)	1.00810495	+7.2%	< 0.01
SR	0.00988507	+0.9%	< 0.01
SLO	0.01760955	+1.8%	< 0.01
SAC	−0.01742499	+16.5%	< 0.01
PCA_C1	−0.02087312	−0.5%	0.92
PCA_C2	−0.06770999	+0.1%	0.12
PCA_C3	0.03479174	−0.2%	0.25
PCA_C4	−0.26629625	+10.6%	< 0.01
LAT	0.00235243	+0.2%	0.06
LONG	−0.00186938	+2.3%	< 0.01
ELE	−0.0002498	+2.8%	< 0.01

However, if we only use a Linear Regression model we run the risk of missing important nonlinear relationships. Using the **A3** package, we can quickly explore different modeling techniques and compare them to the linear results. Table 7 shows the A3 results for applying a Random Forest model to the multifunctionality data.

Table 7. A3 results for the Random Forest model for the multifunctionality data.

	Average Slope	CV R^2	Pr(>R^2)
-Full Model-		67.9%	< 0.01
SR	0.001592165	+1.2%	< 0.01
SLO	0.002248085	−1.4%	0.87
SAC	−0.00465617	+3.4%	< 0.01
PCA_C1	0.07261525	+1.5%	< 0.01
PCA_C2	0.0274955	+0.3%	0.01
PCA_C3	0.01562067	−0.1%	0.24
PCA_C4	−0.090020195	+0.7%	< 0.01
LAT	0.01274717	+0.9%	< 0.01
LONG	0.00047633	+0.4%	0.04
ELE	0	+0.4%	< 0.01

As with the housing data, there are two things to note. First the Random Forest model has much higher predictive accuracy compared to the Linear Regression model (0.679 R^2 compared to 0.526). Second is that our inferences change. For instance, *PCA_C1* was not significant when using a Linear Regression. When using the Random code model, however, it became highly significant. As with the housing data, we can see that variables which do not appear statistically significant when using classical Linear Regression, may actually be highly significant; a fact that can sometimes be revealed when using more data adaptive modeling methods such as Random Forests.

We can also use the added R^2 's to go beyond simple statistical significance to determine the importance of the different features. From these, we can see that *SAC* and *PCA_C1* are the two most important features in the Random Forest model which is different from the *SAC* and *PCA_C4* features identified as the most important in the Linear Regression model[11]. If scientific or policy decisions were based on this data, this would raise the primary question of what important is being affected by the *PCA_C1* feature, a question that would not have been raised if we had simply relied on the Linear Regression results.

[11] It should be noted that in the Random Forest results, the added R^2 's are by and large much lower than those seen for the Linear Regression results. This is most likely due to correlation between the features. The Random Forest model, in this case, is better at exploiting the duplicated information leading to lower added R^2's.

Controlling A3 Accuracy and Computation Time

For large data sets or complex model construction algorithms, the **A3** package may require significant computation effort. There are several arguments that can be used to fine-tune the behavior of the package and to either speed up computation or obtain increased precision in results.

n.folds: is the number of folds to use in the *k*-fold cross-validation. Increased number of folds leads to increased computation time. Generally speaking, a small number of folds will lead to an over-estimation of model error and the higher the number of folds, the more accurate the results will be. Since small numbers of folds leads to over-estimation of error, you may generally safely reduce the number of folds if computation is taking too long and you will obtain a conservative estimate of model accuracy. The maximum number of folds is the number of observations (Leave One Out Cross Validation) and a value of 0 for *n.folds* is shorthand to use this.

p.acc: controls the desired precision for the calculation of *p* values. A value of 0.01, for instance means *p* values will be calculated to the second decimal place. A value of 0.001, indicates that they will be calculated to the third decimal place. *p.acc* has a very important effect on performance. Roughly speaking, *a3* computation time is proportional to 1/p.acc. A value of *NULL* for *p.acc* specifies that *p* values should not be calculated at all.

features: controls the calculation of results for each feature. By default, *p* values and added R^2's are calculated for each of the features in the model. You can turn off the calculation of these by setting the *features* argument to false. When *features* is true, computation time is proportional to the number of features.

Generally speaking, given a model construction algorithm that takes time *t* to complete for a given data set, the **A3** package will, if *features* is *TRUE*, require *T* time as approximated in the following equation:

$$T \approx \frac{(n.features + 1) \times n.folds \times t}{p.acc}$$

If *features* is *FALSE*, *T* may be approximated with the following equation:

$$T \approx \frac{n.folds \times t}{p.acc}$$

Fortunately, the A3 method falls under the category of “embarrassingly parallelizable” algorithms. Although the package does not yet contain built-in parallelization code, this is planned for a future version of the package. Given enough resources for parallel computing, it would be conceivable to achieve close to $T \approx t$ in practice (assuming the model construction algorithm itself cannot take advantage of the parallel resources).

Dealing with Correlation between Observations

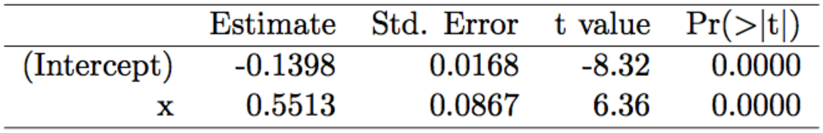
The basic A3 method makes no assumptions about the generation of the data except for one: independent and identically distributed observations. However, many prediction and inference tasks may in fact violate this assumption. For instance, some form of correlation between observations will often be the norm when dealing with temporally sequential observations. Similarly a correlation structure should be expected when dealing with geographically distributed observations. In fact, both of the illustrative case studies presented in applications section of this chapter, potentially exhibit a spatial correlation structure.

When correlations between observations exists, *p* values will be biased. Fortunately, the A3 method contains a built-in way to directly correct for these biases. The method of calculating exact *p* values in the A3 method is based on generating random data with the same properties as the original data. As a consequence, this makes it generally straightforward to adjust for well-defined correlation structures or other issues: simply replicate that correlation structure in the randomly generated data.

As an illustrative example we can generate two first order auto-correlated data series: *x* and *y*. These series are independent of each other. The auto-correlated nature of the series, however, when not corrected for, creates artificially significant *p* values when attempting to use one to predict the other (Table 8).

```
R> set.seed(1)
R> createAutoCorrelatedSeries <- function(n, r) {
+      dat <- rnorm(n, 0, 1)
+      for (i in 2:n) dat[i] <- dat[i - 1] * r + dat[i] *
(1 - r)
+      dat
+  }
R> sample <- data.frame(x =
createAutoCorrelatedSeries(100, 0.95), y =
createAutoCorrelatedSeries(100,
+      0.95))
R> reg <- lm(y ~ x, sample)
R> print.reg(reg, caption = "Biased $p$~values as a
result of auto correlation",
+      label = "genAutoCor")
```

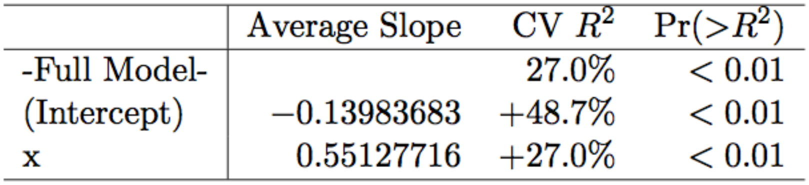
Table 8. Biased *p* values as a result of auto correlation.



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.1398	0.0168	-8.32	0.0000
x	0.5513	0.0867	6.36	0.0000

The exact method of calculating *p* values used by the A3 method can adjust for this by generating stochastic noise with the same correlation structure as the observed data. If this is not done, however, the method will also generate biased *p* values (Table 9).

Table 9. Biased *p* values in the A3 method as a result of auto correlation.

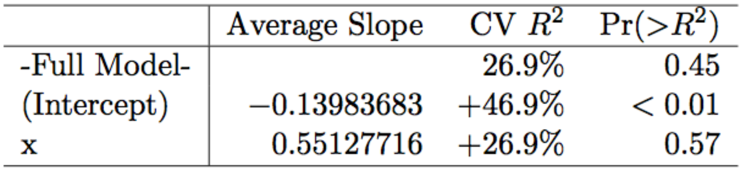


	Average Slope	CV R^2	Pr(>R^2)
-Full Model-		27.0%	< 0.01
(Intercept)	-0.13983683	+48.7%	< 0.01
x	0.55127716	+27.0%	< 0.01

The *a3* and *a3.lm* functions contain an argument *data.generating.fn* which can be used to specify the method for generating random noise. This argument takes a list of functions one for each of the independent columns in the model matrix. By default, the *data.generating.fn* is primarily a resampling based method. However, this is not valid for auto-correlated data. Our simulated data can be correctly analyzed by setting the *data.generating.fn* argument to the *a3.gen.autocor* function which generates first-order auto correlated data with the same properties as the original data. The results with corrected *p* values are shown in Table 10[12].

```
R> out <- a3.lm(y ~ x, sample, data.generating.fn =
list(a3.gen.default, a3.gen.autocor))
R> xtable(out, caption = "Corrected p values in the A3
method adjusted to account for auto correlation",
+   label = "A3AutoCorCorrect")
```

Table 10. Corrected p values in the A3 method adjusted to account for auto correlation.



	Average Slope	CV R^2	Pr(>R^2)
-Full Model-		26.9%	0.45
(Intercept)	-0.13983683	+46.9%	< 0.01
x	0.55127716	+26.9%	0.57

Discussion and Conclusions

As a thought experiment, we can conceptualize the space of all predictive models as a mountain range. Each point in the range represents a different model and the height of the range at that point corresponds to the accuracy of that model given a

[12] Please note that in this table, the intercept has a higher added R^2 than the R^2 for the overall model. This simply indicates that without the intercept term, the model is actually worse than the null model. In fact, if we remove the independent variable from the model, we can see the R^2 value will be 0, as we are left with just the intercept which is the definition of the null model.

prediction task. Each peak in the range might consist of a single type of model. For instance, one peak might correspond to Linear Regressions, another peak to Support Vector Machines, yet another peak to a set of mechanistic models, and so on. What is often done in practice when building models for prediction or inference is to explore only one peak in this range of mountains. Whether it be just Linear Regressions or mechanistic models, researchers and practitioners often only explore a single model family and then make broad conclusions about prediction or inference based on the results of this limited analysis.

Conclusions drawn from such a narrow exploration of an infinitely large model space cannot but be viewed with skepticism. In practice, it is almost impossible to say *a priori* for an unknown data generating process that a, for instance, Linear Regression model will do the best job of available algorithms in approximating it. It is imperative that practitioners attempt to further explore the model space beyond a single peak -- beyond a single model family. The **A3** package facilitates this exploration by defining an adaptable algorithm and reporting format that allows the direct comparison of results between different predictive model families.

When inferences and predictions drawn from models affect policy and scientific decisions, it is of great importance that the most predictive modeling techniques available for a given application be used. The two worked applications have demonstrated the use of the **A3** package to facilitate the exploration of the model space in two example prediction and inference tasks (one primarily illustrative, one extending published work from the journal Science). In both cases, it was a straightforward process to obtain significantly more predictive models (10%–15% additional explanation of the squared error) compared to the Linear Regression models simply by exploring one or two additional model families. More importantly, the more predictive models revealed altered inferential conclusions. Variables that were not statistically significant in the Linear Regression model became statistically significant in the more predictive Random Forest models, and vice versa[13]. Were the Random Forest models right and the Linear Regression models wrong in these cases? Of course not; clearly neither perfectly reflects reality. However, given that the Random Forest models in both these cases have significantly more predictive accuracy, it is impossible not to give more weight to their inferential conclusions compared to the Linear Regression models in these applications.

References

[13] In the applications in this paper, the Random Forest models performed the best and the Linear Regression models performed the worse in both cases. This should not be viewed, however, as a critique of Linear Regression in general or an argument that Random Forest is a better modeling approach. For other data sets and applications it is quite possible that a Linear Regression model would outperform the Random Forest model according to the results from the A3 method. What this is a strong argument for, however, is the need to explore different model families when doing model based prediction or inference.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(1), 5–32.

Breiman, L., Friedman, Olshen, & Stone. (1984). *Classification and Regression Trees*. Wadsworth International Group.

Burnham, K. P., & Anderson, D. R. (2002). *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer-Verlag.

Cortes, C., & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, *20*(3), 273–297.

Dahl, D. B. (2013). *\pkg{xtable}: Export Tables to \LaTeX or HTML*.

Falk, R., & Greenbaum, C. W. (1995). Significance Tests Die Hard: The Amazing Persistence of a Probabilistic Misconception. *Theory & Psychology*, *5*(1), 75–98. doi:10.1177/0959354395051004

Frank, A., & Asuncion, A. (2010). *UCI Machine Learning Repository*. Irvine, Ca: University of California, Irvine, School of Information and Computer Sciences.

Haller, H., & Krauss, S. (2002). Misinterpretations of Significance: A Problem Students Share with Their Teachers. *Methods of Psychological Research Online*, *7*(1), 1–20.

Harrison, D., & Rubinfeld, D. L. (1978). Hedonic Housing Prices and the Demand for Clean Air. *Journal of Environmental Economics and Management*, *5*(1), 81–102.

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

Loftus, G. R. (1993). A Picture is Worth a Thousand p Values: On the Irrelevance of Hypothesis Testing in the Microcomputer Age. *Behavior Research Methods*, *25*(2), 250–256.

Maestre, F. T., Quero, J. L., Gotelli, N. J., Escudero, A., Ochoa, V., Delgado-Baquerizo, M., et al. (2012). Plant Species Richness and Ecosystem Multifunctionality in Global Drylands. *Science*, *335*(6065), 214–218. doi:10.1126/science.1215442

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., & Leisch, F. (2012). *\pkg{e1071}: Misc Functions of the Department of Statistics (e1071), TU Wien*.

Muñoz, I. D., & van der Laan, M. (2012). Population Intervention Causal Effects Based on Stochastic Interventions. *Biometrics*, *68*(2), 541–549. doi:10.1111/j.1541-0420.2011.01685.x

Nickerson, R. S. (2000). Null Hypothesis Significance Testing: A Review of an Old and Continuing Controversy. *Psychological Methods*, *5*(2), 241–301.

R Core Team. (2012). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://www.R-project.org/

3. *Home Range Estimation through Dynamic Competitive Estimator Selection*

Introduction

The concept of the home range is an important one in ecology and it will be the focus of this chapter and the next. Where the prior chapter of this dissertation focused on developing a very generally method; this chapter and the next are more specific, exploring the issue of parameter estimation and model selection in terms of density estimation. Of which, home range estimation is a specific application. In the literature, there are numerous references to the concept of home range including those by Charles Darwin, e.g. "…most animals and plants keep to their proper homes, and do not needlessly wander about; we see this even with migratory birds, which almost always return to the same spot" (Darwin, 1861). Despite an extensive body of work, the precise definition of the home range remains slightly vague. The following definition of a home range, put forward in 1943, is still used, "that area traversed by the individual in its normal activities of food gathering, mating, and caring for young. Occasional sallies outside the area, perhaps exploratory in nature, should not be considered as in part of the home range" (Burt, 1943). Operationally, animal home ranges are generally represented using either utilization distributions – a probability density function representing an animal's frequency of inhabiting a given region – or isopleths – highest density regions of the utilization distribution containing a specified fraction of the density.

Home ranges can be estimated from observations of the location of animals within a population. Locations are typically recorded using radio-telemetry or GPS receivers. The resulting presence-only collection of location observations may be converted into a home range using a variety of estimators. Primarily in practice, two techniques are used in the estimation of animal home ranges from animal location observations: the minimum convex polygon and kernel density estimation (Laver & Kelly, 2008). An illustration of these two methods is shown in Figure 7.

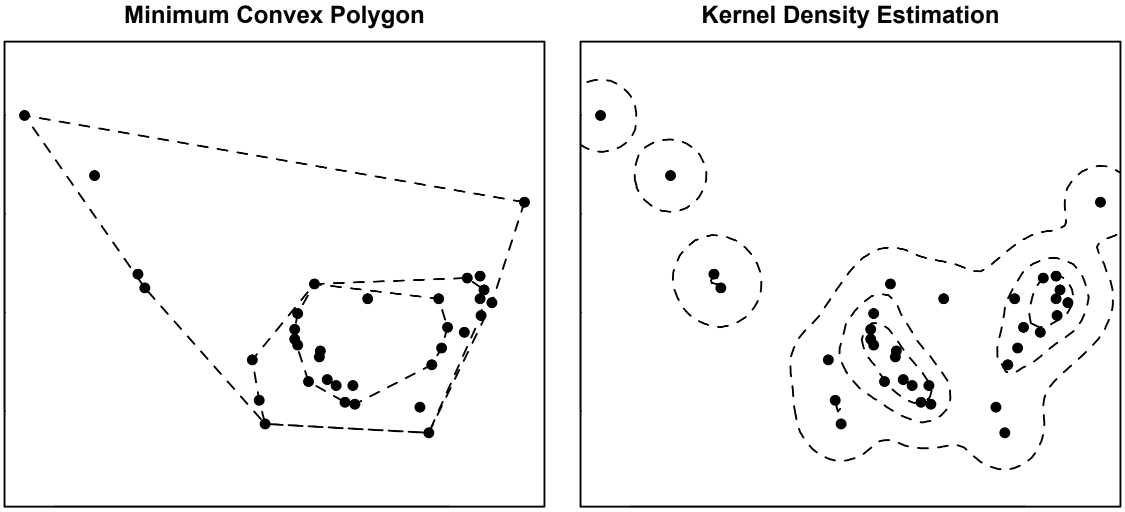


Figure 7. Minimum convex polygon (points included = 100%, 80%, 50%) and kernel density estimates (isopleths = 99%, 80%, 50%) of a Blanding's turtle's home range given observation locations.

The minimum convex polygon (MCP) is the smallest convex polygon containing a set of observations. It can be thought of as estimating the 100%-isopleth, i.e., the full area utilized by the animal. The MCP estimator is sometimes defined with a parameter for the exclusion of a percentage of outlier points. The MCP is a crude estimator that fails to account for holes or non-convexity in a species' utilization of space. It is simple to calculate and readily understood, however, and also has a long history of utilization making it useful as a comparative tool.

Kernel density estimation (KDE) is generally a much better technique of estimating home ranges. KDE is a general estimation technique (that is widely used outside of the field of ecology) that functions by fitting a probability density function – a kernel – on top of each location observation and then averaging these kernels to obtain an estimate of the utilization density (Silverman, 1986). Excluding an inherent assumption of independence between observations, KDE is well suited to estimating an animal's utilization distribution and it was first proposed for this use in the 1980's (Worton, 1989). KDE creates an entire probability density distribution function model of an animal's utilization of space that may subsequently be converted into isopleths representing the most frequently used areas by species. Given a probability α, the α-isopleth represents the region of highest usage by the animal so that 100 × α% of the observations of that animal will fall within that region. A number of studies in the ecological literature have explored the uses and properties of KDE as applied to home ranges (e.g., (Blundell, Maier, & Debevec, 2001), (Fieberg, 2007), (Gitzen & Millspaugh, 2003), (Seaman et al., 1999), (Seaman & Powell, 1996), (Worton, 1995)).

In addition to MCP and KDE, other home range estimators have been proposed. These techniques include, but are not limited to, those based on α-hulls (Burgman & Fox, 2003), Fourier transformations (Anderson, 1982), characteristic hulls (Downs

& Horner, 2009), local convex hulls (LoCoH) ((Getz & Wilmers, 2004), (Getz et al., 2007)), clustering (Kenward, Clarke, Hodder, & Walls, 2001), and Brownian bridges (Horne, Garton, Krone, & Lewis, 2007). While a thorough review of these techniques is beyond the scope of this chapter, one is discussed for purposes of illustration: LoCoH. LoCoH functions by generating a convex hull for each observation out of the set of nearest neighbors to that observation. The rules for selecting the quantity of nearest neighbors vary between the different variants of LoCoH. For instance, in *k*-LoCoH, the tuning parameter *k* directly specifies the number of nearest neighbors selected for each observation. Conversely, in *r*-LoCoH, all observations within a radius of the parameter *r* from the source observation are used to create the hull. In *a*-LoCoH, the maximum number of observations such that the sum of their distances from the source observation is less than or equal to the parameter *a* are used to create the hull. The hulls are then combined from smallest to largest to create isopleths. The exact rules of this combination vary between LoCoH variants, but in one variant the α-isopleth can be defined as the area created by the union of the α × *n* hulls with greatest density of points.

Of natural interest is this selection of the optimal home range method for a given data set. This includes not only selection between competing home range methods and algorithms, but also the selection of tuning parameters for a given method. One notable result in this arena is provided by (Horne & Garton, 2006b) who developed an information theoretic approach to home range model selection. Such a technique, however, requires a probability density distribution for the home range, something not all home range estimators generate. Notably, some methods, such as LoCoH or MCP, will have a likelihood of zero for any set of tuning parameter values due to the nature of their construction[14]. Thus preventing an attempt to directly apply information theoretic, likelihood based techniques.

Before proceeding with this chapter, some important notation needs to be introduced that is used in the chapter. The vector of *n* point location observations will be denoted $\boldsymbol{X}$ with the *i*th observation being denoted $\boldsymbol{X}_i$. The observations will assumed to be independent and identically distributed according to a utilization distribution density: $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n \sim D$. The highest density region of *D* containing 100×α% of the density is the α-isopleth and is denoted R^α. R^α is in effect a parameter describing *D*. Given an arbitrary region *R*, the probability of a new observation sampled from *D* being contained in that region is denoted *p*. Estimates of *D*, R^α, and *p* are denoted $\widehat{D}$, $\hat{R}^{\alpha}$, and $\hat{p}$ respectively. Given a set of observations distributed according to *D* and a chosen α, we denote the function $\psi(\mathbf{X}, \alpha)$ an estimator of the parameter R^α of *D* so that $\psi(\mathbf{X}, \alpha) = \hat{R}^{\alpha}$. For some estimators, such as KDE, which can

[14] The convex hull generation technique necessitates that if a boundary point is removed from the data set, as occurs during cross-validation, that point will not be included in a convex hull generated on the subsetted data. Such a boundary point will then be outside the home range and its likelihood for sharp-edged, convex hull-based home ranges will consequently be 0. The likelihood for an entire home range is the product of the likelihood of the independent observations. Thus the likelihood for the entire home range estimate will also be 0 given that any single one of the observations has a likelihood of 0.

estimate full densities; we will use ψ without the α specified to denote the estimation of the density: $\psi(\mathbf{X}) = \widehat{D}$

Each method for home range construction can be considered an estimator of R^{α}. Thus we have ψ_{MCP}, ψ_{KDE}, $\psi_{\text{a-LoCoH}}$ and so forth. Furthermore, most methods have one or more tuning parameters that control the behavior of the estimator. Each combination of method and associated tuning parameters can be considered as a discrete estimator. For instance, in KDE a bandwidth parameter, often denoted h, controls how strongly the density estimate is smoothed. As a consequence, there are multiple estimators based on KDE with different tuning parameters such as $\psi_{\text{KDE}}{}^{\text{h=0.5}}$, $\psi_{\text{KDE}}{}^{\text{h=1}}$, $\psi_{\text{KDE}}{}^{\text{h=2}}$ and so on. For estimators that contain internal methods for estimating tuning parameters, we can consider the entire combined algorithm as an estimator. For example, the estimator created using Least Squares Cross Validation as a technique to determine KDE's bandwidth parameter can be denoted as $\psi_{\text{KDE}}{}^{\text{h= LSCV}}$.

This chapter proposes a new estimator; which, for the purposes of this dissertation, is referred to as Isopleth Curve Estimation (ICE): ψ_{ICE}. It differs from other estimators, such as KDE or LoCoH, in that it does not itself contain any mechanisms for internally proposing home range isopleths or densities. Instead, it selects between home range isopleths proposed by external candidate estimators which may include methods like KDE or LoCoH. ICE can conceptually be thought of in two different ways: either as a tool for home range model selection or as a hybrid home range estimator comprised of individual, competing estimators. Figure 8 displays a conceptual overview of the technique. By selecting between candidate estimators, ICE can provide a higher level of accuracy in aggregate than any single candidate. ICE shares significant conceptual similarities and overarching goals with the A3 method described the previous chapter as they both use predictive accuracy as the basis for model selection.

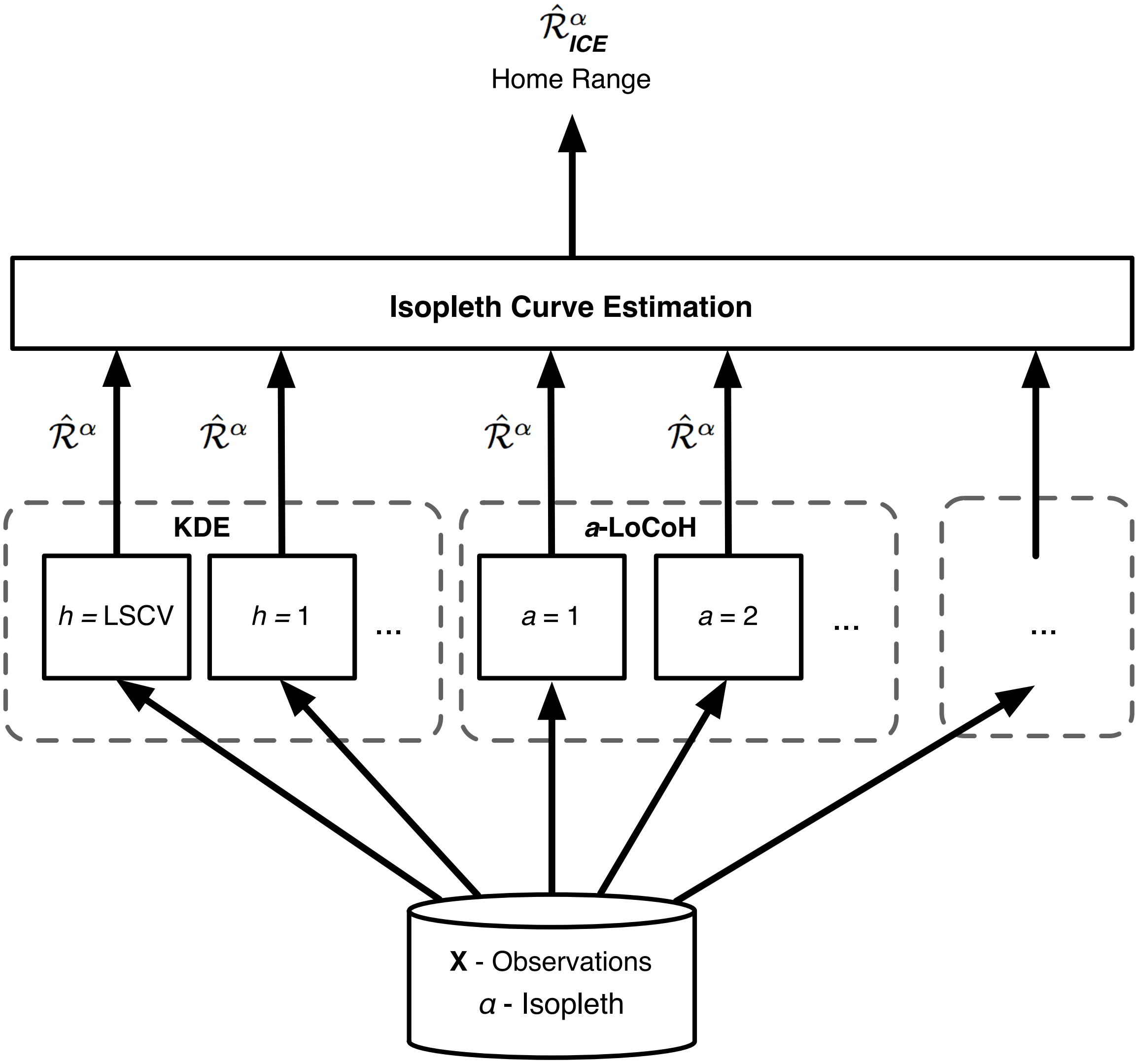

$\hat{\mathcal{R}}^{\alpha}_{ICE}$
Home Range
Isopleth Curve Estimation
$\hat{\mathcal{R}}^{\alpha}$
$\hat{\mathcal{R}}^{\alpha}$
$\hat{\mathcal{R}}^{\alpha}$
$\hat{\mathcal{R}}^{\alpha}$
KDE
h = LSCV
h = 1
...
a-LoCoH
a = 1
a = 2
...
...
X - Observations
α - Isopleth

Figure 8. Isopleth Curve Estimation overview: an estimator of estimators. A group of candidate estimators uses a set of location observations to estimate an animal's home range. ICE is then used to select between these competing home range estimates to obtain a final home range estimate.

Since the ICE estimator does not generate its own home range regions, it depends completely on the selection of candidate estimators that are fed into it. If all the candidate estimators perform poorly, then the result of the ICE estimator will also perform poorly. Conversely if the candidate estimators are of a high quality, the results of the ICE estimator will also be of a high quality. By and large, however, any given estimator might vary in its performance. For instance, one can hypothesize that KDE will generally perform better on data with a gradually changing, smooth density than LoCoH. Conversely, LoCoH might perform better on data with sharply changing densities and hard boundaries. By using a hybrid estimator – an estimator of estimators – the technique can select between different techniques based on the empirically properties of the observations themselves.

ICE has a number of parallels to the discrete Super Learner (Van Der Laan & Rose, 2011). The discrete version of the Super Learner selects from within a library of candidate estimators to find the one best suited to a given predictive task. A possibly more commonly used version of Super Learner creates a weighted combination of algorithms instead of selecting the single best as is the case for the discrete Super Learner. The library of estimators can include many diverse algorithms including staples such as linear regression or more novel techniques such as RuleFit. A key theoretical result underpinning the Super Learner and potentially also ICE, is that the result of the cross validation selection process converges to the oracle estimator given certain non-restrictive conditions (Dudoit & van der Laan, 2005). This means that asymptotically, the Super Learner will perform as well as the best available candidate estimator. Although this chapter does not attempt to formally show that these asymptotic results apply to ICE, they are still encouraging for the general approach.

In this chapter, ICE is developed as novel approach of home range estimation based on the dynamic selection between competing estimators like KDE or LoCoH. This approach does not require an animal's utilization density, but is instead based on isopleths. Selecting home ranges based on an isopleth-centric criterion allows the comparison and selection of home ranges between a wider ranges of methods than those explorable by the information theoretic technique of (Horne & Garton, 2006b). Moreover, when the development of an isopleth is the final goal of a study (as may be the case when assessing the amount of region required to contain a fixed fraction of animal space utilization), a method focusing only on the accuracy of that specific isopleth may be more accurate for that usage case than one attempting to match entire densities. The rest of this chapter first describes the proposed approach in detail and then applies it to both simulated and empirical data sets.

Materials and Methods

Isopleth Curve Estimation

Isopleth Curve Estimation is based on the definition of an isopleth as being the region R^{α} such that:

1. If x ~ D, then p($x \in \hat{R}^{\alpha}$) = p = α.

2. Given the set of all regions such that (1) is true (which will be infinite in size), R^{α} is the region in that set with the smallest area.

When ICE runs, it asks its candidate estimators to generate estimates of R^{α}. Depending on the number of candidate estimators used, the size of the set will vary. Given this finite set of estimators, several fundamental issues arise in directly applying the above definition. First, the probability calculated in (1) generally cannot be known precisely for a given region and instead must be estimated. Second, it is probable that some, if not all, of the candidate estimators' proposed regions will contain more or less than α × 100% of the density. Lastly, we have the fundamental question: Given our finite estimator set, the variability in estimating p, and the fact that some of estimators will likely contain more or less of α × 100% of the density, how do we select between the different candidate estimators to find the one that is closest to the true R^{α}?

This section's remaining subsections, present the conceptual framework to understand an algorithmic selection process to choose between candidate estimators. First we define and describe what we call the "isopleth curve." Then, given this curve, we discuss the selection of the optimal candidate estimator within a set candidate estimators. Finally, we address the issues of the estimation both of the isopleth curve and the estimation of p for a given region.

The Isopleth Curve

Given an arbitrary bivariate density D, we can construct regions (R) containing a portion of the density (Figure 9.a). These regions may have arbitrary geometries that may or may not be continuous. Two properties of each region are of particular interest. The first is the area of the region, which for most regions will be trivial to calculate (and will be so for all regions expected to be encountered during home range estimation). The second property of interest in the regions is the fraction of the density D that falls within the regions (denoted p). If D is known, this can be determined simply by integrating D under R[15].

[15] For our purposes, D is unknown. If R was obtained without access to $\mathbf{X}$, then a good estimate of p would simply be the number of observations falling beneath R divided by the total number of observations. If $\mathbf{X}$ must be used in the construction of R, as is common in home range applications, then resampling based techniques may be used as we will show.

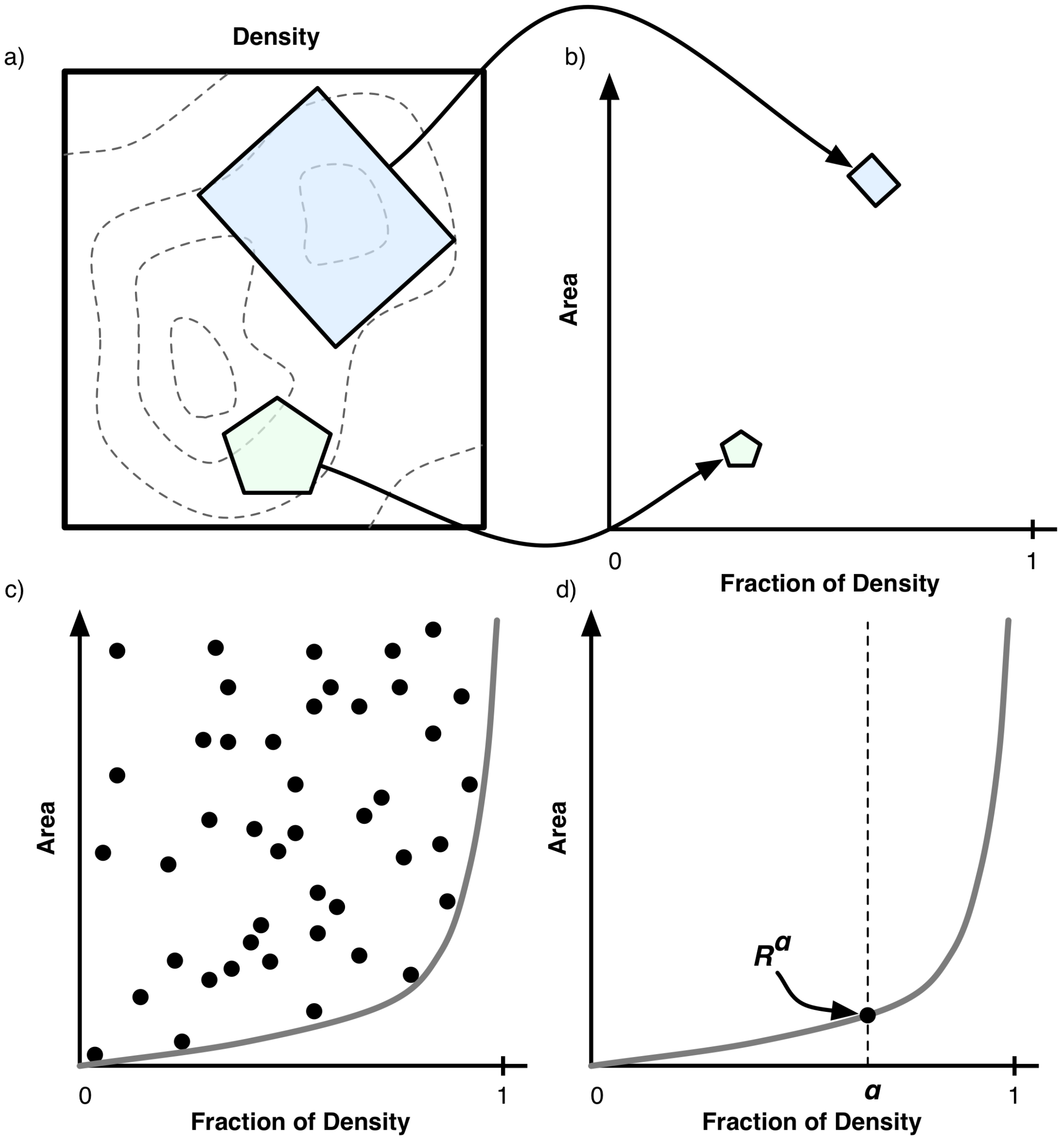


Figure 9. Illustration of the identification of the isopleth curve and R^{α}. a) An arbitrary bivariate density with two arbitrary regions drawn on it. b) The plotting of the two regions from (a) on a bivariate plot showing their areas and the fraction of the density they contain. c) Illustration of a multitude of different regions plotted on the same two axes. Further illustrates the isopleth curve bounding the possible combinations of areas and *p*. d) Given the isopleth curve the *p* associated with a given isopleth level α is straightforward to identify in the bivariate plot.

Once obtained, the area of a region and its associated *p* may be plotted on a bivariate scatterplot. Figure 9.b illustrates the placement of the two example regions from Figure 9.a on the bivariate plot where the horizontal axis is the region p and the vertical axis is the region area. Any region, including discontinuous regions may be plotted in like manner. Figure 9.c conceptually illustrates what might occur if we created dozens of regions and then plotted their associated areas and *p*. Figure 9.c illustrates an important fact: There is a lower bound to the location of these points. For a given isopleth level α, a minimum area is required in order to contain that fraction of the density *D*. Any region with a smaller area will simply not be able to encompass α × 100% of the density no matter how it is configured. As *p* increases, the requisite area required to encompass that *p* also necessarily increases.

We term this lower bound the "isopleth curve." This term comes from a key property of the curve. If a vertical line is constructed at α, the intersection between this line and the isopleth curve marks the area of R^{α} (Figure 9.d). Since the α × 100%-isopleth is by definition the smallest region that contains α × 100% of the density, this property naturally follows. As we demonstrate in the next section, the isopleth curve is an important tool for selecting between home range estimates.

Selecting the Best Candidate

Assuming knowledge of the isopleth curve and a set of estimates of the home range, how do we select the $\hat{R}^{\alpha}$ which is closest to the true R^{α}? The first step in answering this question is to define what we mean as "closest". Look at Figure 9; it would be tempting to define "closest" as some form of scaled Euclidean distance in the area/*p* plane between an estimate and the intersection between the isopleth curve and the vertical line located at α. From an implementation standpoint this is straightforward, but how would we interpret such a measure of distance in terms of the original regions? Such an interpretation is not clear and more vexing, Euclidean distance is not a good practical measure, as the direction of the deviation in this plane has significance in terms of the underlying region geometries, a point we shall see momentarily.

Rather than jumping straight to an distance metric in terms of the area/*p* plane, it is more fruitful to discuss first what distance means in terms of the original home range regions, and then determine whether the isopleth curve can help us apply this definition within the area/*p* plane. Given two regions, it is straightforward to define the distance (or error) between the two regions as the area of the symmetric difference between the regions (Figure 10). This metric is intuitive, easy to interpret and understand, and also simple to calculate. The metric can be criticized in that it does not penalize the geometric distance between non-overlapping regions, and instead applies the same penalty no matter how close or far apart the non-overlapping regions are. Nevertheless, it is not clear that non-overlapping regions should be penalized by geometric distance and in any event the benefits of this metric may outweigh this issue. Moreover, in usage, home range estimates should generally be close together such that this is not a practical issue.

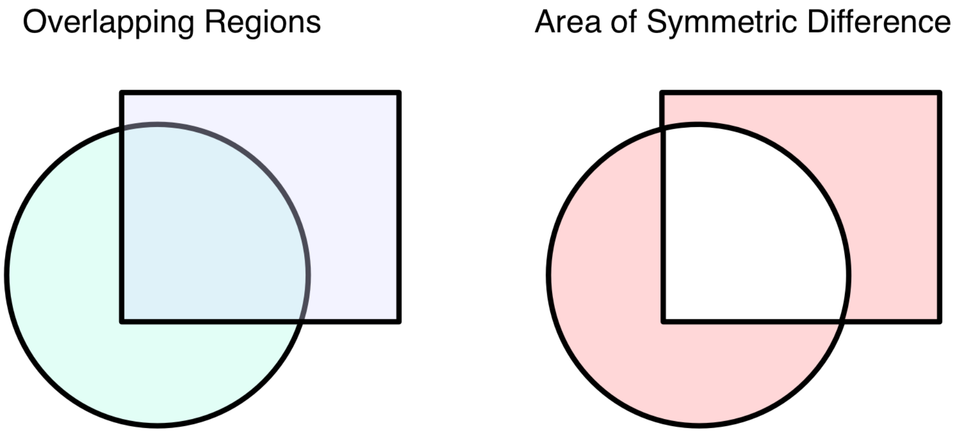


Figure 10. The area of the symmetric difference between two regions is taken as the distance or error between the regions.

Given the isopleth curve, the area of the true α-isopleth is known, but we have no other information about its geometry. Given this, it might seem difficult or even impossible to apply the symmetric difference distance metric to compare and select between home range estimates based solely on the location of regions in relation to the isopleth curve. However, further consideration of the geometry of the area/p plane reveals that the curve does contain valuable information that allows users to make inferences about the symmetric difference error for a given home range estimate.

First, consider the case where a home range estimate is located directly on the isopleth curve. Here the value of the distance metric can be calculated precisely and quickly. It is simply the difference in areas between the estimate and the true home range area. This is known because both regions are isopleths as they fall on the isopleth curve. Thus, one necessarily contains the other completely. In other words, the distance as defined as the symmetric geometric difference between the regions is solely the difference between their two areas.

If however regions lie in the interior of the isopleth curve, it is impossible to precisely calculate the distance metric. For points inside the isopleth curve, multiple distinct regions collapse to the same point and thus we cannot uniquely identify the distance from the true isopleth for any point on the interior of the curve. Nevertheless, we can identify a range of distances within which the true distance must fall. To see this consider two home ranges, one with an area of 10 square kilometers and other with an area of 30 square kilometers. What is the distance between these two home ranges in terms of the area of the symmetric geometric difference between them?

Clearly, we do not have enough information to answer this question precisely, but we can say the following. The worse case (i.e. greatest distance) is when the two home ranges do not overlap at all. In this case, the distance would be 40 square kilometers (10 square kilometers plus 30 square kilometers). The best case (i.e.

minimum distance) is when the larger home range completely contains the smaller home range. In this instance the distance would be 20 square kilometers (30 square kilometers minus 10 square kilometers). Thus, the true distance must be between 20 and 40 square kilometers inclusive. We cannot pinpoint in this range where the true distance is, but we know it is bounded by the given upper and lower bounds.

The existence of the isopleth curve enables us to do even better with our definition of this boundary. The geometry of a specific isopleth curve allows us to narrow the upper and lower bound on the distance for a given home range beyond using the trivial calculations demonstrated in the above example. The calculations to do so are non-intuitive and presented in the detailed Appendix A. Figure 11 illustrates the results of these calculations for an illustrative isopleth curve and an isopleth level of 0.8. As seen in the figure, the boundaries on the distance do not increase consistently as one moves away from the intersection between the isopleth curve and the vertical line located at α. Conversely and interestingly, movement towards smaller fractions of the density enclosing and smaller areas initially results in the lowest increase in distance for a given linear distance moved in the area/*p* plain.

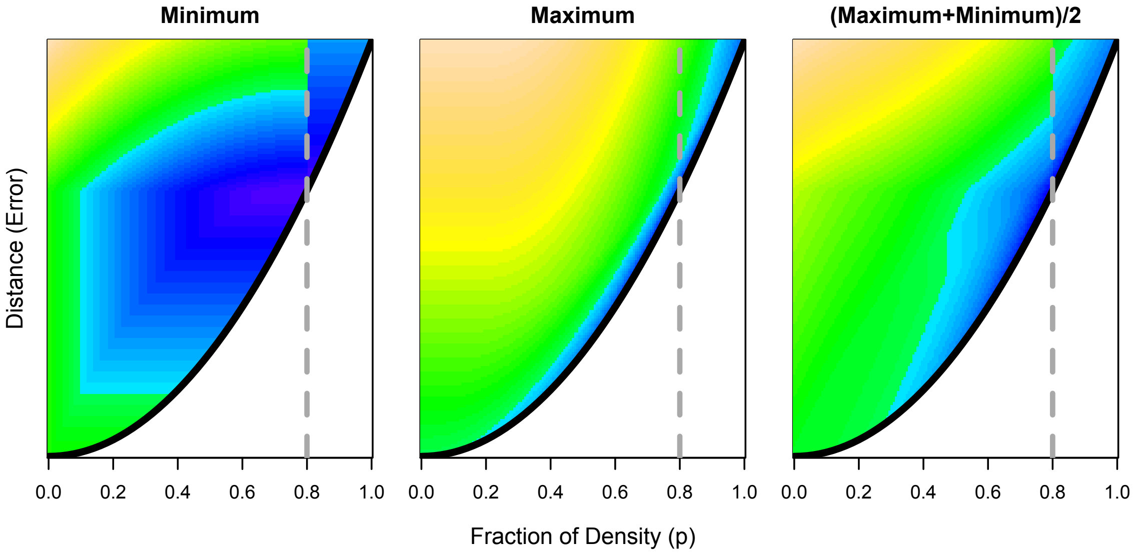


Figure 11. Distances for an illustrative isopleth curve and an α of 0.8. The coloring indicates distance from the true isopleth and ranges from purple (close fits) to orange (distant fits).

Estimating the Isopleth Curve

In practice, the true form of the isopleth curve will be unknown for a given density and it must be estimated from the data. We can state several general properties of the curve, nonetheless, that will be consistent between densities:

1. The curve passes through the data point of zero area and zero *p*. Simply put, for a well-defined home range a region of no area must encompass none of the density.
2. The curve is monotonically increasing. As the fraction of density enclosed by an isopleth increases, the isopleth must necessarily increase in size.
3. The slope of the curve increases as the fraction of the density contained by the curve increases. As α increases, the isopleth must grow increasingly large to encompass an additional given amount of density[16].

These three properties are important, but, on their own, they do not provide us enough theory or information to estimate an unknown density's isopleth curve from a given set of observations. Fortunately, we have access through our candidate estimators to a crucial tool for reconstructing the isopleth curve: the home range estimates themselves. Each home range estimate is in actuality an attempt to estimate a point on the isopleth curve. We can use our candidate estimators not only to estimate the isopleth we are interested in, but also to estimate isopleths for other α's along the curve. The resulting estimates can be combined with the point (0, 0) (given property (1) from above), and then used to construct a convex hull defining the isopleth curve. This convex hull will satisfy the properties (1) and (2) of an isopleth curve. Each line segment on the border of the curve will have a constant slope so item (3) is not strictly satisfied, but between line segments the slope will be increasing as required by (3).

In general, there will be error in isopleth estimates which becomes error in the estimate of the isopleth curve. Fortunately, if the candidate estimators are guaranteed to converge to the true isopleths as the sample size increases, the point defining the curve can also be guaranteed to converge to the true curve as the sample size increase. Since Least Squares Cross Validation KDE is an estimator which is guaranteed to asymptotically converge to the true density (Silverman, 1986), its inclusion as a candidate estimator is sufficient to guarantee the isopleth curve estimation procedure presented here converges asymptotically.

[16] This last point will not be true for densities with "flat" regions consisting of a constant likelihood. However, such densities will not have unique isopleths in any case for α's corresponding to the flat region in the density. This issue will not be considered further.

Regardless of asymptotic behavior, for finite sample sizes the results of this procedure are biased to be too high (i.e. area values of points on the curve have a positive bias). The reason for this is simple: estimated area/*p* points will be located on the curve or in the curve's interior; they may never be below the curve (in other words, estimates of home ranges that are larger than the optimal value may be generated, but estimates that are better than the optimal value are impossible). Thus when the convex hull is formed around these points, the bias is for the curve to be too high. This bias will be most significant for small numbers of observations. As the numbers of observations increase, the bias will be reduced.

We have developed a simple adjustment procedure to attempt to identify the magnitude of this bias and correct for it. This procedure works by first generating the isopleth curve for the full data set. Then 50% of the observations are sampled and is again estimated using this subset of the data. When the two curves are close to identical, this indicates negligible bias, as the bias would increase with a reduction in sample size. If the curve for the reduced sample size shifts upward significantly, this indicates the potential for significant bias. An elegantly simple approach to account for this is to estimate the effect of sample size on bias on then extrapolate this towards a non-finite sample size to correct for the bias. Invoking Occam's razor, for a given observation size n this relationship is assumed to be: $bias_n = k/n$ where *k* is a constant depending on the original density. *k* may be estimated with as little as two observations and this technique is straightforward to apply.

Estimating p

Without knowledge of the underlying density, *p* cannot be determined analytically and must be estimated numerically. A naive approach to estimating *p* for a given set of observations **X** and estimator ψ, is to take the fraction of observations that appear within $\psi(\mathbf{X}, \alpha)$:

$$\hat{p} = \frac{1}{n}\sum_{i=1}^{n} I_{\psi(\mathbf{X},\alpha)}(\mathbf{X}_i)$$

This estimator for *p* will however generally have a positive bias, as **X** is being used both to create the home range estimate and to assess the accuracy of that estimate. To see why, imagine the contrived example of an estimator that places an infinitesimally small disc around 100×α% of the observations. Using the above naive metric, α will equal $\hat{p}$ for that estimator. However, in reality, p should be effectively 0 as if new observations were sampled from *D* they would fall outside the infinitesimally small discs. The same phenomena of overfitting should occur to a lesser, but potentially still significant, extent in practice for actual home range estimators.

Cross validation is used to create an improved estimator for p. Briefly, cross validation is a resampling based technique that divides a data set into subsets, builds a predictor with some of the subsets and then assesses the predictor with a subset that was not used in building the estimator. This process is repeated until all the subsets are used to both build and assess the predictor in turn. In terms of estimating p, we remove a subset of observations from **X**, estimate the home range using that reduced set, and then obtain $\hat{p}$ based on the subset we removed.

A key choice in cross validation is selecting the size of the subsets. The subset size that will result in the least biased results (though not the lowest variance and not necessarily the minimum error) is a subset of one, a method also known as Leave One Out Cross Validation (LOOCV). Here, we use LOOCV to estimate p, employing the notation $\boldsymbol{X}_{(-i)}$ to denote $\boldsymbol{X}$ sans the ith observation:

$$\hat{p} = \frac{1}{n} \sum_{i=1}^{n} I_{\psi(\mathbf{X}_{(-i)}, \alpha)}(\mathbf{X}_i)$$

For large data sets, a subset size of one may prove to be prohibitively costly in terms of computational resources. In this case, a larger subset size such as 10% of the data (10-fold cross validation) or 5% of the data (20-fold cross validation) could be used. The larger the subset, however, the more biased $\hat{p}$ will be.

Candidate Estimators

In this chapter, three families of candidate estimators are used: MCP, KDE and *a*-LoCoH. As discussed in this chapter's introductory section, many other home range estimators exist and their inclusion as candidate estimators should, where time and resources are no constraint, improve the resulting hybrid estimator. However, to keep this study tractable from the standpoint of computational cost and complexity, the number of candidate families was limited to three. KDE and MCP were chosen due to their wide use. *a*-LoCoH was chosen as it is hull-based method that can provide good fits yet is quite dissimilar to KDE thereby providing a good contrast to that technique.

MCP

The MCP home range estimator is denoted ψ_{MCP}.

KDE

The candidate KDE estimators in this chapter employed bivariate Gaussian kernels. Two distinct estimators were used that differ in how they selected the bandwidth smoothing parameter h.

Bivariate Normal Reference

This estimator ($\psi_{\text{KDE}}{}^{\text{h=Ref}}$) is derived to provide an optimal bandwidth assuming the unknown distribution of the observations is a bivariate normal density

(Silverman, 1986). In this method, h is determined based on the sample size and dispersion of the data:

$$h^{\text{Ref}} = \frac{1}{2}(\sigma_x + \sigma_y)\ n^{-1/6}$$

Least Squares Cross Validation

This estimator ($\psi_{\text{KDE}}^{\text{h=LSCV}}$) minimizes the mean integrated squared error between the estimate of the utilization distribution ($\widehat{D}$) and the true distribution (D):

$$h^{\text{LSCV}} = \underset{h>0}{\arg\min} \int\int (\psi_{\text{KDE}}^{h=h}(\mathbf{X}) - \mathcal{D})^2\ dx\ dy$$

Given that D is unknown, this equation is not directly solvable. It can be shown (Silverman, 1986), however, that minimizing the following equation, which can be done directly from the observations, is equivalent:

$$h^{\text{LSCV}} = \underset{h>0}{\arg\min} \int\int \psi_{\text{KDE}}^{h=h}(\mathbf{X})^2\ dx\ dy - \frac{2}{n}\sum_{i=1}^{n} \psi_{\text{KDE}}^{h=h}(\mathbf{X}_{(-i)})(\mathbf{X}_i)$$

a-LoCoH

In determining the value of the tuning parameter a, (Getz et al., 2007) recommend two approaches. The first is a holistic approach where home range estimates are generated and inspected for a range of tuning parameter values. It is recommended that the researcher examines both the raw isopleth estimates and accompanying summary statistics, such as isopleth area, in order to identify critical boundary values of a that result in significant changes in home range estimates. This knowledge is used to subjectively choose a value for the a parameter. The second approach is a "rule of thumb" method to generate a “ballpark” acceptable value for a from the observations. In this approach a is set to the maximum distance between any two observations in the data set. Although ad hoc and sensitive to outliers, this method generally appears to generate adequate values for a in many cases. The rule of thumb a-LoCoH estimator ($\psi_{\text{a-LoCoH}}^{\text{a=Thumb}}$) is used as one of the candidate estimators in this chapter.

Previously, no statistical technique had been developed to choose the value of a. That said, in addition to using ICE to select between different families of candidate estimators, we may also use it as a tool to select between tuning parameter values for a single type of candidate estimators. Applying ICE solely to a-LoCoH to select between values of a, we obtain a second a-LoCoH candidate estimator we denote $\psi_{\text{a-LoCoH}}^{\text{a=ICE}}$.

Assessing Estimator Accuracy

For a given known data generating process, α, ψ and observation sample size n; the following steps are taken to assess the estimators' performance:

1. A sample of *n* observations are generated according to the data generating process.
2. The α-isopleth estimate $\hat{R}^{\alpha}$ is created using ψ.
3. The error between $\hat{R}^{\alpha}$ and the known R^{α} of the data generating process is calculated.
4. Steps 1-3 are repeated 40 times in order to account for the inherent stochasticity in the process and the resulting distribution is aggregated and summarized.

In order to assess the errors of the estimators on simulated data, the area of the symmetric difference between the estimated isopleth $\hat{R}^{\alpha}$ and the true isopleth R^{α} is taken and normalized by the area of the true isopleth.

Computation

Computational algorithms including ICE were implemented primarily using the R statistical software environment (R Core Team, 2012). The adehabitatHR package in R was used, with modifications, for the MCP and KDE estimators (Calenge, 2006). The a-LoCoH estimator was coded independently and was implemented in C++ and integrated into R using the Rcpp package (Eddelbuettel & Romain, 2011).

Data

Both simulated and empirical data are used to assess the estimators. Simulated data allow for the analysis of the estimators using objective error criteria. Empirical data are used to illustrate the estimators’ usage in practice and discuss their qualitative results.

Simulated Data

The simulated data enable us to quantitatively assess the proposed estimator. Two different forms of simulated data were employed:

- Independently sampled points from a mixture of bivariate normal densities.

- Independently sampled points from manually specified bivariate densities.

Both data generating processes, described further below, were defined such that true isopleth levels could be determined for comparison with the estimated value of the isopleths. During analysis, multiple sets of simulations were carried out for each of the two processes. In each iteration, a separate set of points was sampled from the underlying density or process. Although these simulation processes are crude approximations of true movement data and animal behavior, they are similar to those used in other studies (e.g. (Getz et al., 2007), (Horne & Garton, 2006a), (Worton, 1995)). It is possible of course to refine by adding increased realism and assumptions of animal movement. Such modifications would, however, limit the applicability of the simulated data to increasingly specific domains of animals raising further issues of realism. Together, these data generating processes replicate several key properties of empirical data and are sufficient to assess the estimators. Figure 12 shows samples from the two different processes and the following sections detail them.

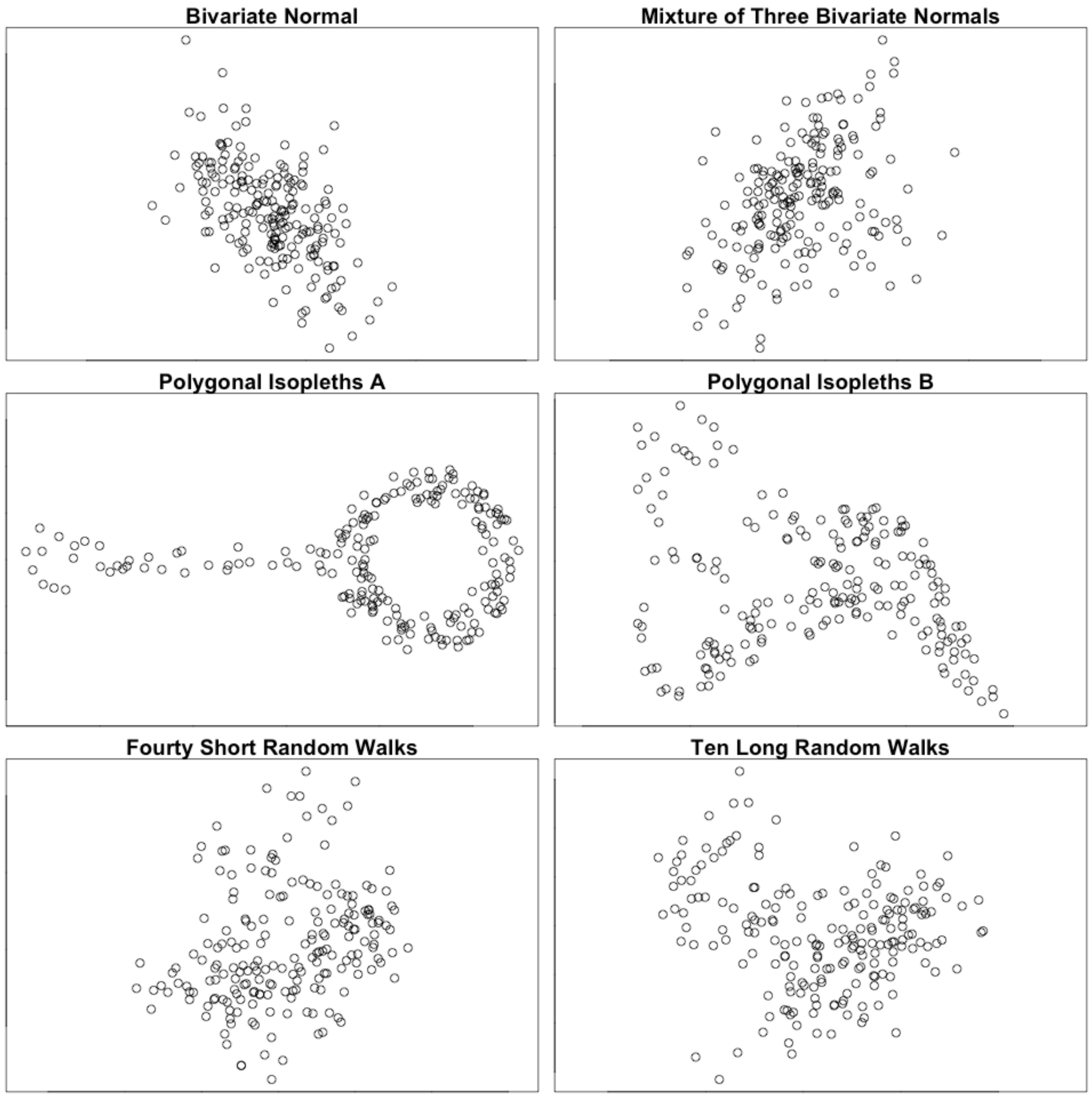


Figure 12. Illustrative simulated data sets (*n*=200). Simulations were repeated multiple times with varying random seeds leading to different samples of the underlying densities.

Mixture of Normals

The following function provides the density of the bivariate normal where **μ** is a two-element vector of means, and **Σ** is the distribution's covariance matrix:

$$f(\boldsymbol{x};\boldsymbol{\Sigma},\boldsymbol{\mu}) = \frac{e^{-1/2(\boldsymbol{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}}{2\pi|\boldsymbol{\Sigma}|^{1/2}}$$

During the simulation study in this chapter, **μ** and **Σ** are both selected stochastically, leading to a range of normal distributions to be explored. Data are simulated both for single normal distributions and for a mixture of three normals. In the case of a mixture of *m* normal distributions, the density is given as follows where **μ** is a vector of mean vectors and **Σ** is a vector of the covariance matrices for the different distributions:

$$f(\boldsymbol{x}; \boldsymbol{\Sigma_i}, \boldsymbol{\mu_i}, i \in 1..m) = \frac{1}{m} \sum_{i=1}^{m} \frac{e^{-1/2(\boldsymbol{x}-\boldsymbol{\mu_i})' \boldsymbol{\Sigma_i}^{-1} (\boldsymbol{x}-\boldsymbol{\mu_i})}}{2\pi |\boldsymbol{\Sigma_i}|^{1/2}}$$

A closed form, analytic solution to the α-isopleth does not appear to exist for the bivariate normal distribution. In order to obtain an accurate estimate of the true α-isopleth for a given combination of normals, a fine, two-dimensional grid was constructed and the density sampled at each grid cell. The α-isopleth was found my combining the highest density cells in this grid until the region encompassed 100×α% of the density.

Polygonal Isopleths

Numerous hard boundaries exist in natural environments. For instance, a river may bound one side of an animal's home range resulting in a discontinuity of the utilization distribution density from potentially a high value on one side of the river to a value of zero on the opposite side. With continued development, the number of potentially hard boundaries increases with the proliferation of barriers to animal movement such as roads and fences. The smooth densities of mixtures of normals are not well suited to modeling this phenomenon, as utilization distribution densities in systems with hard boundaries are more likely to be characterized by discontinuities.

In order to assess the estimators in terms of densities containing hard boundaries, a second form of data generating density distributions was explored where the isopleths of the distributions were directly defined using manually specified polygons. The resulting distributions have sharp boundaries and edges as defined by the polygon boundaries. Polygons were specified for the 80%-isopleth and the 100%-isopleth of these densities and data sampled accordingly. These distributions test an estimator's ability to handle hard sharp boundaries and discontinuities in the underlying densities.

Empirical Data

Five empirical data sets are included in this chapter, which recorded the locations of five different species respectively: elk (Cassirer, Freddy, & Ables, 1992), warblers (Jette, Hayden, & Cornelius, 1998), black bears (Sager, n.d.), turtles (Johnson, n.d.), and hawks (Edward Garton, n.d.). These data sets were originally used in (Horne & Garton, 2006b) and were obtained for this current study from the supplementary material to that chapter. The precise data collection processes are not known for the data sets.

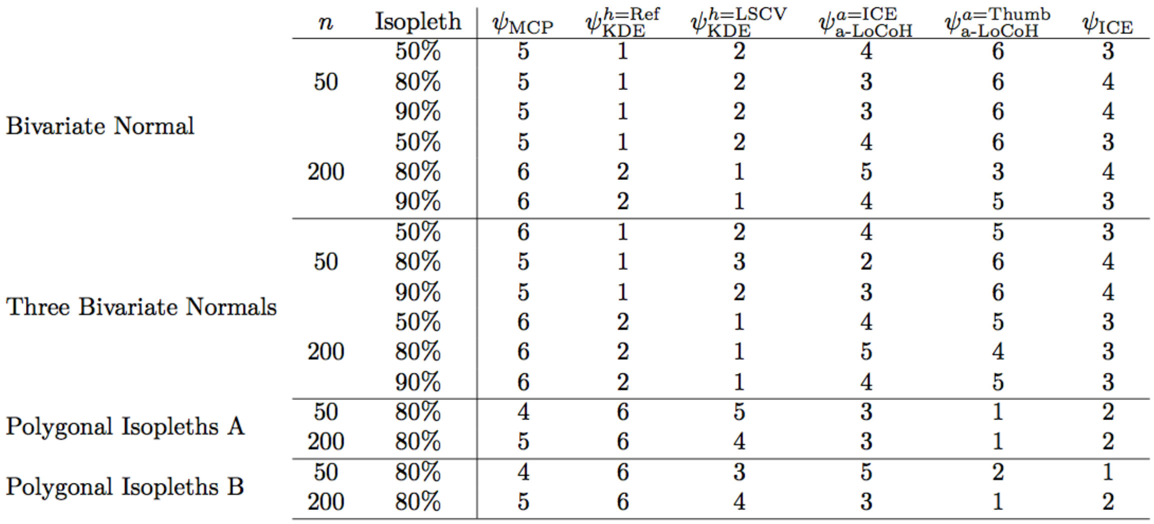
Results

Results are available both for the simulation study and for the analyses run on the empirical data sets.

Simulation Study Results

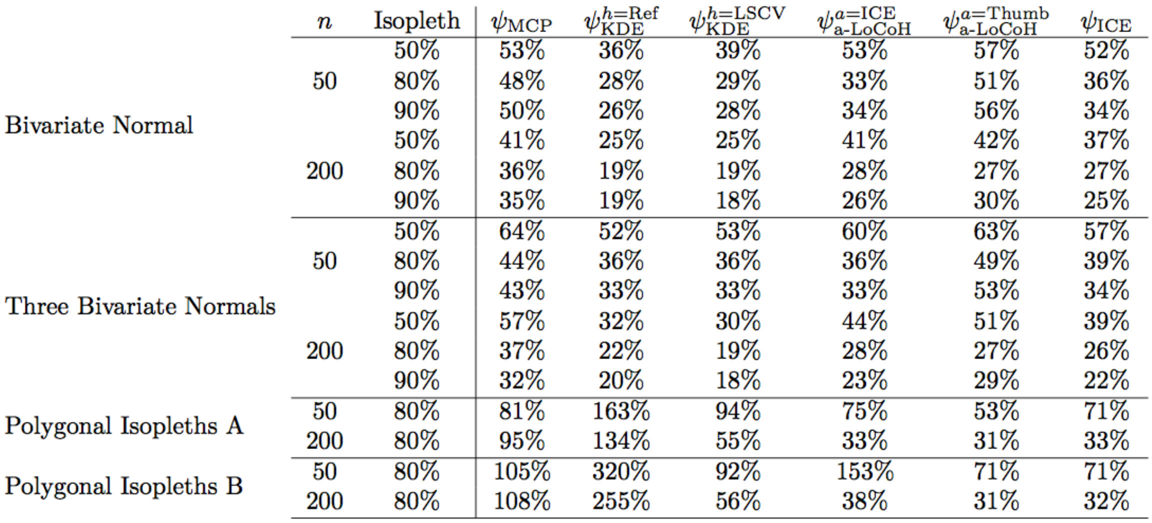
Overall quantitative quality of the different home range estimators is reported in Table 1, which ranks the estimators from best to worse for each combination of estimator, sample size, isopleth level, and data generating process. Table 2 shows the raw percentage errors for each combination of estimator and data generating process. Figures 7-10 illustrate the accuracy of the individual estimators using box plots to summarize the distributions of errors for the 40 different stochastic runs of each data generating process. In general, if the notches for two boxes do not overlap, this is "strong evidence" of a difference in the average accuracies between the two estimators (Chambers, Cleveland, Kleiner, & Tukey, 1983).

Table 11. Overview of methods rankings results from the simulation study (1 indicates the best estimator for a given set of conditions while 6 indicates the worst estimator).



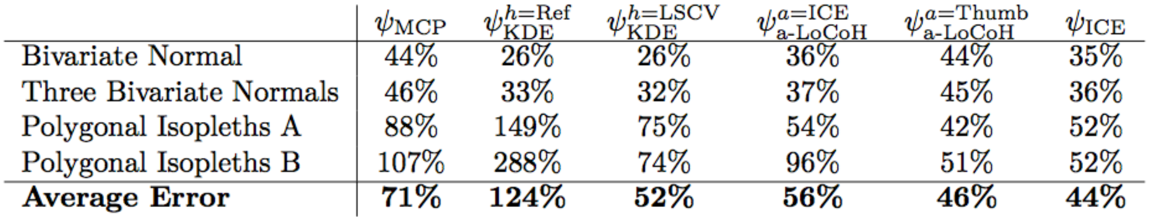
	n	Isopleth	ψ_{MCP}	$\psi_{\text{KDE}}^{h=\text{Ref}}$	$\psi_{\text{KDE}}^{h=\text{LSCV}}$	$\psi_{\text{a-LoCoH}}^{a=\text{ICE}}$	$\psi_{\text{a-LoCoH}}^{a=\text{Thumb}}$	ψ_{ICE}
Bivariate Normal	50	50%	5	1	2	4	6	3
		80%	5	1	2	3	6	4
		90%	5	1	2	3	6	4
	200	50%	5	1	2	4	6	3
		80%	6	2	1	5	3	4
		90%	6	2	1	4	5	3
Three Bivariate Normals	50	50%	6	1	2	4	5	3
		80%	5	1	3	2	6	4
		90%	5	1	2	3	6	4
	200	50%	6	2	1	4	5	3
		80%	6	2	1	5	4	3
		90%	6	2	1	4	5	3
Polygonal Isopleths A	50	80%	4	6	5	3	1	2
	200	80%	5	6	4	3	1	2
Polygonal Isopleths B	50	80%	4	6	3	5	2	1
	200	80%	5	6	4	3	1	2

Table 12. Overview of percentage error results from the simulation study.



	n	Isopleth	ψ_{MCP}	$\psi^{h=\text{Ref}}_{\text{KDE}}$	$\psi^{h=\text{LSCV}}_{\text{KDE}}$	$\psi^{a=\text{ICE}}_{\text{a-LoCoH}}$	$\psi^{a=\text{Thumb}}_{\text{a-LoCoH}}$	ψ_{ICE}
Bivariate Normal	50	50%	53%	36%	39%	53%	57%	52%
		80%	48%	28%	29%	33%	51%	36%
		90%	50%	26%	28%	34%	56%	34%
	200	50%	41%	25%	25%	41%	42%	37%
		80%	36%	19%	19%	28%	27%	27%
		90%	35%	19%	18%	26%	30%	25%
Three Bivariate Normals	50	50%	64%	52%	53%	60%	63%	57%
		80%	44%	36%	36%	36%	49%	39%
		90%	43%	33%	33%	33%	53%	34%
	200	50%	57%	32%	30%	44%	51%	39%
		80%	37%	22%	19%	28%	27%	26%
		90%	32%	20%	18%	23%	29%	22%
Polygonal Isopleths A	50	80%	81%	163%	94%	75%	53%	71%
	200	80%	95%	134%	55%	33%	31%	33%
Polygonal Isopleths B	50	80%	105%	320%	92%	153%	71%	71%
	200	80%	108%	255%	56%	38%	31%	32%

Table 13. Aggregate of percentage error results from the simulation study.



	ψ_{MCP}	$\psi^{h=\text{Ref}}_{\text{KDE}}$	$\psi^{h=\text{LSCV}}_{\text{KDE}}$	$\psi^{a=\text{ICE}}_{\text{a-LoCoH}}$	$\psi^{a=\text{Thumb}}_{\text{a-LoCoH}}$	ψ_{ICE}
Bivariate Normal	44%	26%	26%	36%	44%	35%
Three Bivariate Normals	46%	33%	32%	37%	45%	36%
Polygonal Isopleths A	88%	149%	75%	54%	42%	52%
Polygonal Isopleths B	107%	288%	74%	96%	51%	52%
Average Error	**71%**	**124%**	**52%**	**56%**	**46%**	**44%**

These multifaceted results are understandably difficult to summarize succinctly, as the estimators' performances vary as a function of data generating process, sample size and isopleth level. In order to obtain a high-level understanding of the accuracy of the estimators, the accuracies for each estimator were averaged across sample size and isopleth levels for a given data generating process and the results summarized in Table 3. Given this averaging process which may or may not reflect the types of empirical data a user will see in practice, statistical significances are not estimated and these results are presented in order to give a general indication of the quality of the estimators. Table 3 also averages these results across the four different data generating processes explored here to obtain a final, aggregate measure of the different estimators' accuracies. Although the ICE estimator is not the most accurate for any single of the four different data generating process, ICE is demonstrated to be the most accurate of the estimators in aggregate when the errors for the four processes are averaged.

Of note is that while the “ad-hoc” KDE with reference bandwidth and MCP estimators are shown to perform quite poorly compared to the other estimators (as would be expected), interestingly the “rule of thumb” *a*-LoCoH estimator performs well in this study and performs better than the pure *a*-LoCoH ICE estimator. Why this occurs is not obvious; clearly due to its sensitivity to outliers, data generating processes could be defined on which it would perform poorly. However, for the data generating processes explored here, it turns out to do quite well.

Effect of Data Generating Process

According to the estimator performance rankings shown in Table 1, performance varied by data generating process. For the bivariate normal data generating processes, members of the KDE family of candidate estimators were consistently able to achieve the lowest errors on average. Conversely, for the polygonal isopleth data generating processes, members of the a-LoCoH family of estimators were able to achieve the lowest error on average. The MCP estimator performed uniformly poorly, although it performed relatively better on the polygonal data generating processes compared to the bivariate normal data generating processes.

These results support the key hypothesis made in the introduction of this chapter that different types of home range estimators perform relatively better or worse on different kinds of data. Specifically, the results support the proposal that KDE estimators would perform better on home ranges governed by smooth gradually changing densities (as typified by the bivariate normal data generating processes), while LoCoH estimators would perform better on home ranges governed by densities contain sharp edges and discontinuities as typified by the polygonal isopleth data generating processes.

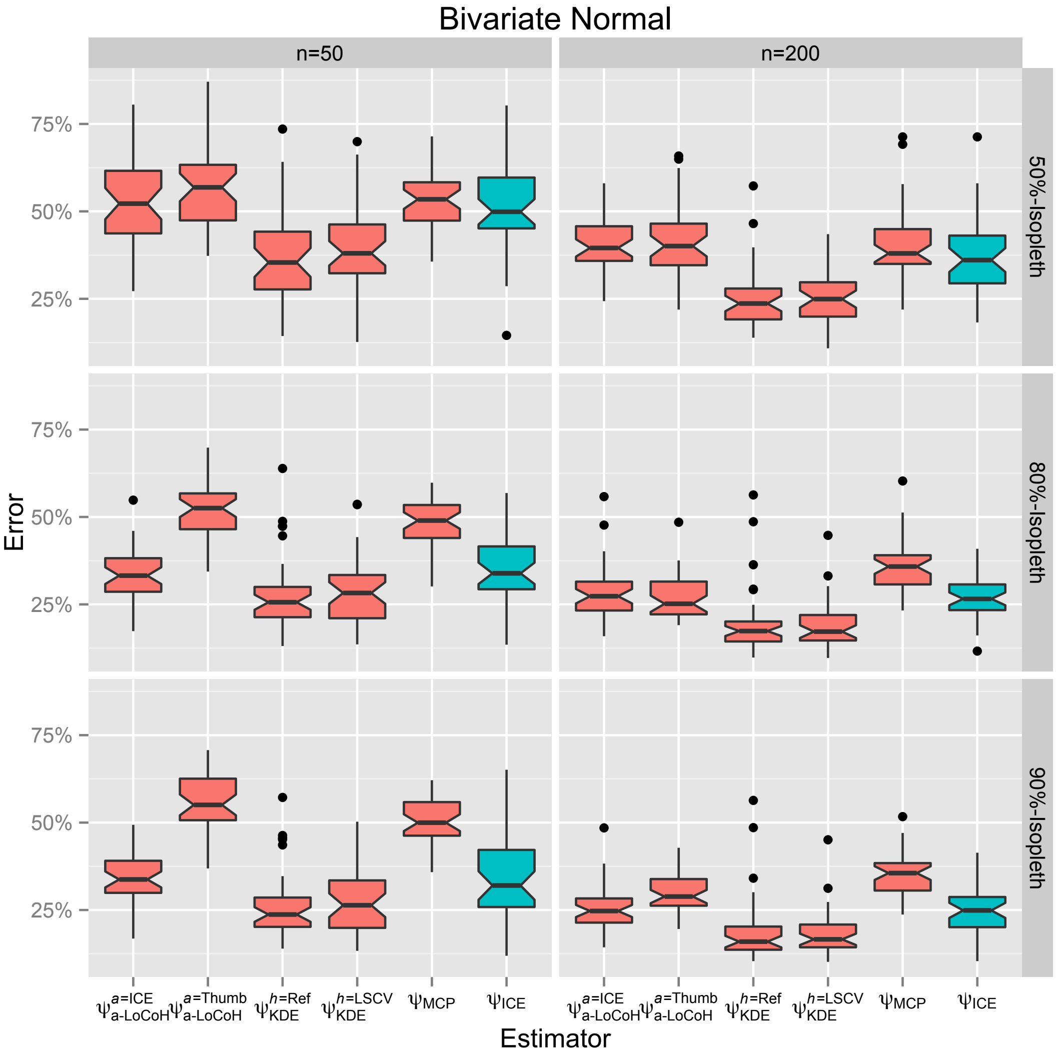


Figure 13. Errors for estimators applied to simulated data from bivariate normal densities.

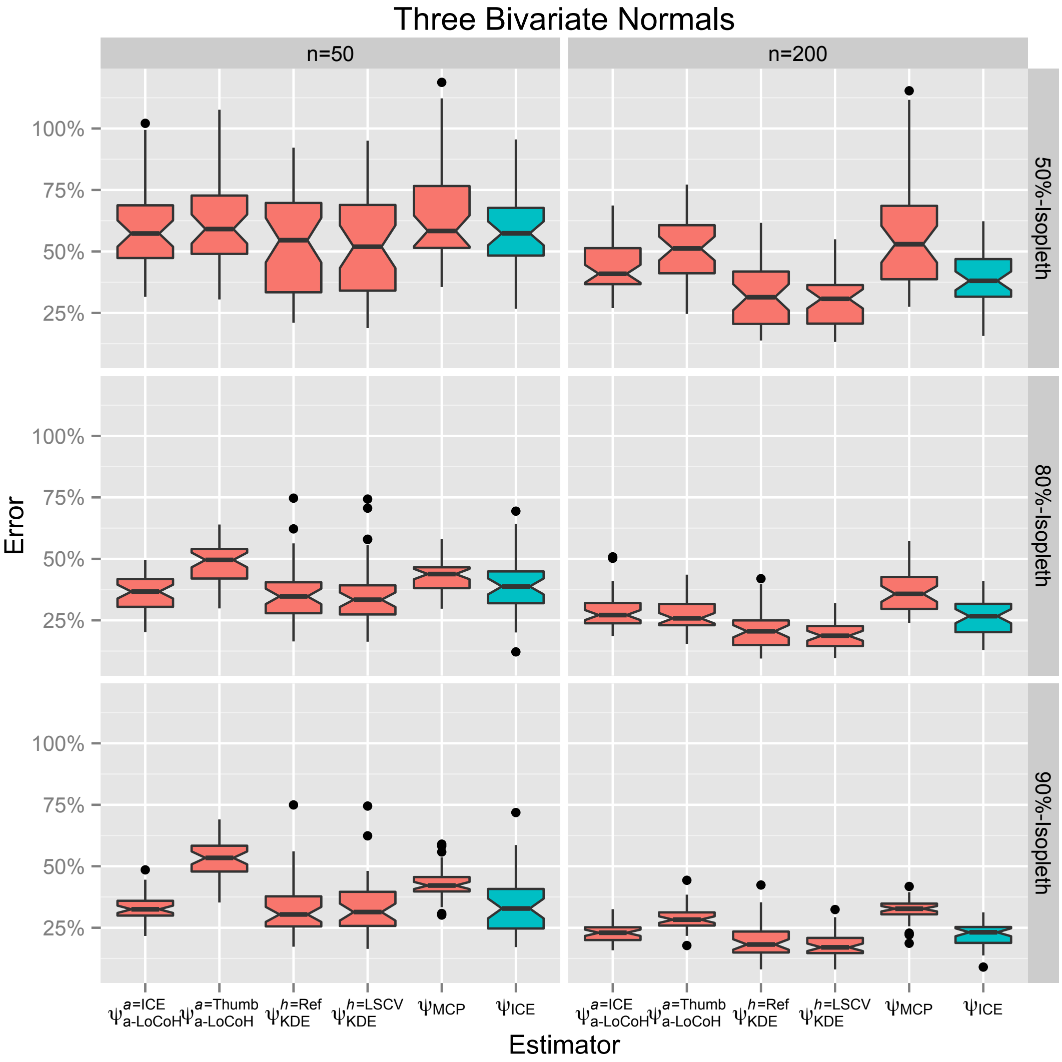


Figure 14. Errors for estimators applied to simulated data from three bivariate normal densities.

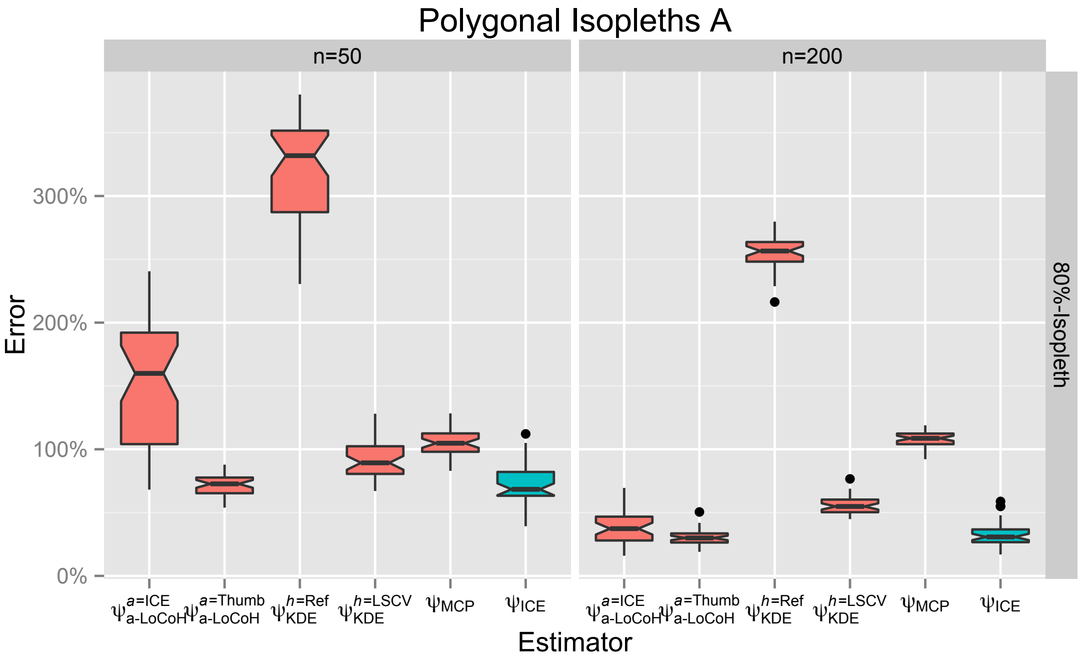


Figure 15. Errors for estimators applied to simulated data from manually specified polygonal isopleths, pattern A.

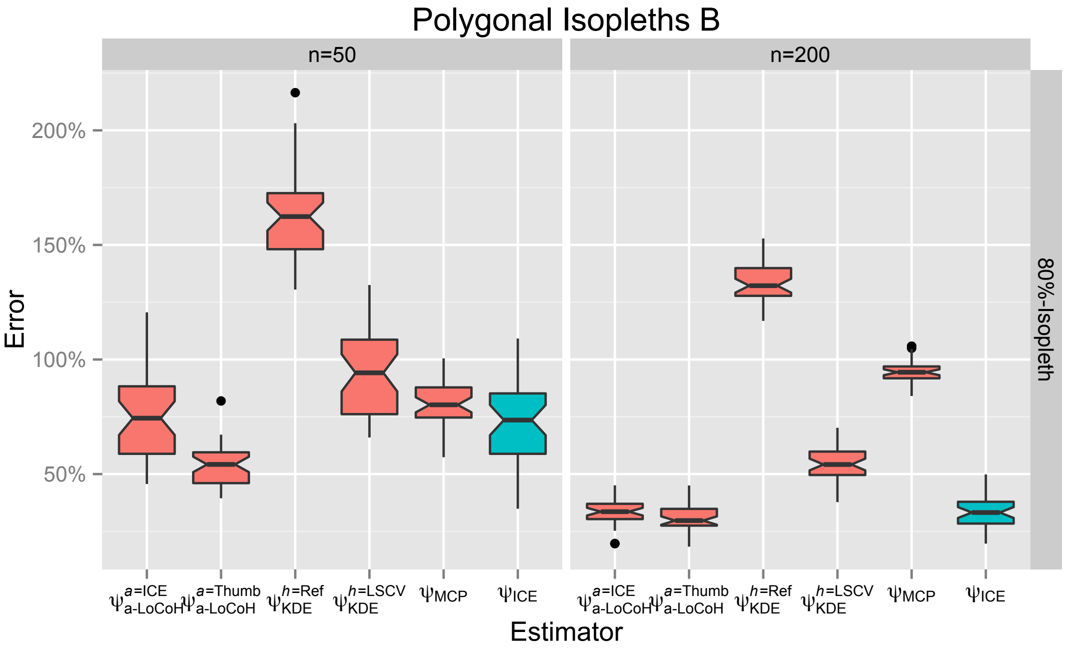


Figure 16. Errors for estimators applied to simulated data from manually specified polygonal isopleths, pattern B.

Effect of Sample Size and Isopleth Level

Increased sample size consistently leads to increased accuracy of the estimators, the exception being the MCP estimator where this did not hold true for the polygonal isopleths data generating processes. Additionally, for the polygonal isopleths data generating processes, the variability of the estimators' accuracies also typically drops significantly as the sample size increases – a phenomenon not seen to such an extent in the bivariate normal data generating processes.

The majority of the simulations were carried out with two sample sizes: 50 observations and 200 observations. Figure 17, extends this analysis for one data generating process to 50 observations, 200 observations, and 400 observations. This figure shows, with the exception of the MCP estimator, the estimators' performances improving as the sample size rises. The rate of the improvement falls off as the sample size rises, indicating decreasing marginal value with additional data.

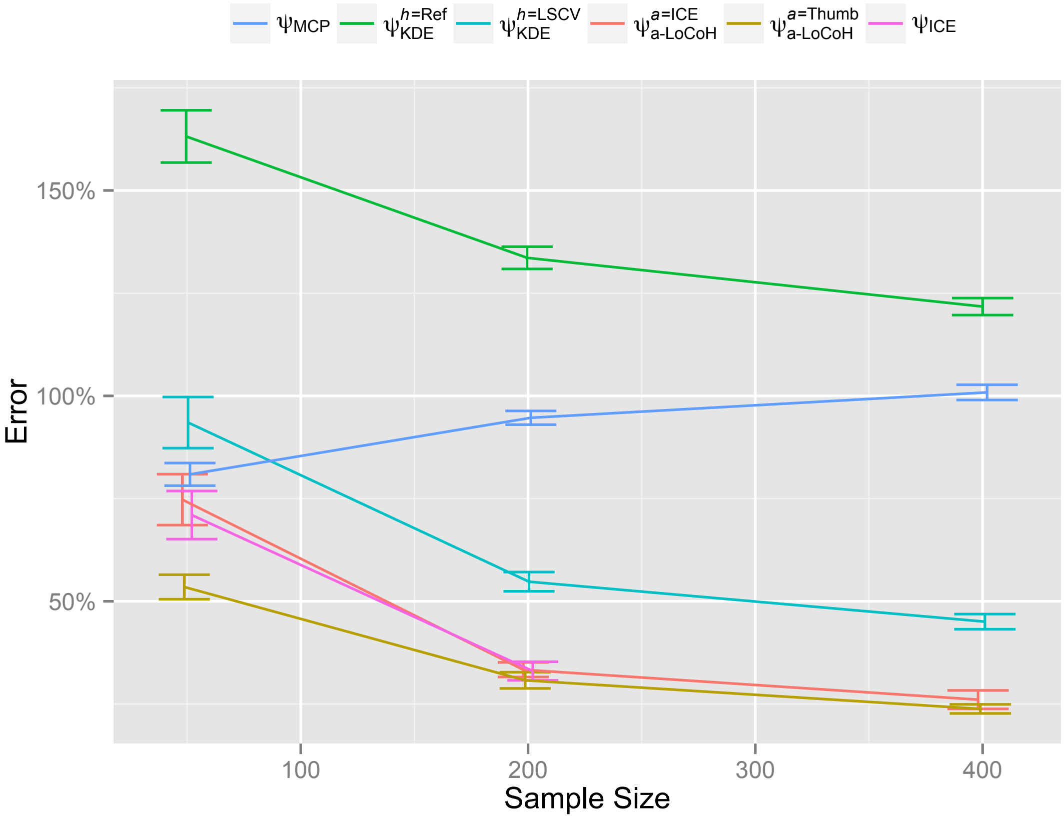


Figure 17. Effect of sample size (*n*=50,200,400) on error for the polygonal isopleths B data generating process. 95% confidence intervals are also plotted.

In regards to the impact of isopleth level, the results in Figures 7-10 indicate that for the bivariate normal data generating processes there is a general reduction in magnitude and variability of the errors as α is increased from 0.5 to 0.9.

Empirical Data Results

ICE was applied to the five data sets described earlier in this work. 90%-, 70%- and 50%-isopleths were calculated for these data sets. The resulting home ranges are displayed in Figure 18. Although the selected candidate estimators are not labeled, they can be clearly identified from the shape of the displayed home ranges[17]. Thus, by way of example, it can be seen that in the case of the Black Bear data set, one of the a-LoCoH estimators is selected for the 50% and 70%-isopleths while a KDE estimator is selected for the 90%-isopleths. Conversely, for the ELK data set, MCP is selected for all three isopleths. The results here show that not only do the best estimators, as identified by ICE, vary between animals and studies, the best estimators may also vary for a single data set between isopleths.

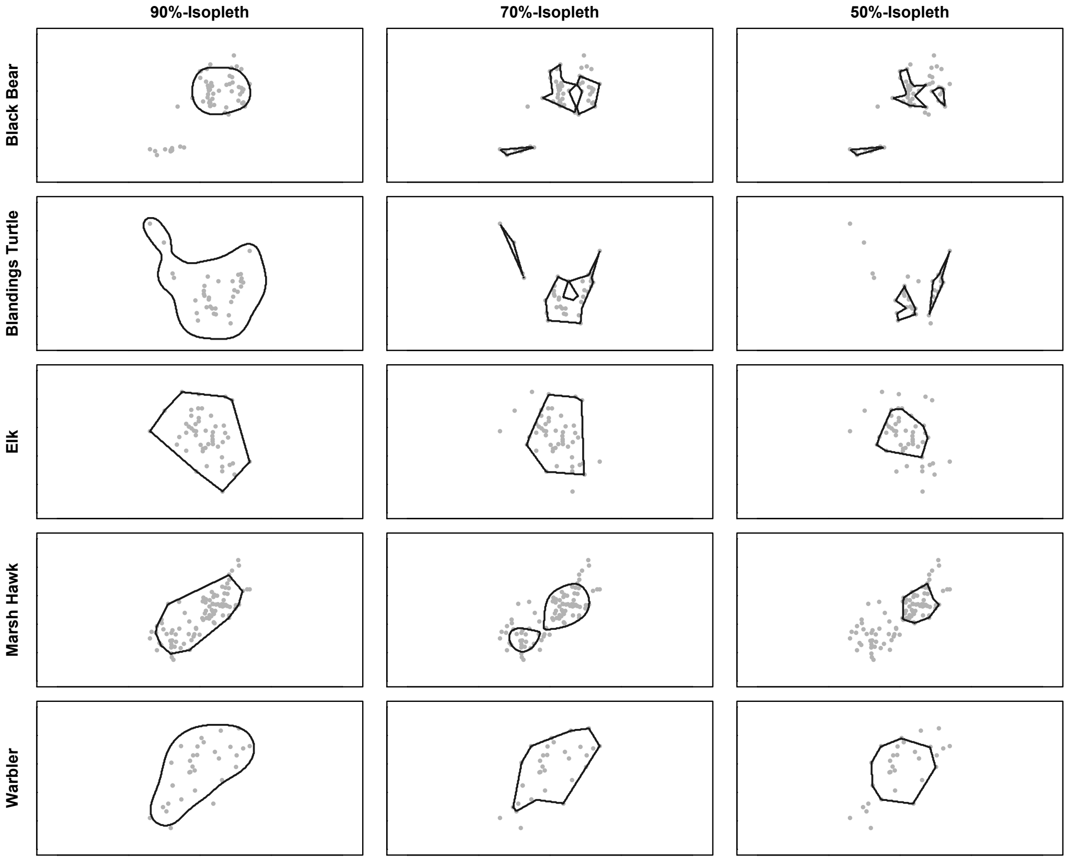


Figure 18. ICE applied to five empirical data sets at three isopleth levels.

In general the choice of candidate estimator by ICE for these empirical data sets appears to be related to the relative level of detail in the isopleth being estimated. For original isopleths that are largely smooth (such as those generally at the 90% level), KDE or MCP are selected. For original isopleths containing significant variability (such as those with lower α's for some species), MCP is not selected (as

[17] KDE is the only estimator that generates smooth curves. a-LoCoH regions will contain concavities while MCP home range will not.

this estimator is simply not able to capture significant variability) and a-LoCoH is often selected. This result illustrates the ICE approach adapting to the properties of the data itself as it selects between competing candidate estimators.

Discussion and Conclusions

Isopleth Curve Estimation (ICE) illustrates an approach to home range estimation where multiple candidate estimators propose home ranges. In this competitive process, the proposals are evaluated and the competing proposal that best fits the data is selected. The simulation study carried out in this chapter demonstrates the need for a hybrid estimator of estimators by illustrating the better suitability of KDE for certain types of location data, while at the same time showing the better suitability of *a*-LoCoH for alternative forms of data. By selecting between multiple candidate estimators ICE is able to select an estimator best tailored for a given data set and this chapter's simulation study has shown that an ICE estimator composed of three families of home range estimators (MCP, KDE and LoCoH) performed better than any of its individual constituents.

As a predictive home range estimator that selects between individual candidate estimators, the accuracy of the ICE estimator depends therefore to a significant extent on the quality of the estimators fed into it. A diverse set of high quality candidate estimators should lead to a high quality ICE estimator that improves upon the individual candidates. When lacking a high quality set of diverse candidate estimators, the ICE estimator may suffer. As an example of this limitation, the ICE estimator composed solely of *a*-LoCoH candidates used in this chapter performed relatively poorly. Most likely the relative weakness of ICE in this limited case arose because the usage solely of *a*-LoCoH estimates resulted in insufficient data points to properly estimate the isopleth curve. This illustrates the importance of obtaining a rich sample of diverse and high quality home range estimates. The ICE estimator combining *a*-LoCoH, KDE, and MCP estimates appears to obtain a sufficient level of diversity as it outperformed the individual candidate estimators. The inclusion of additional candidate estimators, if the added computational cost can be afforded, should only serve to ICE estimator further.

It is necessary to highlight issues that may be encountered when applying ICE in practice. First, a key statistical assumption ICE is that of independence between observations. This assumption is a cornerstone of ICE's cross-validation procedure used to estimate *p*. This assumption is also one that most other statistical density estimation methodologies – such as LSCV KDE – make. In practice, sequentially recorded observations of an animal's location, such as those collected using a GPS collar, can violate this assumption. Any violation biases results (as such violations would, for instance, bias the results of a home range generated with LSCV KDE) and the level of that bias depends on the extent of the violation. In general, the longer animals are observed and the higher the ratio of total observation period to the sampling frequency is, the less important the bias will be.

Additionally, in the case where the correlation structure between observation processes is known, it may be possible to adjust the estimation of *p* in ICE to directly account for the correlations. Another issue we should highlight is the computational cost of the ICE algorithm. The algorithm employs LOOCV in the estimation of *p*. Thus, for each candidate estimator it generates n home ranges. For small data sets, this should not be a computational issue but as the size of the data set increases the computational effort will generally grow exponentially. If there are thousands of observations in the data set, for instance, this cross validation could result in a significant computational burden. To address this, *n*-fold cross validation may be used for larger data sets where some smaller number of folds may be used such as 10 or 20. This will increase the bias in the estimate of *p*, but for large data sets this bias may be insignificant.

The ICE estimator fills a hole in existing home range estimation tools. It provides an estimator of estimators that can select between isopleths proposed by other estimators to attempt to find one that best fits the observed data. By selecting between competing candidate estimators it can provide greater accuracy than any individual candidate. There are many intriguing extensions to the ICE estimator. For instance, an extension of ICE towards the estimations of full densities instead of simply individual isopleth is an interesting prospect. Additionally, ICE as currently proposed selects between different candidate estimators. An alternative would be to "average" competing candidates using a mechanism that gave more weight to candidates nearer the intersection of the isopleth curve. Such an idea is similar in concept to techniques such as the non-discrete Super Learner machine learning algorithm (Van Der Laan & Rose, 2011) and could potentially decrease the variance of ICE thereby increasing its overall accuracy. However, such an extension would require the development of an approach to sensibly averaging two-dimensional regions, a potentially difficult challenge.

References

Anderson, D. J. (1982). The home range: a new nonparametric estimation technique. *Ecology*, 103–112.

Blundell, G., Maier, J., & Debevec, E. (2001). Linear home ranges: effects of smoothing, sample size, and autocorrelation on kernel estimates. *Ecological Monographs*, *71*(3), 469–489.

Burgman, M., & Fox, J. (2003). Bias in species range estimates from minimum convex polygons: implications for conservation and options for improved planning. *Animal Conservation*, *6*(01), 19–28.

Burt, W. (1943). Territoriality and home range concepts as applied to mammals. *Journal of Mammalogy*, *24*(3), 346–352.

Calenge, C. (2006). The package “adehabitat” for the R software: A tool for the analysis of space and habitat use by animals. *Ecological Modelling*, *197*(3-4), 516–519. doi:10.1016/j.ecolmodel.2006.03.017

Cassirer, E. F., Freddy, D. J., & Ables, E. D. (1992). Elk responses to disturbance by cross-country skiers in Yellowstone National Park. *Wildlife Society Bulletin*, *20*(4), 375–381.

Chambers, J., Cleveland, W., Kleiner, B., & Tukey, P. (1983). *Graphical Methods for Data Analysis*. Pacific Grove, CA USA: Wadsworth and Brooks/Cole.

Darwin, C. (1861). *On the origin of the species by means of natural selection*. London: Murray.

Downs, J. A., & Horner, M. W. (2009). A Characteristic-Hull Based Method for Home Range Estimation. *Transactions in GIS*, *13*(5-6), 527–537. doi:10.1111/j.1467-9671.2009.01177.x

Dudoit, S., & van der Laan, M. J. (2005). Asymptotics of cross-validated risk estimation in estimator selection and performance assessment. *Statistical Methodology*, *2*(2), 131–154. doi:10.1016/j.stamet.2005.02.003

Eddelbuettel, D., & Romain, F. (2011). Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, *40*(8), 1–18.

Fieberg, J. (2007). Kernel density estimators of home range: smoothing and the autocorrelation red herring. *Ecology*, *88*(4), 1059–1066.

Garton, Edward. (n.d.). Circus cyaneus. Moscow, Idahow: Unpublished.

Getz, W., & Wilmers, C. (2004). A local nearest-neighbor convex-hull construction of home ranges and utilization distributions. *Ecography*, *27*(4), 489–505.

Getz, W., Fortmann-Roe, S., Cross, P., Lyons, A., Ryan, S., & Wilmers, C. (2007). LoCoH: nonparameteric kernel methods for constructing home ranges and utilization distributions. *PLoS ONE*, *2*(2), 207.

Gitzen, R. A., & Millspaugh, J. J. (2003). Comparison of least-squares cross-validation bandwidth options for kernel home-range estimation. *Wildlife Society Bulletin*, 823–831.

Horne, J., & Garton, E. (2006a). Likelihood cross-validation versus least squares cross-validation for choosing the smoothing parameter in kernel home-range analysis. *Journal of Wildlife Management*, *70*(3), 641–648.

Horne, J., & Garton, E. (2006b). Selecting the best home range model: an information-theoretic approach. *Ecology*, *87*(5), 1146–1152.

Horne, J., Garton, E., Krone, S., & Lewis, J. (2007). Analyzing animal movements using Brownian bridges. *Ecology*, *88*(9), 2354–2363.

Jette, L., Hayden, T., & Cornelius, J. (1998). *Demographics of the Golden-cheeked Warbler (Dendroica chrysoparia) on Fort Hood, Texas.*

Johnson, G. (n.d.). Emydoidea blandingii. Potsdam, NY: Unpublished.

Kenward, R., Clarke, R., Hodder, K., & Walls, S. (2001). Density and linkage estimators of home range: nearest-neighbor clustering defines multinuclear cores. *Ecology*, *82*(7), 1905–1920.

Laver, P. N., & Kelly, M. J. (2008). A Critical Review of Home Range Studies. *Journal of Wildlife Management*, *72*(1), 290–298. doi:10.2193/2005-589

R Core Team. (2012). *\proglang{R}: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://www.R-project.org/

Sager, K. (n.d.). *Black bear distribution patterns in a temperate forest environment, Olymptic National Part*. University of Idaho, Moscow, Idaho.

Seaman, D., & Powell, R. (1996). An evaluation of the accuracy of kernel density estimators for home range analysis. *Ecology*, *77*(7), 2075–2085.

Seaman, D., Millspaugh, J., Kernohan, B., Brundige, G., Raedeke, K., & Gitzen, R. (1999). Effects of sample size on kernel home range estimates. *The Journal of Wildlife Management*, *63*(2), 739–747.

Silverman, B. (1986). *Density estimation for statistics and data analysis*. *Monographs on Statistics and Applied Probability. …*. London: Chapman & Hall.

Van Der Laan, M., & Rose, S. (2011). *Targeted Learning: Causal Inference for Observational and Experimental Data*. New York: Springer.

Worton, B. (1989). Kernel methods for estimating the utilization distribution in home-range studies. *Ecology*, *70*(1), 164–168.

Worton, B. (1995). Using Monte Carlo simulation to evaluate kernel-based home range estimators. *The Journal of Wildlife Management*, *59*(4), 794–800.

4. Contingent Kernel Density Estimation[18]

Introduction

In the previous chapter, we explored a home range estimation technique that placed candidate estimators in direct competition with each other. One of the candidate estimators utilized in that chapter was Kernel density estimation. In this chapter, a novel extension to the Kernel density estimation technique is developed that provides improved density estimation in cases when the data is observed with a specific error structure. Given data collected with this type of error, we can obtain improved models and parameter estimates using the technique developed here.

Kernel density estimation is a popular method for using a sample of points to estimate the distribution that generated those points. During application, a probability function known as a kernel is applied to each sampled point and the kernels are summed to obtain an estimate of the original distribution. This estimation technique is employed in diverse fields such as signal processing (Chang and Ansari), econometrics (Powell, Stock, and Stoker) and ecology (Millspaugh et al.).

A subcategory of distribution estimation problems occurs when points are observed with some error. When non-negligible, the errors can lead to biases in the kernel density estimate. Examples of errors include observation bias where points are more likely to be observed in certain regions of the sampling space (independent of the original distribution) or measurement error where, for instance, the observed location of a point has been randomly displaced with known noise statistics from its true location. Methods have been developed for dealing with both these types of errors (Stefanski and Carroll; Horne, Garton, and Sager-Fradkin).

One type of error that has not been considered is the case where points within a defined area are nominally placed at a designated location within that area. This is quite relevant to home range estimation as an example of this issue are these sampling regimes include ecological applications in which observations of species are mapped to predefined geographic entities, such as the center of the region in which the observation took place or to the center of distinct squares of a grid overlaid on that region. Another example is the analysis of types of location data generated by new social websites such as Twitter or Facebook.

To address this form of error, a kernel density estimation method is presented here in which the shape of the kernel is contingent on the location in space of the point given a known location-specific error distribution. In this new "contingent

[18] This chapter was previously published as Fortmann-Roe, S., Starfield, R., & Getz, W. M. (2012). Contingent Kernel Density Estimation. PLoS ONE.

kernel density estimation" method, the contingent kernel is determined by the convolution a kernel function and an error distribution function. If the sampling errors vary between observations, the form of the contingent kernel will change on an observation-by-observation basis.

The technique presented here is a rigorous statistical technique, but it is also, to large extent, a narrative one. The reason for this classification is that the method is primarily assumption driven. As we will see, assumptions are made about the form of the errors and then the analysis progresses based on these axioms. If these assumptions are correct, the results of the model should be accurate but no assessment of this is made. Furthermore, in some cases asymptotic statements may be made about the predictive accuracy of the model, but no measurements during modeling are made of the predictive error for the given finite sample of data being analyzed. Given this, the model presented here is a narrative model rather than being a predictive model.

In this chapter, we first derive contingent kernels for several common kernel and sampling regime combinations. We then validate the proposed estimation technique using simulated data and finally demonstrate the utility of the method using location observations gathered from the social-networking site Twitter. Our results indicate that the contingent kernel density estimation can lead to more accurate density estimates where characterizable error varies across space.

Symbols used in the Chapter

$f(x)$	Probability density function whose distribution will be estimated
h	Scaling or bandwidth parameter $h > 0$ (increased values of h lead to increased smoothing in the estimate)
$K(x)$	A family of kernel functions (e.g., Gaussian kernel or Epanechnikov kernel)
$f(x\|h)$	Kernel density estimate of $f(x)$ which is dependent on the bandwidth (h)
$\Psi(x\|U)$	Error distribution; often it is defined by one or more covariates denoted by the variable U
$C(x\|h,U)$	Contingent kernel function; it is dependent on both the parameters of the error distribution (U) and the selection of a standard kernel function
$\mathbf{X}$	A set of n points $\{X_1,\dots,X_n\}$ sampled or drawn from $f(x)$

Background

Assuming a sample of *n* points $\mathbf{X}=\{X_1,\ldots,X_n\}$ drawn from an unknown, univariate probability density function f. The kernel density estimate $\hat{f}$ of f is

$$\hat{f}(x|h) \equiv \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x-X_i}{h}\right), \qquad (1)$$

where K is a user-defined kernel function and h is a user-defined parameter that controls smoothing (often called the bandwidth or the window width). The kernel function may theoretically be of any form such that integration across its domain equals one. In practice, though, kernel functions are typically symmetrical and unimodal around the origin. A commonly used kernel function that satisfies these constraints is the Gaussian kernel, a kernel that is defined as a normal distribution with a mean of zero and a standard deviation of one.

Kernel density estimation is a challenging problem, not the least because results are highly sensitive, as illustrated in Figure 19, to the method used to determine the 'best' value for the bandwidth parameter h (Silverman; Worton). Beyond this, problems occur when data points are missing, but methods for compensating for such error have been developed (Horne, Garton, and Sager-Fradkin; Jones).

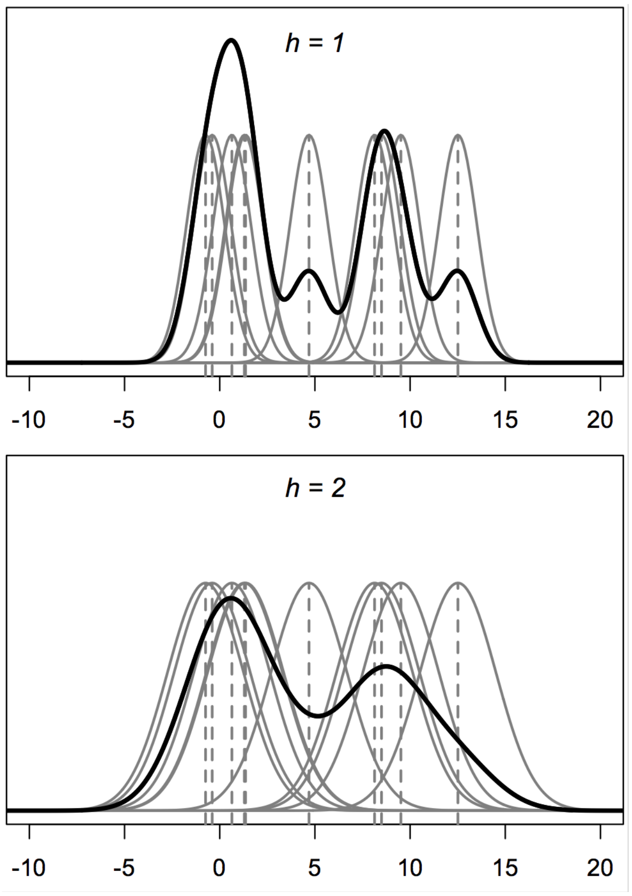


Figure 19. Kernel density estimation illustration. The dark line represents the kernel density estimate while the grey lines are scaled Gaussian kernels for each of ten individual points sampled from the original probability density function. The original probability density function is a mixture function defined as an equal combination of two Gaussian distributions: one with a mean of 0 and standard deviation of 1, the other with a mean of 10 and a standard deviation of 4. The top panel uses a bandwidth of 1 while the bottom panel uses a bandwidth of 2.

Another form of error that has been addressed is additive measurement error. A number of studies have looked at the case in which the observations can be modeled as the additive effects of the original probability density function and some error probability density function (e.g., (Stefanski and Carroll), (Fan) and (Fan)). In this case, the observed distribution of points is the convolution of the original probability density function and the error probability density function. Fourier transformations can be used to develop deconvolution kernels that compensate for the error. The deconvolution kernel density estimator D is defined below in Equation 2 where φ_K and η_Z are the Fourier transforms of the kernel K and the error distribution Z respectively.

$$D(x|h) \equiv \frac{1}{2\pi}\int e^{-itx}\,\frac{\eta_K(t)}{\eta_Z(t/h)}\,dt \qquad (2)$$

Although conceptually straightforward, the application of deconvolution kernels can be difficult in practice. For instance, Equation 2 is often intractable to calculate. As a result, kernels with compactly supported Fourier transforms are preferred thereby limiting kernel choices (Delaigle and Gijbels). Furthermore, such deconvolution methods cannot be applied to data gathered in certain common sampling schemes, as when observed points are relocated to the center of the nearest of a set of predefined geographic entities.

An example of such sampling schemes occurs when observations are assigned to the center of a Transect-Range-Section grid that has historically been used in the description of locations in species surveys (Figure 20). If a deconvolution kernel were applied to this case, it could result in artificial reductions in density at the centers of the grids thereby introducing a new form of bias into the estimate.

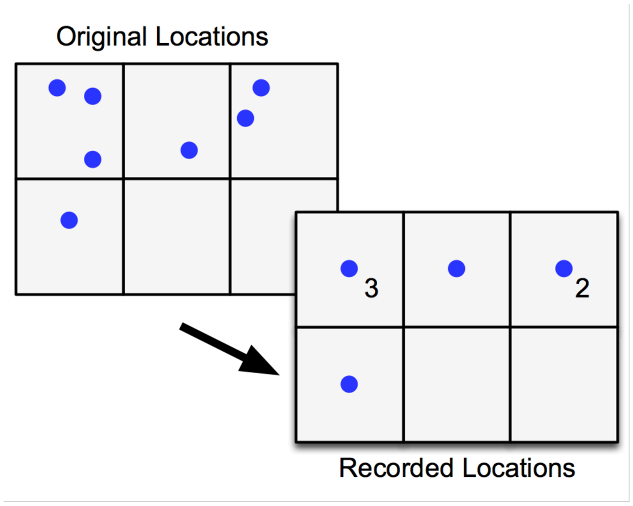


Figure 20. Contamination of observations. An illustration of Transect-Range-Section sampling where the locations of observations within a grid are assigned to the center of enclosing grid squares. Such cases are not suitable for deconvolution techniques to adjust for bias.

In the case of the Transect-Range-Section grid, the geographic entities are all the same size. However, in other sampling regimes this is not the case. For instance, given a statewide survey where observations are assigned to the centers of the nearest county, the size and shape of each county differ. In other cases, the entities themselves may overlap.

Another factor that can lead to differences in the forms of errors is the combination of datasets from multiple surveys. This is especially true given longitudinal surveys whose duration intersects with the development of new technologies and methods. As an example, take an ecological study that surveys the locations of an animal species. If the species is surveyed many times over a period spanning decades, the first surveys could have been carried out using visual identifications, the later surveys carried out using radio collar triangulation, and the most recent surveys completed with GPS sensors. Each technique has its own types of errors. A similar problem arises when combining data from remote sensing satellite systems with different levels of resolution (e.g. MODIS has spectral bands with resolutions ranging from 250-1000 m while Quickbird has a much finer multispectral resolution of 2.4 m).

Unfortunately, many historical ecological datasets are lost or misplaced once their primary investigators move on to other interests. A number of repositories archive these historical studies. The Museum of Vertebrate Zoology at the University of California Berkeley is one such facility containing field notebooks, dating back over one hundred years, that document flora and fauna distributions (The Museum of Vertebrate Zoology at Berkeley). The ability to integrate such historical observations with more modern ones under a unified framework is needed. Such a technique would aid in modeling the effects of a changing environment.

To address these issues – to compensate for this form of error and allow the integration of diverse data sources in a unified analyses framework – this chapter proposes the contingent kernel density estimation methodology. As defined herein, contingent kernel density estimation is the same as standard kernel density estimation except that the kernel is treated as a contingent kernel that changes form on a point-by-point basis in response to error at that point.

The contingent kernel C is defined as the scaled convolution of a standard kernel function and the probability density function that describes the potential error distribution Ψ, which is in turn specified by a set of parameters U. The resulting univariate formula is given below in Equation 3 for the case where Ψ is constant from point to point although U may vary. Note that the denominator term is a normalizing factor to ensure that the area under the contingent kernel will be 1. The precise shape of the contingent kernel can change on a point-by-point basis in response to changes in the magnitude of the error function through variations in the parameters U.

$$C(x|h,U) \equiv \frac{\int K\left(\frac{x-\psi}{h}\right)\Psi(\psi|U)\,d\psi}{\iint K\left(\frac{x-\psi}{h}\right)\Psi(\psi|U)\,d\psi\,dx} \qquad (3)$$

Whereas any standard kernel function can be converted into a contingent kernel through this calculation, the error distribution is defined by the sampling regime. For instance, in the case of a Transect-Range-Section grid study, the error distribution would be a bivariate uniform distribution with the same dimensions as a single grid square. If the size of the grid squares changed, U would represent that changing size.

For example, assuming a constant error distribution Ψ while allowing for changes to its magnitude or shape through variation in U, such that U has specific values U_i associated with sampling point X_i, the contingent kernel density estimate is calculated in much the same way as the standard kernel density estimate (Equation 4). The primary change is that the contingent kernel is not based on a one parameter family of kernels $K(x)$, but takes multiple parameters instead of the single distance scaled by the bandwidth that is used in standard kernel density estimation.

$$\hat{f}(x|h) \equiv \frac{1}{n}\sum_{i=1}^{n} C(x - X_i|h,U_i) \qquad (4)$$

Methods

This chapter consists of three sections: the calculation of example contingent kernels, numerical validation of accuracy improvements, and an application to the location of Twitter users.

Calculation of the Demonstrative Contingent Kernels

Mathematical analyses were carried out both by hand and with the use of computer software. Specifically, the software program *Mathematica* version 8.0.1 was used as an aid to the symbolic analyses and the calculation of contingent kernels for common usage cases.

Numerical Validation of Accuracy Improvements

Computer simulations were carried out to numerically validate the improvement of the contingent kernel method as compared to standard kernel methods. The programming environment R version 2.13.0 was used to develop the analyses.

The validation process required four components:

1. The original probability density function f that the two methods attempted to estimate,
2. the sampling protocol which introduced errors into the observed locations of points,
3. the implementation of the standard and contingent kernel density estimation methods including the selection of a kernel and bandwidth, and
4. an error criterion to compare the accuracy of the original probability density function to the estimates $\hat{f}$.

With respect to point (1), an equal mixture function of three Gaussian kernels $N(\mu,\sigma)$ was used with the following means μ, and standard deviations σ: $N(-4,2)$, $N(3,1)$ and $N(0,0.75)$ (see Reference curve in Figure 23). Other studies assessing the accuracy of kernel density estimation have looked at mixture functions of Gaussians (Seaman et al.; Jones, Marron, and Sheather; Hall et al.) and experimentation with several other forms of functions did not change the results.

With respect to point (2), a filter was applied to the sampled observations in order to recreate the type of sampling behavior associated with Transect-Range-Grid sampling and other types of sampling with similar effects. A grid of equally spaced locations was overlaid on the range of the mixture function and the observations drawn from the mixture function were relocated to the nearest grid location. In order to simulate heterogeneous sampling behavior (e.g., if some data are assigned to the county level, while other data are assigned to the state level), three different resolutions of grids were used: the first with a spacing of 0.5, the second with a spacing of 1, and the third with a spacing of 2.

With respect to point (3), the kernel density estimation methods were implemented in R. To apply the methods, both the kernel form and the bandwidth value had to be selected. The Gaussian kernel, as it is commonly used, was selected for the analyses. The selection of the kernel bandwidth is a critical issue (see Figure 19) and since the task is a somewhat subjective choice it has received considerable attention in the literature (e.g. see (Sheather and Jones) and (Jones, Marron, and Sheather)). In our analysis, for consistency, we selected the bandwidths that minimized in each case the error between the original distribution f (which we know by construction of our artificial dataset) and its estimate $\hat{f}$, thereby resulting in different bandwidths for the different cases.

Finally, with respect to point (4), errors were assessed quantitatively and qualitatively. Several methods for quantifying errors have been proposed (Marron and Tsybakov; Jones). The most common method is to take the squared deviation between the original and estimated distributions (Equation 5). To assess deviations, errors were calculated using the Integrated Squared Error (ISE) (Equation 5 below) between the original and estimated probability density functions. Sensitivity tests compared the error to the sampling size for sample sizes ranging from 15 to 600 points. Mean Integrated Squared Error (MISE) (Equation 5 below) and the 95% confidence interval for the estimate were calculated for each sample size with 20 replicates. Plots comparing the resulting estimates are also shown to provide a more qualitative understanding of the form of these errors.

$$\begin{aligned} ISE(f,\hat{f}|h) &= \int \left[f(x) - \hat{f}(x|h) \right]^2 dx \\ MISE(f,\hat{f}|h) &= E\int \left[f(x) - \hat{f}(x|h) \right]^2 dx \end{aligned} \tag{5}$$

Application: Twitter User Locations

To demonstrate the utility of our contingent kernel density estimation procedure, we applied it to location data gathered from the social networking site, Twitter. Location data were collected for 10,000 Twitter users using Twitter's publically accessible *Streaming API*. Only data that users chose to be publically accessible to third party companies and researchers were collected and identifying information was deleted prior to analysis.

Twitter provides two forms of location information from those users who choose to make it publically accessible. The first is precise geolocation data obtained using a GPS device. The majority of Twitter users either do not have such devices or do not choose to make that data public. The more common form of location data is a free form, user-enterable text string describing a user's location.

This location string is definable by the users with little to no restrictions and thus there are high variations in the strings' precisions. For instance, take three hypothetical users living in San Francisco; one user might enter “San Francisco, California,” another “California,” and the third simply “USA.”

Regardless of users’ choice of precision, the first challenge in processing the data was to convert the strings to geolocation coordinates that could be mapped and compared. Such a task is non-trivial and either requires extensive manual labor or a large database of place names along with flexible text parsing software (e.g., the software needs to be able to determine that “SanFran, California” and “USA, San Francisco” refer to the same place).

To convert Twitter location strings to geolocated entities, *Yahoo PlaceFinder*, a free service provided by Yahoo, was used. Researchers can send *Yahoo PlaceFinder* free-form location strings, such as those strings entered by Twitter users. The service then processes the information and returns geolocation data for that string. The geolocation information is returned in the form of a disc defined by its center (as a latitude and longitude) and a radius (in meters). For example, if the user entered “San Francisco,” the center of the city and the approximate radius of the city would be returned. If, on the other hand, the user entered “USA,” the center of the country would be returned along with a much larger radius representing and average radius of the country. It is assumed that there is equal probability of the user being anywhere within that resulting disc. A more refined approach could use actual normalized density maps (i.e. converted to the probability of locating individuals in named region at a particular point *x* in that region), when available for the different localities, as the underlying kernels. For purposes of demonstration of our contingent estimation method, though, we simply used a Gaussian kernel basis for the estimation process. We implemented in R both bivariate standard and contingent kernel density estimation techniques for the sample of 10,000 Twitter users. In comparing the results, as discussed below, we identified some critical differences.

Several critiques can be made of using one of the standard family rather than special sets of kernels that conform to regional boundary constraints in our contingent kernel density estimation, as would be the case if kernels were normalized population density maps used for each named region. Even if density maps were not available, it would still be better to use a compact kernel that coincides with the boundaries of the named region in place of a disc that is maybe a very poor approximation to this boundary. Currently the boundaries for certain classes of regions, such as municipalities, are difficult to obtain. Also, use of a set of special but irregular kernels would require a numeric approach, instead of an analytical one, which would greatly slow down computation of a contingent kernel estimate using Equation 4.

Another potential critique is that instead of assuming an equal probability of a user within a region, census or other demographic data could be used to create more accurate predictions of the distribution of users within regions. This modified approach has two primary potential weaknesses. One is that the distribution of

Twitter users could be fundamentally different from the distribution obtained by a census. Factors that result in an individual using Twitter could potentially result in them have a different geographic distribution than general surveys report for the population at large. Secondly, it can be hypothesized that the level of specificity with which a user enters their location depends on that location itself. For instance, take two hypothetical users living in California. One lives in San Francisco while the other lives in a small town in the Central Valley. A priori, one could hypothesize that the user living in San Francisco would be more likely to specify their city (because they know other Twitter users will have heard of it and so it would then be meaningful to them) while the user living in the small town would be less likely to specify their town (because they know it would likely not have meaning to other Twitter users). In this hypothesis were true, the usage of demographic data could result in bias towards large and popular cities and regions to a greater extent then the assumption of a uniform distribution throughout a region.

Results

Calculation of Demonstrative Contingent Kernels

An example calculation of a contingent kernel (as defined using Equation 3) derived for a standard Gaussian kernel $K\left(x\right)=\frac{e^{-x^2/2}}{\sqrt{2\pi}}$ and an error probability density function defined as a uniform distribution (Ψ) with a half-width of *r* is provided in Equation 6.

$$C(x|h,U)=\frac{\displaystyle\int\frac{e^{-\left(\frac{x-\psi}{\sqrt{2}h}\right)^2}}{\sqrt{2\pi}}\cdot\left\{\begin{matrix}1/2r & |\psi|\le r\\ 0 & |\psi|>r\end{matrix}\right\}d\psi}{\displaystyle\iint\frac{e^{-\left(\frac{x-\psi}{\sqrt{2}h}\right)^2}}{\sqrt{2\pi}}\cdot\left\{\begin{matrix}1/2r & |\psi|\le r\\ 0 & |\psi|>r\end{matrix}\right\}d\psi\,dx}
=\frac{\mathrm{Erf}\left[\frac{r-x}{h\sqrt{2}}\right]+\mathrm{Erf}\left[\frac{r+x}{h\sqrt{2}}\right]}{4r} \qquad (6)$$

Additionally, contingent kernels for a number of common usage cases were calculated. Combinations between three separate kernels (Gaussian, Epanechnikov, and Uniform) and two types of errors (Uniform defined by a half width *r* and Gaussian defined by a standard deviation σ) were developed. The resulting contingent kernels are shown in Table 14.

Table 14. Contingent kernels (C) for combinations of univariate standard kernels (K) and two forms of errors (*Ψ*).

		Ψ	
		Uniform Error U **: half-width (*r*)**	**Gaussian Error** U **: standard deviation (**σ**)**
K	**Gaussian Kernel**	$\frac{\text{Erf}\left[\frac{r-x}{h\sqrt{2}}\right]+\text{Erf}\left[\frac{r+x}{h\sqrt{2}}\right]}{4r}$	$\frac{e^{\frac{-x^2}{2(h^2+\sigma^2)}}}{\sqrt{2\pi}\sqrt{h^2+\sigma^2}}$
	Epane-chnikov Kernel	$\begin{cases} \frac{\text{Cos}\left[\frac{\pi x}{2h}\right]\text{Sin}\left[\frac{\pi r}{2h}\right]}{2r} & \lvert x\rvert \le h-r \\ \frac{1}{2r} & \lvert x\rvert \le r-h \\ \frac{1+\text{Sin}\left[\frac{\pi(r+x)}{2h}\right]}{4r} & 0<-x<h+r \\ \frac{1+\text{Sin}\left[\frac{\pi(r-x)}{2h}\right]}{4r} & 0<x<h+r \\ 0 & \text{otherwise} \end{cases}$	$\frac{\pi}{16h}\left(\text{Erf}\left[\frac{2h^2+2hx-i\pi\sigma^2}{2\sqrt{2}h\sigma}\right]+\right.$ $\text{Erf}\left[\frac{2h^2-2hx+i\pi\sigma^2}{2\sqrt{2}h\sigma}\right]+$ $\left(\text{Erf}\left[\frac{2h^2+2hx+i\pi\sigma^2}{2\sqrt{2}h\sigma}\right]+\text{Erf}\left[\frac{2h^2-2hx-i\pi\sigma^2}{2\sqrt{2}h\sigma}\right]\right)\cdot$ $\left(\text{Cos}\left[\frac{\pi x}{h}\right]+i\,\text{Sin}\left[\frac{\pi x}{h}\right]\right))\cdot$ $\left(\text{Cosh}\left[\frac{\pi(4ihx+\pi\sigma^2)}{8h^2}\right]-\text{Sinh}\left[\frac{\pi(4ihx+\pi\sigma^2)}{8h^2}\right]\right)\Bigg)$
	Uniform Kernel	$\begin{cases} \frac{1}{2h} & \lvert x\rvert < h-r \\ \frac{1}{2r} & \lvert x\rvert < r-h \\ \frac{h+r+x}{4hr} & 0<-x<h+r \\ \frac{h+r-x}{4hr} & 0<x<h+r \\ 0 & \text{otherwise} \end{cases}$	$\frac{\text{Erf}\left[\frac{h-x}{\sigma\sqrt{2}}\right]+\text{Erf}\left[\frac{h+x}{\sigma\sqrt{2}}\right]}{4h}$

These contingent kernels directly take the role of the contingent kernel C in Equation 4 for calculating the contingent kernel density estimate. In some cases, such as a Gaussian kernel and a Uniform error, the contingent kernel is a simple function. In other cases, such as the Epanechnikov kernel and a Uniform error, the contingent kernel is a piecewise function where test conditions should be evaluated sequentially until one is determined to be true. In either case, the implementation of these contingent kernels in a mathematical or programming environment is straightforward.

All contingent kernels use the bandwidth as a parameter. Figure 21 illustrates the effect of the bandwidth parameter on the shape of the contingent kernel formed from the standard Gaussian kernel and a Uniform error (Equation 6). The parameters defining the error probability density function are also parameters to the contingent kernels and affect their shape.

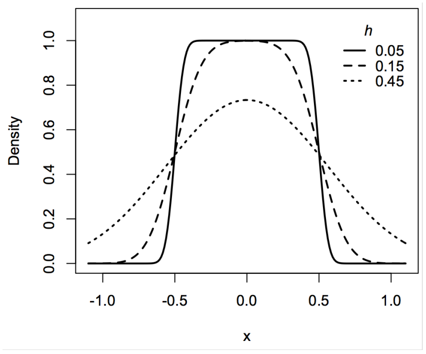


Figure 21. Example contingent kernel. The contingent kernel for a standard Gaussian kernel applied to a uniform half-width of radius 0.5. Contingent kernels for three different bandwidths (*h*) are plotted.

It should be noted that not all the contingent kernels are defined for the Uniform error distribution when the width of the Uniform distribution is equal to 0 (i.e. $r = 0$). Such a scenario would correspond to the case of no measurement error. Without measurement error, we would wish that the contingent kernel would reduce to the standard kernel that it was derived from. Indeed, this is the case. For example, although Equation 6 is not defined for $r = 0$, the limit of the equation as r approaches 0 reduces to the equivalent of what would be the standard Gaussian kernel (Equation 7). Code that implements the contingent kernels should check for such boundary conditions and adjust calculations accordingly.

$$\lim_{r\to 0}\frac{\mathrm{Erf}\left[\frac{r-x}{h\sqrt{2}}\right]+\mathrm{Erf}\left[\frac{r+x}{h\sqrt{2}}\right]}{4r}=\frac{e^{-\frac{x^2}{2h^2}}}{h\sqrt{2\pi}} \tag{7}$$

Numerical Validation of Accuracy Improvements

As described in the methods section, a numerical simulation was constructed to validate the accuracy of contingent kernel density estimation on synthetic datasets.

In this experiment, the Mean Integrated Square Error (MISE) of both the standard kernel and contingent kernel estimates fell as the sample size of points drawn from the original distribution increased (Figure 22). Furthermore, although the standard and contingent methods are similar in accuracy for small sample sizes, the contingent kernel error falls faster compared to the standard kernel, as the sample size increases. At large sample sizes, the contingent kernel has approximately one-half the MISE as the standard kernel.

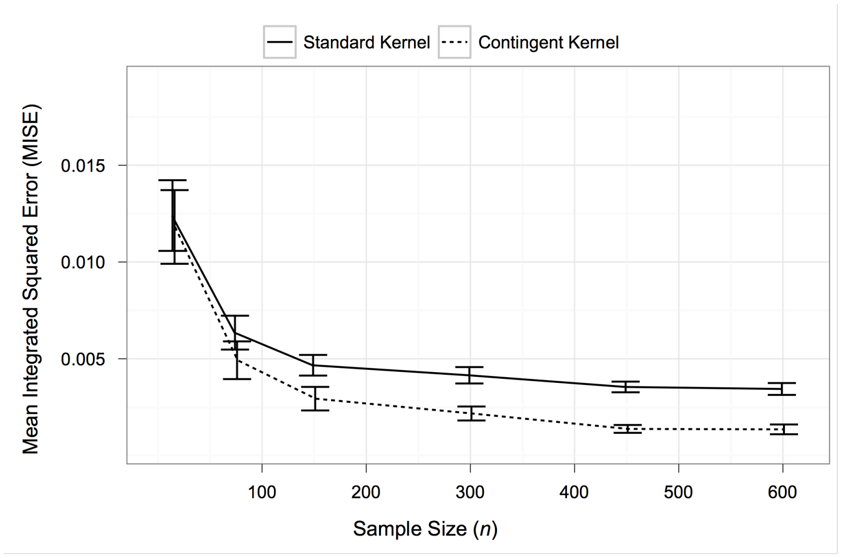


Figure 22. Error for standard and contingent kernels for different sampling sizes from test distribution. Test distribution is a mixture function of three normal distributions. Each normal distribution was sampled with a different frequency. From left to right (2-unit bins, 1-unit bins, and 0.5-unit bins). Mean Integrated Square Error and 95% confidence intervals for the standard and contingent kernels are plotted. 20 Monte Carlo simulations were carried out for each sampling size.

A better qualitative understanding of the methods is obtained by looking at the exact estimates produced by the standard and contingent kernel techniques (Figure 23). The standard kernel method is more susceptible to overestimation of the density at the locations where the points are relocated to, due to the sampling grid filter. The standard method also exhibits much higher variability. The contingent kernel method, on the other hand, reduces variability at these locations by smoothing the kernels based on uncertainty. Even as it smooths in these places, the contingent kernel method still does a better job than the standard kernel method at identifying the spike in density at location 0. Since that peak is not associated with high sampling uncertainty, little contingent smoothing occurs at the peak.

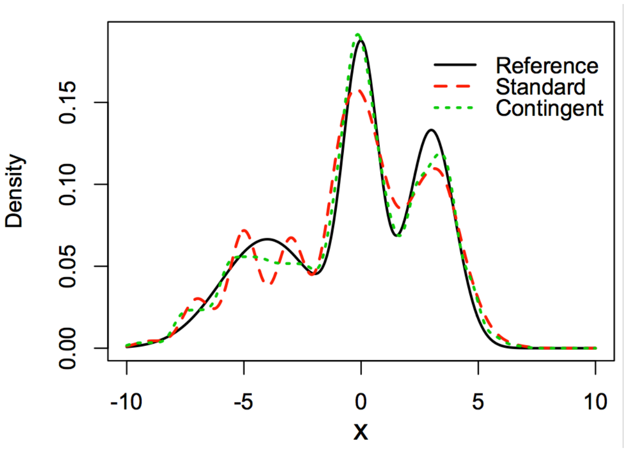


Figure 23. Sample standard and contingent kernel density estimates of test distribution. The distribution is the same as in Figure 4. Sample size is 600 points. The contingent kernel has significantly lower error compared to the standard kernel.

Application: Twitter User Locations

Data from 10,000 Twitter users were analyzed. As discussed in the methods section, location information for each user was represented as a disc with the user having equal probability of being located anywhere in the disc. Figure 24 contains both a plot of the center point for each user and also the full disc that defines the location of the users. Points and discs are jittered and plotted with slight transparency to allow the estimation of the number of users at dense locations.

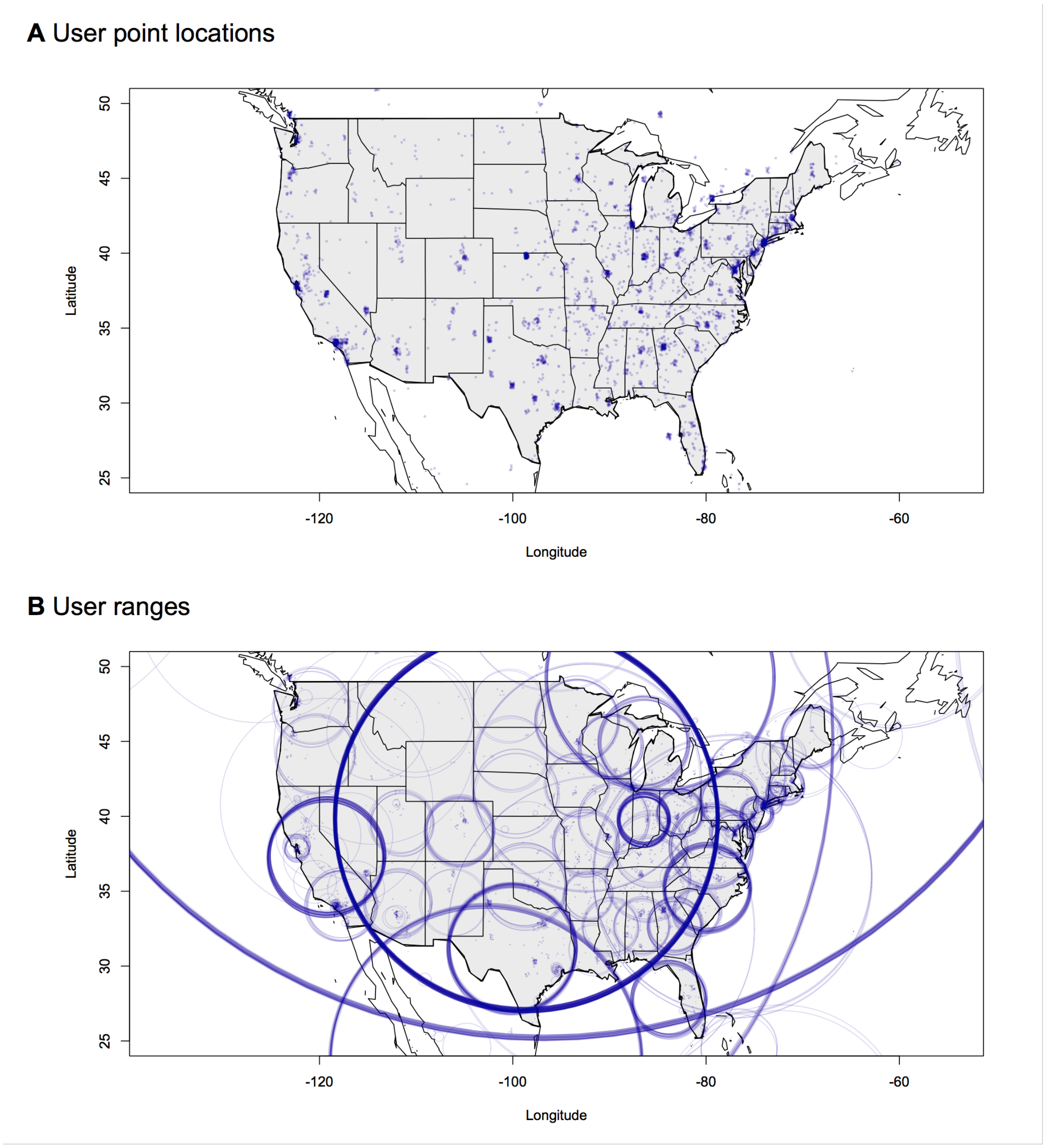


Figure 24. Plots of a subsample of 10,000 Twitter user locations. Panel A is the plot of just the center of the locations. This is what a standard kernel estimate would use to estimate distribution. Panel B is a plot of the discs defined by the center and the radius of uncertainty. This is what the contingent kernels use to estimate distribution. In panel A, note the high concentration of users on the Kansas-Nebraska border. This is due to users entering "USA" as their location and should be paired with high uncertainty. It is converted to the large ring shown in panel B. Points and discs are jittered by up to 2 degrees to improve the readability of the results.

Standard kernel density estimation uses the center of the discs. As can be seen in panel A of Figure 24, using this criterion there is a heavy concentration of users directly on the Kansas-Nebraska border. In reality, there is no such heavy concentration of Twitter users in that location, instead the heavy density there is due to users who specified their location as "USA." The Kansas-Nebraska border is the rough center of the United States and so Twitter users whose location is only identified as the country are assigned to that location. When, the discs are viewed in panel B of Figure 24, the spike on this border disappears as it is transformed to a set of large discs encompassing most of the United States. Similar spikes in densities can be seen in panel A at the center of California or in the center of Texas. When the discs are plotted instead, these spikes disappear and become discs approximately encompassing these states.

Standard and contingent kernel density estimates were carried out for these data using a bivariate Gaussian standard kernel and its contingent kernel equivalent (a scaled rotation of Equation 6). The standard kernel density estimate clearly demonstrates strong bias and inaccuracy in that the spike of users who select the United States as their location is represented in the standard kernel density estimate as a large increase in density on the Kansas-Nebraska border (Figure 25). This density spike is approximately equivalent to the spike in density located at the San Francisco Bay Area, a region that is the headquarters of Twitter and that is known for its relative high population of technologically savvy Twitter users. Clearly this is quite unrealistic and is just an artifact of the sampling regime.

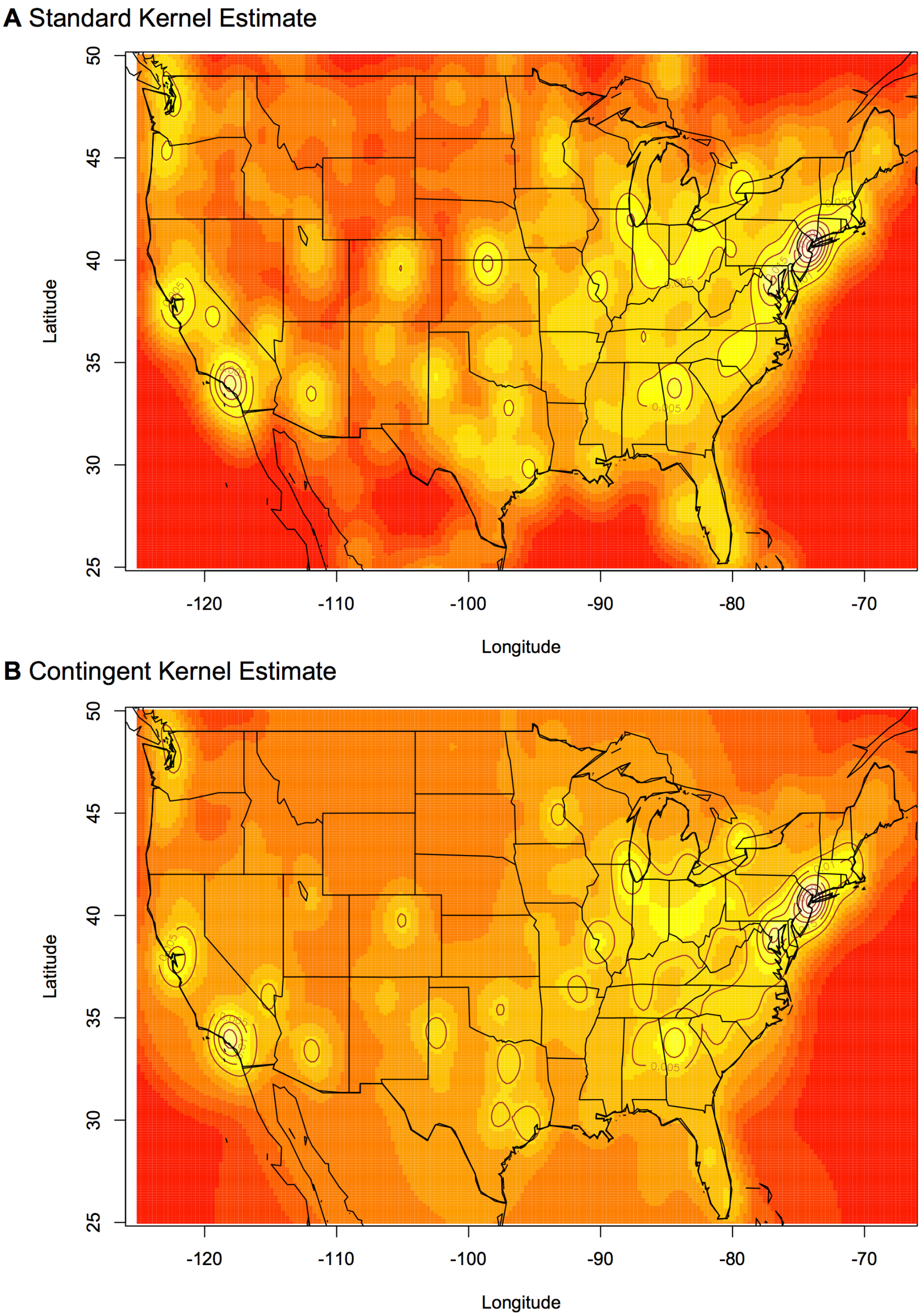


Figure 25. Standard and contingent kernel density estimates of the distribution of 10,000 Twitter users in the United States. Note the high-level of density on the Nebraska-Kansas border found in the standard kernel estimate. In the standard kernel, this level of density is equivalent to the level of density in the San Francisco Bay Area: an area with a known high-level of Twitter use. This is an artifact caused as a result of failing to take the uncertainty of users locations into account. The artifact disappears in the contingent kernel estimate.

In the contingent kernel density estimate, this sampling artifact completely disappears. In addition, population densities and cities with an expected high number of Twitter users such as Denver, Colorado, are better identified in the contingent kernel density estimate than in the standard kernel density estimate.

Discussion and Conclusions

The contingent kernel density estimation technique has been shown to be effective in compensating for certain forms of errors such as those observed in the Twitter location dataset or those associated with Transect Range and Section grids. It is primarily a narrative method that allows the integration of diverse data sources, each generated with various levels of measurement confidence and potentially different types and structures of uncertainty. Furthermore, it is more straightforward to derive contingent kernels for standard kernels and elementary error distributions than it is to derive some other adjustment methods such as deconvolution kernels. As error distributions become more complex (for instance when the error region is defined using an arbitrary geographic boundary such as a country border), deriving contingent kernels analytically may be infeasible but numerical approximations can be used to estimate the contingent kernel.

The primary limitation of this method now needs to be addressed: the contingent kernel is dependent upon the error distribution. In this chapter's examples, the error distribution was treated as a uniform function (the computer simulation validation experiment) and as a uniform disc (the applied Twitter analysis). This error distribution will be referred to as $\hat{\Psi}$. The error distribution that appears in Equation 3 is, for each point, in reality the normalized product of $\hat{\Psi}$ at that point and original probability density function (f) (Equation 8).

$$\Psi(x) = \frac{\hat{\Psi}(x) f(x)}{\int \hat{\Psi}(x) f(x) dx} \qquad (8)$$

Since f is unknown, Ψ cannot be directly calculated. By using $\hat{\Psi}$ in place of Ψ, the results are biased. The significance of this bias depends on both $\hat{\Psi}$ and f. As shown in the computer simulation and in the applied Twitter application, even given the use of $\hat{\Psi}$, the contingent kernel density estimation can still result in superior density estimates as compared to standard kernel density estimation. This is because in these cases f has a relatively small effect on Ψ so $\hat{\Psi}$ and Ψ are quite similar. However, other combinations of $\hat{\Psi}$ and f can result in situations where the use of $\hat{\Psi}$ as an approximation of Ψ will result in contingent kernel density estimates that are potentially worse than their standard kernel counterparts. A hypothetical example of such a case is when f is a sharp normal distribution while Ψ is a much broader normal distribution. In this case, f will have a strong effect on the shape of f and the use of $\hat{f}$ as its approximation is insufficiently accurate.

One subsequent goal then, is the determination of better approximations of f from $\hat{f}$ and the data. One possible method that this chapter has not explored would be to carry out an iterative analysis where the initial contingent kernel density estimate f is used in place of f in Equation 8 to generate more accurate error distribution functions, which are, in turn, used to construct an improved contingent kernel density estimate. Potentially, this procedure could be carried out with multiple iterations and adjustments to the error function.

In general, it can be assumed that any reported measurement was made with some error. Sometimes these errors are purely additive such as when observations were displaced using a normal error distribution from their original location. Other times, errors are a result of a procedure where observations have been aligned to a grid such as in the computer validation example developed in this chapter. In other cases, errors may be due to selected effort or observation bias leading points to be detected with different frequency in different regions. Whatever the form, whether or not these errors have a significant result on any analysis must be evaluated on a case-by-case basis. If it is determined that the errors could significantly impact the analysis, the choice of analytical tools must be adjusted to account for this impact. This chapter develops one such tool: a narrative method for adjusting for errors that are found in certain types of sampling regimes. Studies that use such sampling regimes can make use of this technique to improve the accuracy of their analyses.

References

Chang, C, and R Ansari. “Kernel Particle Filter for Visual Tracking.” *Ieee Signal Processing Letters* 12.3 (2005): 242–245. Web.

Delaigle, A, and I Gijbels. “Frequent Problems in Calculating Integrals and Optimizing Objective Functions: a Case Study in Density Deconvolution.” *Statistics and Computing* 17.4 (2007): 349–355. Web.

Fan, J. “Asymptotic Normality for Deconvolution Kernel Density Estimators.” *Sankhya-the Indian Journal of Statistics Series a* 53 (1991): 97–110. Print.

Fan, J. “Global Behavior of Deconvolution Kernel Estimates.” *Statistica Sinica* 1.2 (1991): 541–551. Print.

Fortmann-Roe, Scott, Richard Starfield, and Wayne M Getz. “Contingent Kernel Density Estimation.” *PLoS ONE* 7.2 (2012): e30549. Web.

Hall, P et al. "On Optimal Data-Based Bandwidth Selection in Kernel Density Estimation." *Biometrika* 78.2 (1991): 263. Print.

Horne, JS, EO Garton, and KA Sager-Fradkin. "Correcting Home-Range Models for Observation Bias." *Journal of Wildlife Management* 71.3 (2007): 996–1001. Print.

Jones, MC. "Kernel Density-Estimation for Length Biased Data." *Biometrika* 78.3 (1991): 511–519. Print.

Jones, MC. "The Roles of ISE and MISE in Density-Estimation." *Statistics & Probability Letters* 12.1 (1991): 51–56. Print.

Jones, MC, JS Marron, and SJ Sheather. "A Brief Survey of Bandwidth Selection for Density Estimation." *Journal of the American Statistical Association* 91.433 (1996): 401–407. Print.

Jones, MC, JS Marron, and SJ Sheather. "Progress in Data-Based Bandwidth Selection for Kernel Density Estimation." *Computational Statistics* 11 (1996): 337–381. Print.

Marron, JS, and AB Tsybakov. "Visual Error Criteria for Qualitative Smoothing." *Journal of the American Statistical Association* 90.430 (1995): 499–507. Print.

Millspaugh, JJ et al. "Analysis of Resource Selection Using Utilization Distributions." *Journal of Wildlife Management* 70.2 (2006): 384–395. Print.

Powell, James L, James H Stock, and Thomas M Stoker. "Semiparametric Estimation of Index Coefficients." *Econometrica* 57.6 (1989): 1403–1430. Print.

Seaman, DE et al. "Effects of Sample Size on Kernel Home Range Estimates." *The Journal of Wildlife Management* 63.2 (1999): 739–747. Print.

Sheather, SJ, and MC Jones. "A Reliable Data-Based Bandwidth Selection Method for Kernel Density Estimation." *Journal of the Royal Statistical Society. Series B (Methodological)* 53.3 (1991): 683–690. Print.

Silverman, BW. *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall, 1986. Print.

Stefanski, L, and RJ Carroll. "Deconvolving Kernel Density Estimators.." *Statistics: A Journal of Theoretical and Applied Statistics* 21.2 (1990): 169–184. Print.

The Museum of Vertebrate Zoology at Berkeley. "MVZ -- Fieldnotes, Photos, & Map Collection." *http://mvz.berkeley.edu/FieldnotePhotoMap_Collection.html*. N. p., May 2011. Web. May 2011.

Worton, BJ. "Kernel Methods for Estimating the Utilization Distribution in Home-Range Studies." *Ecology* 70.1 (1989): 164–168. Print.

5. Effects of Hue, Saturation and Brightness on Color Preference in Social Networks[19]

Introduction

The final research chapter of this dissertation explores parameter estimation in a case that is conceptually more straightforward than the earlier chapters but which is made more difficult due the quantity of data being analyzed. As demonstrated in this chapter, "big data" is opening exciting new avenues for developing models or estimating properties of a population, but it also poses its own unique challenges that can significantly complicate even relatively straightforward analyses.

Color is a key method of self-expression and understanding this self-expression provides the basis for this chapter. People express themselves through the colors of the clothes they wear, the car they buy, and even the color they use to paint their house. Conversely, color is one of the characteristics that we use to evaluate others. Assumptions we make about a woman dressed all in black are likely to be different than those made for a woman dressed in a highly saturated red.

Consequently, the choice of colors on public-facing surfaces is both a function of personal preference and personal expression of identity. A number of studies have explored color preferences (Camgöz, Yener and Güvenç 2002, Child, Hansen and Hornbeck 1968, Cubukcu and Kahraman 2008, Eysenck 1941, Guilford and Smith 1959, Manav 2007). Many describe color in terms of hue, saturation and brightness; terms that are easily accessible to the layperson. In general, studies found that blue hues were most favorable, while yellow and yellow-green hues were least favorable (Camgöz, Yener and Güvenç 2002, Eysenck 1941, Guilford and Smith 1959). On average, brighter and more saturated colors were preferred (Camgöz, Yener and Güvenç 2002, Child, Hansen and Hornbeck 1968). Color preference comparison between genders has shown a variety of differences in color preference between males and females (Child, Hansen and Hornbeck 1968, Eysenck 1941, Helson and Lansford 1970) while some studies have shown no differences (Camgöz, Yener and Güvenç 2002).

Nevertheless, the experimental designs of some studies could have induced bias in subject behavior. For instance, in most studies which present static color swabs, yellow is the least preferred, yet in a study designed to determine preferences for building exterior colors, it was determined to be the most preferred (Cubukcu and

[19] A version of this chapter was previously published as Fortmann-Roe, S. (2011). Effects of hue, saturation, and brightness on color preference in social networks: Gender-based color preference on the social networking site Twitter. Color Research and Application.

Kahraman 2008). Additionally, full saturation and brightness were not found to be preferred color choices in architectural situations (Manav 2007, Cubukcu and Kahraman 2008). In short, subjects may prefer one color in a lab-based, controlled study, but prefer a different one for clothing or other applied choices. Thus it continues to be important to look at real-world color selection preferences.

In the past decade social networking sites; such as Facebook, Twitter and MySpace; have become popular within the United States and the larger world. Such sites enable users to connect and share through a digital medium. The sites offer the ability to create a "Profile Page" where users present themselves and share information about their lives. Each site has different customization and design capabilities for profile pages. Of the three listed, Facebook is the most restricted and only allows users to change their personal information such as location and job. MySpace is more flexible and enables users to completely redesign their page. Twitter offers a middle ground, providing a restrictive template design, but allowing users to specify the colors of the key portions of their page such as the background and text colors.

When a Twitter user chooses colors for their profile page, the choice expresses both color preference and how they wish to present themselves. A literature search has revealed no published studies exploring color choice on social networks such as Twitter. For color research, Twitter provides us a natural experiment to better understand user preferences for key color attributes and gender differences in these preferences among a massively wide audience working in real time. In September 2010, Twitter alone had over 300 million registered users (Twitter Inc. 2010). This large sample base allows us to obtain highly precise and statistically significant results.

Methods

Interfacing with Twitter

Twitter releases several publically accessible Application Programming Interfaces (API's) that may be used to extract information about its users. To access these API's, a script was developed using the programming language PHP (The PHP Group 2010). The script downloaded information about individual status updates posted by Twitter users (known as "Tweets"), which it then aggregated into information about individual users. The script extracted metadata about these users including the users' first names and information about what images and colors they had chosen for their Twitter profile page.

This script ran continuously from October 4, 2010 to October 29, 2010. During that time, information for 1,321,174 individual Twitter user accounts was downloaded. This chapter treats these users as a sample representative of the universe of Twitter users in late 2010.

Twitter provides the following information about each user's homepage design:

- Page background color,
- text color,
- hyperlink color,
- sidebar color,
- sidebar border color,
- background image and whether it should cover the whole background.

The page background color was treated as the representative color of the entire page. This is a safe assumption when the background image is not tiled to cover the whole page and the analysis was restricted to just those users for whom this was true. A visit to a Twitter profile page can verify this assumption as the background color covers the majority of the non-white surface that cannot be edited by the user.

Data Management and Analysis

Given the high number of pieces of information downloaded from Twitter (status updates were collected at a rate of up to 300,000 a day) and the resulting large amount of data collected, data management and analysis were a challenge. Standard tools such as an Excel spreadsheet could not handle the large amount of raw data, so it was instead stored in the open-source, database MySQL which is designed for very large datasets (Oracle 2010).

Twitter does not specifically report user gender. To determine gender, the first names of downloaded Twitter users were analyzed using an automatic classification algorithm. The open-source "GenderforName" PHP application was employed (Warden, Orwant and Daly 2009). This application applies a number of heuristics to determine gender based on a subject's first name. The application was run on a high level of accuracy in order to minimize potential errors even if it reduces the ability to determine gender for a large number of users. Of the 1.3 million Twitter accounts collected, a little under one-half (605,894) could be classified as either male of female by the application at this high level of accuracy. Twitter does not provide any verification of the names that users use (except for certain celebrity Twitter accounts) so it is inevitable that at least some users use fake names. It is a safe assumption, however, that the proportion of such users within the larger sample is small and it is certain that the proportion of users that use fake names of the opposite gender is even smaller.

Analysis was carried out both directly within MySQL and the scientific computation software R (R Development Core Team 2010). MySQL contains features for calculating simple descriptive statistics such as averages and standard deviations. For more complex analyses and hypothesis testing, subsets of the database were exported to R. Hue density plots were created using an R package called Circular (Lund and Agostinelli 2010).

Color Model

Twitter allows users to select individual colors in two ways. The primary method of entry is a Hue-Saturation-Brightness (HSB) color selection interface (Figure 1). HSB is a standard method of color entry for computer-based applications. It is a simple transformation of Red-Green-Blue (RGB) color space that preserves its gamut and is thus well suited for representing colors which will be displayed on a digital screen using red, green and blue light emitters. Hue-Saturation-Value is most often a synonym for the HSB color model. Hue-Saturation-Lightness shares many features with HSB but its transformation to and from RGB space is different leading to slightly different properties (Levkowitz and Herman 1993).

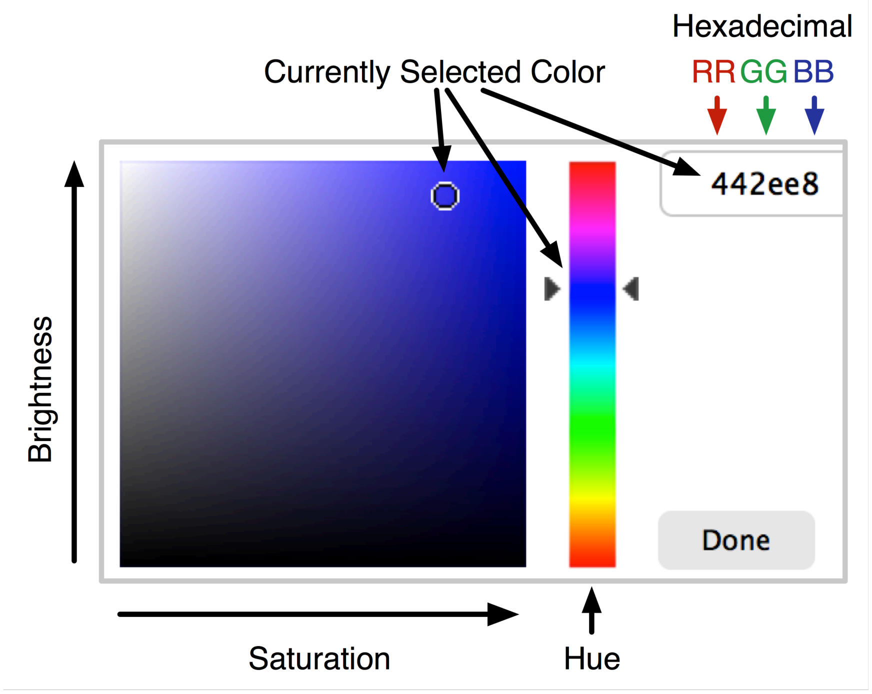


Figure 1. The Twitter color selection interface. Note the two methods of entering colors: a graphical Hue-Saturation-Brightness method, and a Red-Green-Blue hexadecimal entry method. To enter Hue-Saturation-Brightness colors, the user selects a hue with the slider and independently selects the saturation and brightness in the rectangular space.

The second method of color entry that Twitter supports is using an RGB color model and requires the users to enter the Red, Green, and Blue components directly as hexadecimal numbers between 0 and 255. A number between 0 and 255 can be represented using a two-digit hexadecimal number. These three pairs of two digits are then combined into a six-character representation of the RGB color.

The hexadecimal RGB mode of representing colors is the default for many Internet infrastructure technologies such as HTML. However, the HSB selection interface Twitter provides is much more natural to laypersons and is likely to be highly preferred when selecting colors. The relative sizes of the HSB and RGB selection areas in the Twitter interface indicate the relative importance of the two methods to Twitter users.

Since Twitter uses an HSB model and provides the color selection interface for it, this chapter carries out its analysis of colors by analyzing the hue, saturation, and brightness components both individually and jointly.

Twitter Themes

In addition to enabling its users to directly specify colors, Twitter allows them to select between roughly a dozen prebuilt themes for a preferred arrangement of colors and graphical designs.

This provides a potential confounding factor in the analysis. If, for example, we observe that a high number of Twitter users prefer shades of blue, is that because they actually prefer blue or because there are more blue themes than themes of other colors? To compensate for this, most analyses in this chapter are run twice: once for all users irrespective of whether they are using a theme or directly specified their colors, and a second time for only users who are not using one of the prebuilt themes.

Twitter does not directly report whether user colors were chosen by the user or by a theme, but it is possible to identify theme colors based on the number of users with identical colors. Twitter supports 16,581,375 (255^3) unique colors so the odds of many users choosing identical colors by chance are small. If many users have chosen the same color it is probable they are using the same theme. In this analysis, colors where more than 100 users choose the same color are considered to be colors chosen by a theme and are excluded from the latter analyses. Sensitivity testing on this cutoff point showed that results were resilient to changes in the cutoff.

Results

Results on color preference by male and female Twitter users are summarized in Table I. As demonstrated, there is a high degree of statistical significance as all differences were significant at a 5% significance level and all but one were significant at a much higher level. Indeed, the large sample size was chosen to ensure statistical significance in findings. Key points follow.

- Women are more likely than men to customize their profile page and move away from the default theme that Twitter provides to users when they first sign up for a Twitter account.

- Males appear to prefer darker colors than females. The difference between the average brightness for the samples of males and females was 16.7% for the case when the default themes were included in the analysis and 9.4% when they were excluded. Confidence bands on these estimates are very narrow with a minimum difference for a 99% confidence band of 8.5%. Figure 2 graphically illustrates the difference between these two brightnesses for the case where default themes were included in the analysis.
- Males appear to prefer slightly more saturated colors than females. The average difference for the saturation of the male and female samples is 0.4% and 3.2% for the cases of including and excluding the default themes respectively. Though statistically significant, this difference appears to be of little practical significance.

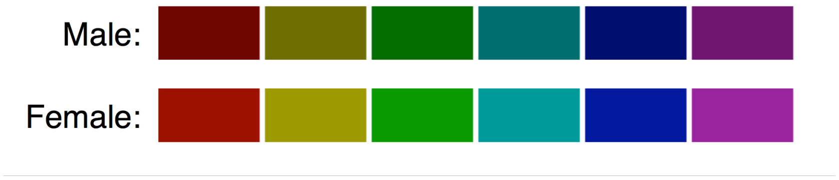


Figure 2. Comparison of fully saturated colors at the mean male sample brightness (44.6%) and the mean female sample brightness (61.3%). These means are for the case when themes are included in the calculation of mean brightness. (N Male = 93,334; N Female = 102,342).

In addition to the differences between the averages of users brightness and saturation choices, it is illuminating to explore the distribution of their choices within the color space. Figure 3 contains bivariate kernel density plots showing the distribution of male and female color choices within a two-dimensional brightness and saturation space. The volume under a given area of the surface is proportional to the preference of users for that area. If the volume under a given "spike" in the surface is 20% of the volume for the entire color space, it indicates that one-fifth of users chose a color in that region.

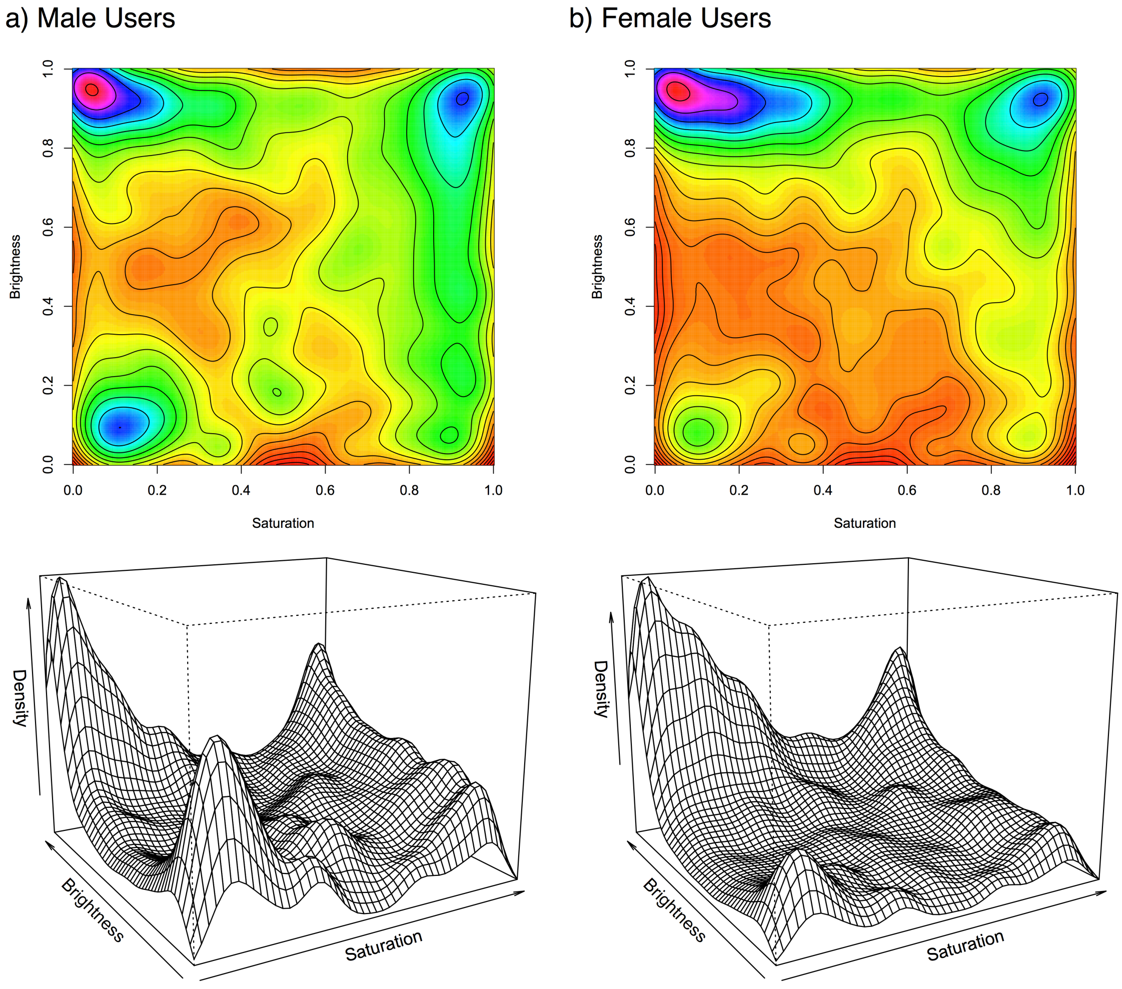


Figure 3. Bivariate kernel density plots of male and female preference in Brightness-Saturation space. A Gaussian kernel was used. Volume under an area is proportional to preference for that area. Both 3D surface plots and 2D contour plots are presented and represent the same data. Subfigure 3.a shows male preferences, subfigure 3.b shows female preferences. Themes were excluded from these analyses. (N Male = 14,613; N Female = 19,385)

As demonstrated by comparing the male and female plots, there are both similarities and differences between the gender-specific choices.

- Both sexes had spikes of preferences at the following locations: low-brightness and low-saturation, high-brightness and low saturation, and high saturation and high brightness.
- Nevertheless, the magnitudes of the peaks are different. It appears that men preferentially choose low-brightness and low-saturation colors more than

women. Additionally, women have a ridge of preference at high-brightness which men lack.

- On the other hand, men have a lesser, but still-significant, ridge of preference for high-saturation colors which women lack.

Analysis of hue preference was carried out by plotting circular kernel density plots for the male and female samples and for the cases of including and excluding themes (Figure 4). As with the bivariate plots, the area under the curves for a region is proportional to preference.

Taking the case of the inclusion of default themes first. Each spike in Figure 4.a corresponds to one or more themes. It is clear that there are both hue preferences and also gender differentiated selection. Shades of blue and red are the most popular hues. The yellow and blue-green hues are the least popular of the colors for which there were clearly identifiable themes (i.e. spikes in Figure 4.a). Green and magenta register almost no selection at all possibly because there were no themes of those hues. Males preferentially choose the blue themes more than women while women preferentially choose the red and orange themes to a greater extent than men.

Since the choice of themes is a confounding factor, the analysis was rerun with the theme color choices deleted. Figure 4.b illustrates the hue choices when the themes have been removed from the analysis. Comparing 4.a to 4.b, we can see that the spikes corresponding to the themes have been removed from this latter figure. As demonstrated, when users specify their own background colors without resorting to prebuilt themes, their choices are similar but different in some key aspects.

- Blues and reds are the most preferred hues.
- Males prefer blue to a greater extent then women.
- Magentas, and specifically pink hues, are selected by both males and females but much more so by females.
- Green hues are generally not selected.

The differences between males and females for both Figures 4.a and 4.b are highly statistically significant. A Kolmogorov-Smirnov Goodness of Fit Test yielded a *P* value of less than 10^{-5} for both of them. The practical significance between the two is also readily visible to the reader. Table II summarizes the information in Figure 4 and contains the percent of males and females that selected for the range of hues that can be broadly classified as red, yellow, green, cyan, blue and magenta.

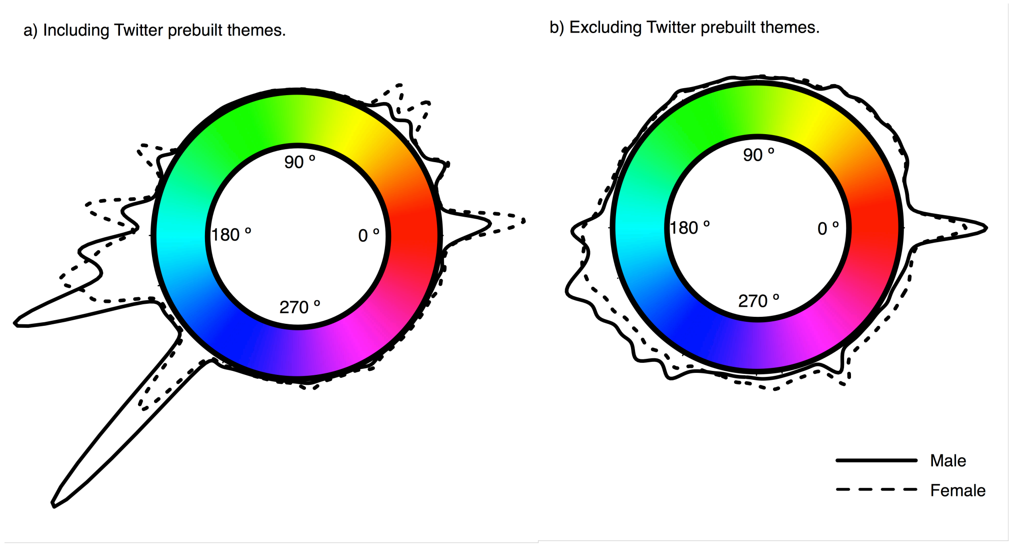


Figure 4. Circular kernel density plots of hue preference. Areas under the curves are proportional to the preference for each portion of the curve. The solid line is the distribution for the male sample, and the dashed line is the distribution for the female sample. Subfigure 4.a illustrates the preferences when Twitter theme color choices are included in the analysis. Subfigure 4.b illustrates the preferences when theme color choices are excluded from the analysis. A von Mises kernel was used in both cases. For both subfigures, the differences between the male and female samples were determined to be statistically significant at P values greater than 10-10 using Kolmogorov-Smirnov Goodness of Fit Tests. (Subfigure 4.a: N Male = 74,632; N Female = 82,873. Subfigure 4.b: N Male = 14,613; N Female = 19,385)

Discussion

This chapter explored color preference through the choices users of the social network Twitter make selecting the colors for their profile pages. The results broadly agree with previous research. Blues were the most preferred hues as previous research has suggested. Yellows and greens were some of the least favorite colors.

The results did not show as some previous studies have done (Camgöz, Yener and Güvenç 2002) that brighter and more saturated colors are preferable. The mean saturation and brightness in this chapter hover around 50%. Possibly this illustrates the difference between laboratory experiments where subjects are attracted to powerful stimuli and real-world color choices where intense colors can be overbearing or distracting. These findings agree generally with other studies that looked at real-world application of color preferences (Manav 2007, Cubukcu and Kahraman 2008).

It is also important to note that the high preference for a hue of 0° (pure red), could potentially be due to an artifact of the construction of the Twitter color selection interface (Figure 1). The hue slider in this interface ranges from 0° to 360°. Thus, the ends of the slider can in effect serve as a magnet when the user selects a color. If the user is attempting to select a red hued color, they will drag the slider towards one end of the control. If they overshoot the area of the control, the slider will be locked into a value of 0° or 360°. If the slider had been constructed without this artificial characteristic – for instance as a perfect circle – it can be predicted that the spike in preference currently located at 0° would be lower and wider as people would be less likely to select the exact boundaries of the spectrum.

It is interesting to note that the highest proportional gender difference in hue selection between male and female users was for the magenta hue class. Women preferentially select this hue roughly three times more than men. This is possibly because this region contains colors similar to pink which are strongly identified as a feminine color (Bridges 1993). Blue, which is often identified as a masculine color, is heavily selected for by men compared to women but the proportional difference is not as great.

Although this chapter reports a significant difference for the probabilities that males and females will change away from their default theme, the cause is not clear. It is colloquially assumed that women are more concerned about appearance than men (Pliner, Chaiken and Flett 1990), and this statistic would seem to support that assumption. However, the default Twitter theme that all new users are assigned when they sign up for an account is blue. As shown in Figure 4, this is a color preferred to a greater extent by men rather than women. Thus the difference in changing colors could be due to men preferring the default theme more than women, rather than women being more likely to customize their public-facing surfaces.

Discussion and Conclusions

This chapter demonstrates the estimation of parameters related to color usage in a novel "big data" context. The technique presented here has a number of advantages and disadvantages in terms of parameter estimation as compared to more traditional techniques of control experiments. The disadvantages include the inability to isolate the subject from distracting or confounding stimuli. Furthermore, it is more difficult or impossible to determine subject characteristics such as age or occupation. However, this approach also provides a number of significant advantages. Primary is the low cost – in both time and money – of gathering very large samples. These data are in a computer-readable format that further lowers analysis costs. A study of this scale which analyzed hundreds of thousands of subjects would be virtually impossible without utilizing an existing platform such as the one provided by Twitter. Such a large sample size allows us to mitigate the effects of noisy data or subject error and to obtain very precise estimates.

It is hoped that the techniques described here can be applied to additional questions and issues related to color research. For instance, studies have looked at changes in color perception in different regions. (Saito 1998, Gao, et al. 2007) Twitter provides information on both the language used by its users and their general location. Thus, further research could look at the how color preferences change in the different regions and cultures where Twitter is used.

Furthermore, this chapter looks at one aspect of Twitter users color choice for their profile page: the page background color. Although, this color is by far the dominant color on the page, an important part of color appearance is the interaction between different colors. Although many preceding studies have followed a similar route to this one and only explored a single color in isolation (Eysenck 1941), other studies have looked at the interactions between different colors. (Camgöz, Yener and Güvenç 2002, Helson and Lansford 1970) Further research using Twitter data could explore the interaction between background, sidebar and text colors on user profile pages. In future work, more complex modeling and parameter estimation techniques such as A3 could also be employed, but the big data context increases the practical difficulties of utilizing these approaches.

References

Bridges, Judith. "Pink or Blue: Gender Stereotypic Perceptions of Infants as Conveyed Birth Congratulations Cards." *Psychology of Women Quarterly* 17, no. 2 (July 1993).

Camgöz, N, C Yener, and D Güvenç. "Effects of hue, saturation, and brightness on preference." *Color Research & Application* (Wiley Online Library) 27, no. 3 (2002): 199-207.

Child, IL, JA Hansen, and FW Hornbeck. "Age and sex differences in children's color preferences." *Child development* (JSTOR) 39, no. 1 (1968): 237-247.

Cubukcu, E, and I Kahraman. "Hue, saturation, lightness, and building exterior preference: An empirical study in Turkey comparing architects' and nonarchitects' evaluative and cognitive judgments." *Color Research and Application* (Wiley Online Library) 33, no. 5 (2008): 395-405.

Eysenck, HJ. "A critical and experimental study of colour preferences." *The American Journal of Psychology* (JSTOR) 54, no. 3 (1941): 385-394.

Gao, X.-P., et al. "Analysis of cross-cultural color emotion." *Color Research & Applications* 32, no. 3 (April 2007).

Guilford, JP, and PC Smith. "A system of color-preferences." *The American Journal of Psychology* (JSTOR) 72, no. 4 (1959): 487-502.

Helson, H, and T Lansford. "The role of spectral energy of source and background color in the pleasantness of object colors." *Applied Optics* (OSA) 9, no. 7 (1970): 1513-1562.

Levkowitz, H, and GT Herman. "GLHS: a generalized lightness, hue, and saturation color model." *CVGIP: Graphical Models and Image Processing* (Elsevier) 55, no. 4 (1993): 271-285.

Lund, Ulric, and Claudio Agostinelli. "circular: Circular Statistics." *Software.* 2010.

Manav, B. "Color-emotion associations and color preferences: A case study for residences." *Color Research & Application* (Wiley Online Library) 32, no. 2 (2007): 144-150.

Oracle. "MySQL." *Software.* September 10, 2010.

Pliner, Patricia, Shelly Chaiken, and Gordon Flett. "Gender Differences in Concern with Body Weight and Physical Appearance Over the Life Span." *Personality and Social Psychology Bulletin* 16, no. 2 (June 1990).

R Development Core Team. "R: A Language and Environment for Statistical Computing." *Software.* Vienna, 2010.

Saito, Miho. "Comparative studies on color preference in Japan and other Asian regions, with special emphasis on the preference for white." *Color Research & Applications* 21, no. 1 (December 1998).

The PHP Group. "PHP." *Software.* 2010.

Twitter Inc. *Twitter.* 2010. http://twitter.com.

Warden, Peet, Jon Orwant, and Eamon Daly. "Gender For Name PHP Class." *Software.* December 31, 2009.

6. Conclusions

The preceding dissertation chapters explored model selection and parameter estimation in disparate modeling applications. To summarize, the four key studies in this work were as follows:

1. In Chapter 2, the A3 method was developed and evaluated. The method creates a standardized model assessment table that can be used to accurately compare and select between competing models. The applied use of the method was illustrated to show not only how its application can, with the availability of suitable modeling algorithms, lead directly to improved predictions, but also how models which better fit the data may lead to fundamentally altered inferences and conclusions about parameter values and significances. An *R* package implementing this method was also developed and released onto CRAN.
2. In Chapter 3, a novel home range estimator was developed which focuses on the selection between competing home range estimators. Home range estimators define (implicitly or explicitly) models of an animal's utilization of space. These models are naturally approximations and in practice they may have heterogeneous performance when applied to differing data sets. The Chapter demonstrated that the LoCoH family of estimators outperformed Kernel Density Estimation for data sets with sharp boundaries while Kernel Density Estimation outperformed LoCoH for data sets with gradually changing densities. The ICE estimator, which selected among these competing estimators, slightly outperformed LoCoH, KDE and the MCP in average across datasets.
3. In Chapter 4, a second novel home range estimator is presented. This estimator is an extension of Kernel Density Estimation and is applicable beyond the specific application of home range estimation. The method addresses the case where data are sampled with a specific error, such as when observations are assigned to the center of the grid cell in which they are located. The method is applied to estimate the distribution of a sample of users from the social networking site Twitter. The user location data are variable, being either tagged with GPS coordinates or with coarse information such as the name of the city, state or country where the user is located. Thus, if the entire data set is desired to be utilized (rather than, for instance, just using the GPS data) the magnitude of errors between observations is quite variable and well suited for Contingent Kernel Density Estimation. This method forms an instructive contrast to the method presented in Chapter 3 in that Contingent Kernel Density Estimation is primarily assumption driven while ICE is primarily data driven.
4. In Chapter 5, I present the results of a large-scale study of data collected from the social networking site Twitter. Based on a data set which

includes over four million users, this analysis employs “big data” to obtain highly accurate inferences. This study also laid the groundwork for a longer-term collection of Twitter data, a dataset that is now over a Terabyte in size, to explore additional questions especially related to disease spread.

Policy Implications and Recommendations

At this point, I would like to step back and take the liberty of drawing broader conclusions from my modeling work as part of this dissertation. To this end, the key conclusion I have come to is the importance of exploring a diverse and competing heterogeneous ecosystem of models rather than pursuing a single monolithic approach to modeling. To understand this, let us revisit our metaphor from the conclusion of Chapter 2 comparing the space of all models to a mountain range where each point represents a model and the height at the point indicates the predictive accuracy of that model for a given modeling task.

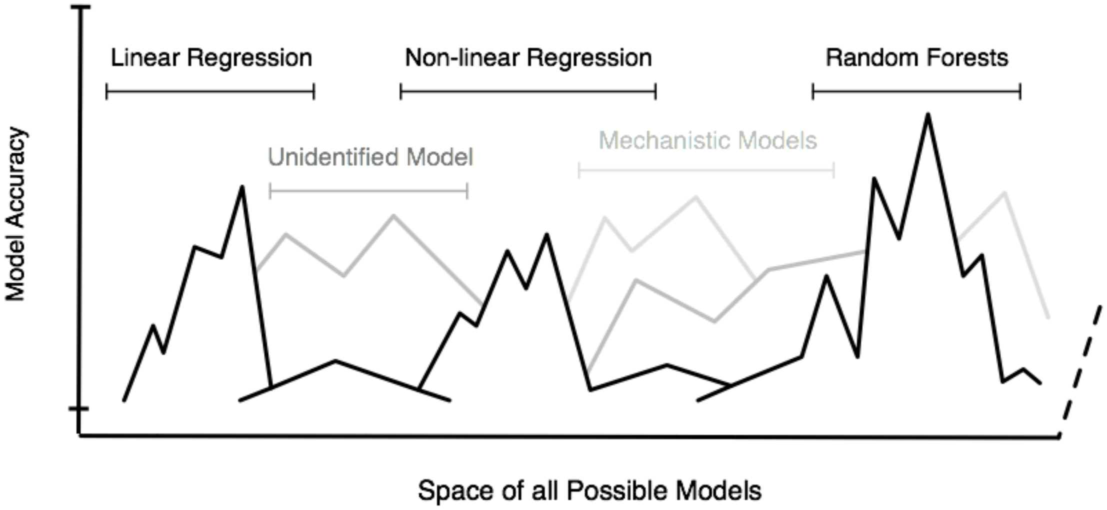


Figure 30. A mountain range as a conceptual metaphor for the space of all models.

Conceptually, each set of peaks within the range corresponds to a given family of models. We can never hope to fully explore all the peaks in the model space. It is of course infinitely large and the mathematics to explore many parts of it are still emerging or to be invented. However, it is incumbent upon us to explore more than just a single peak. If we constrain ourselves to just using a single family of models – whether it be linear regression, random forests, neural networks, or some other technique – then our view is hopelessly constrained and far less realistic than it need be. That we look beyond a monoculture approach to modeling and instead compare and contrast different techniques in a competitive manner is one of the key focuses of this work.

Towards an Agile Model Development

Exploring a multitude of techniques is a challenge. Developing even a single model can take many years and require great expense. How then can we expect to develop a range competing models within a similar time frame and budget?

My belief is that the use of an agile approach to model development is one way forward. Agile development methodologies have their roots in the software development field. They are marked by several key principles which include: rapid iteration, willingness to change and adjust design constraints on the fly, and the desire to never be far away from a runnable and functional software application. This approach to software development is in contrast to traditional approaches that may emphasize significant upfront planning in which detailed schedules and design specifications that are then implemented. This classic "waterfall" approach to development may work well for projects where the required steps are well understood and will not change, but they may be less successful when dealing with uncertainty or rapid changing conditions or designs. There are multiple reasons I favor an agile approach.

First, given the unfortunate lack of model reuse in general[20], most modeling projects will be inherently unique and novel. For example, a biologist and modeler may have developed dozens of different population models in her career, but when it comes to developing a model for a new project, she will almost certainly run into situations and problems she has not encountered before. The quantity and quality of data will differ from previous cases. Or the biology of the species to be modeled will be different. Or the model goals and constraints differ from before, and so on. Second, when building a model, a modeler may often find that many of her operating assumptions may be wrong or otherwise misleading. This can occur with every aspect of model construction: the data the modeler thought would be available will turn out to be non-existent; the equations provided to the modeler by domain experts may end up not working; and the model code the modeler write may invariably have a bug or two that needs to be fixed at the last minute, thereby changing results and conclusions. Because of this and more, the modeler continually needs to adjust and adapt her model as she learns more about the system and what information can be relied upon and what cannot.

[20] A number of scholars have explored the issue and developed recommendations to support model reuse (Overstreet, Nance, & Balci, 2002; Paul & Taylor, 2002; S. Robinson, Nance, Paul, Pidd, & Taylor, 2004). Still when tasked with building a new model, it is my experience that many modelers would choose to develop a new model rather than adapting existing works even if prior models exist that may be very close to their given modeling challenge. As Geoff McDonnell, a Simulation Research Fellow at the Australian Institute of Health Innovation, remarked to the author, "models are like children: they are all ugly except for your own" (McDonnell, 2013).

Such a high likelihood of error and need for readjustment are not well suited to techniques based on sequential project management formats. What good is a great plan if the assumptions it is based on are substantially wrong?

Consequently, I have come to the personal conclusion that most modeling projects are better off to take an agile approach to building models, that involves jumping into the model construction process as early as possible, and remaining flexible throughout the process. When beginning a modeling project my own experience has been that it is best to start building the simplest model possible that addresses the core questions driving the modeling process. We can term this simplest of models the *Minimum Viable Model*[21]. It is the model that contains just enough to minimally represent the system and address the questions driving the modeling process and nothing more. For the Minimum Viable Model, the modeler does not have to worry much about the equations being right or the model being an accurate predictor, as the goal is just to get something up and running as quickly as possible.

Once the Minimum Viable Model is built, the modeler can ask people to review it and begin to incorporate their feedback. Based on the feedback, the model should be rapidly iterated upon in an agile fashion. The modeler can make a change here or add a new component there. If the modeler receives the feedback that no one really trusts the model because it does not contain some key mechanism, then she can add that mechanism to the model[22]. The goal is to steadily adjust and refine the model based on the actual results of the model and the feedback the modeler receives. By putting a stake in the ground with a model that simulates, the modeler allows others to critique and engage with the model, providing them with valuable information about what works and what does not.

The advantage of this approach of rapid iteration is that it enables the quick identification of failures. If a data source is no good, the modeler finds that out sooner rather than later after having spent days, weeks, or months planning the model on the assumption that the data source is really available for use. Rapid iteration – failing fast and failing often – is a key goal in the agile development process and also applicable to modeling. The more approaches and variations a modeler experiments with, the more comprehensive their results and the greater their likelihood of arriving at a satisfactory solution. Of course, such

[21] This idea is adapted from Eric Ries's excellent book *The Lean Startup* (Ries, 2011). In it, he advocates an approach to developing start-up companies and businesses focus on rapid development and innovation. Ries supports developing a “Minimal Viable Product” for the company as quickly as possible and iterating on the feedback received for this initial product.

[22] But the key, in my view, is to wait until you get this feedback. It's easy on your own or with a group of people to make a list of dozens of mechanisms that a model *must* contain to be realistic. Once you have implemented those mechanisms in your model you might find out that in actuality they did not matter as much as expected. It is best to start small and then augment the model when there is a demand for some additional mechanism, than it is to spend a long time implementing a very complex model only to find out much of that work was unnecessary.

experimentation will results in a large number of failures but, by speeding up the process of identifying and iterating past failures, an agile approach to modeling may result in higher quality models completed more quickly.

Competitive Model Construction

Generating many competing models may still be an expensive and time-consuming endeavor for even the most agile of small teams. Another potential approach for achieving this end at a relatively low cost is through crowd sourcing the modeling process. For example, a number of web sites such as 99 Designs (http://99designs.com/) provide organizations the ability to crowd source projects, obtaining a wide range of designs and spec work for a relatively small investment. 99 Designs is targeted at graphic design projects and it may seem at first that such an approach would be unsuitable for the modeling field. However, Kaggle (http://www.kaggle.com/) has done exactly that: created a competitive, crowd sourced market place for modeling.

At the time of writing, Kaggle has run over 100 competitions with prizes of up to $3,000,000 USD to develop predictive models. Teams of individuals can compete for these prizes by attempting to develop the most accurate model for a given task. Once a competition is completed, the most accurate model is given to the client who funded the contest and the winning team is awarded the prize money. For example, for the cost of the ultimate prize of $500,000, the Heritage Health Foundation was able to attract over 1,000 teams to develop 25,000 competing models to predict healthcare outcomes[23]. That turns out to be a price of roughly $20 dollars per model, which is astoundingly cost efficient.

However, there are obvious challenges with public modeling contests. Foremost is the issue of data sensitivity. When publishing confidential health care information, Heritage Health Foundation had to extensively anonymize their data set prior to release, potentially obliterating important signals and handicapping modeling efforts. Nevertheless, there is no reason why such a competitive modeling effort must only be run externally to an organization. A monolithic internal modeling team could be divided into multiple sub-teams or independent individuals each competing to develop the best model. This may seem inefficient with a good deal of replicated work and dead ends, but I believe the competitive approach to model building would yield better results in the long run and over a heterogeneous mix of problems than homogenous, large scale modeling efforts.

Future Work

I believe many avenues for future research follow from this dissertation. Since the analysis developed in Chapter 5, three years of additional data has been collected from Twitter. The data set, now over a Terabyte in size, contains

[23] http://www.heritagehealthprize.com/c/hhp/leaderboard

significant disease related information that could be used to explore disease spread both spatially and within a network. In regards to A3, further work in making the method accessible for a wider audience is needed. Since A3 is dependent on having access to a library of high-quality prediction algorithms with a consistent interface, implementing the software in other languages such as, say, JavaScript would serve limited utility as these languages do not have the same extensive library that *R* possesses. However, a web application enabling the use of the software through a web interface (possibly using the relatively new Shiny server technology[24]) could have a significant impact on the method's adoption.

In terms of the two home range estimation methods, ICE is currently limited in applicability based on its current assumption of independence between and among observations. Given a model of animal movement it is possible that this assumption could be loosened by deriving new methods for assessing the fraction of observations contained in an isopleth. This would improve the flexibility of the method and make it suitable for integrating home range methods such a T-LoCoH or Brownian Bridges which make explicit usage of the correlations between observations.

Outside of the parameters of the specific dissertation chapters, this work has raised the question of what best modeling practices are. A significant amount of material exists on the technical and theoretical aspects of modeling and so does a substantial quantity of work with recommendations for the practice of modeling especially within a given modeling domain [e.g. (Andersena & Richardsona, 1997; Forrester & Senge, 1979; Meadows & Robinson, 1985)]. There are also some comparisons between different techniques such as (Morecroft & Robinson, 2005) which qualitatively compares System Dynamics to Discrete Event Simulation for fishery models. Of particular interest is the quantitative assessment of the relative success of different modeling approaches and techniques across a range of methodologies. Carrying out a study of this would, however, be difficult, as to assess different approaches you must have experts in these approaches addressing the same problem for which a quantitative measure of success is available. Furthermore, results must be replicated across multiple experts to account for variability in ability and effort for a given challenge. I would suggest that the best structure for this type of study may be some form of modeling competition where experts are given a common modeling task.

A number of questions could be investigated through such a mechanism. First is the effect of noisy data and a modeler's ability to compensate for it. For instance, a modeling task could be presented such as, "Predict Y as a function of X_1, X_2, … X_N." Modelers would randomly be assigned to different groups each with a different set of explanatory variables. There would be a common core of explanatory variables, but each group would have an additional set of one or more explanatory variables. The utility of these extra variables would vary between groups with some offering

[24] http://www.rstudio.com/shiny/

additional predictive information with others being pure noise. Exploring how the extra variables are used and effect results as a function of methodology used and the true predictive power of the variable would be illuminating.

Another further question that could be explored in a competitive framework relates to the drivers of model trust explored in the Introduction of this dissertation. For instance, “Model mirrors reality” was select by almost 14% of the surveyed population as the most important element for them to trust a model. It would be informative to provide a modeling task, such as predicting population growth over time, where experts were split into groups where some groups were given added constrains (such as that their model had to adequately simulate the empirical mechanisms, i.e. “mirror reality”), while other groups where able to use whatever modeling approach they chose whether or not it reflected reality. Comparing the results of such an analysis would be highly germane to public policy.

Recruiting experienced (and naturally very busy) modelers to take part in studies like the proposed would be quite difficult as well. However, I believe that such work would shed great light on the intersection between the theory and practice of modeling.

Appendices

Appendix A: Model Attitudes Survey

Surveys were conducted using Google Consumer Surveys, a service offered by Google Inc., which delivers surveys by requiring web surfers to answer survey questions before displaying certain content. Google Consumer Surveys does not directly ask respondents demographic information but instead infers some level of demographic information – gender, age, income, and location – from a number of signals including a respondent's IP address and the DoubleClick[25] cookie used by the respondent. Google Consumer Surveys uses stratified random sampling to more accurately sample the United States population of Internet users. Google Consumer Surveys has been shown to be competitive in accuracy both with other internet based surveys and with more traditional phone based surveys (McDonald, Mohebbi, & Slatkin, 2012; Silver, 2010).

The two surveys were conducted between April 2013 and June 2013; both with a target size of 1,500 respondents. The following sections describe the survey results.

Question 1: "Scientific models have many uses including forecasting and estimation. In general, how much do you trust scientific models?"

Respondent Demographics

There were 1,503 respondents for this question, of which demographic information was inferred for 1,208 respondents. 56.9% of respondents were reached through news sites, 26.9% of respondents were reached through arts and entertainment sites, and 15.8% of respondents were reached through reference sites. Detailed overviews of respondent demographics compared to Current Population Survey data are available below.

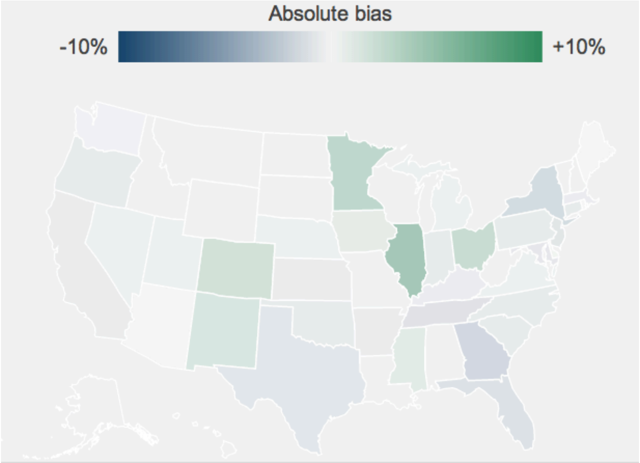
Table 15. Respondent demographics compared to average for the nation.

Group	Response	Non-response	CPS	Bias
Male	56.0%	50.4%	48.4%	7.6%
Female	44.0%	49.6%	51.6%	-7.6%
18-24	8.5%	13.2%	14.8%	-6.3%
25-34	17.8%	17.4%	20.1%	-2.3%
35-44	14.5%	15.2%	19.2%	-4.7%
45-54	22.2%	17.6%	19.7%	2.5%
55-64	23.4%	24.9%	15.4%	8.0%
65+	13.5%	11.6%	10.8%	2.8%
Midwest	30.1%	26.2%	22.2%	7.9%
Northeast	16.3%	19.5%	18.7%	-2.4%

[25] DoubleClick is an Internet advertising company owned by Google.

South	29.1%	30.7%	35.5%	-6.4%
West	24.5%	23.6%	23.6%	0.

Figure 31. Bias in respondent geographic distributions nationally.



Responses

Table 16. Overview of responses.

	All Respondents (*n*=1503)	**With Demographics (*n*=1208)**
Not at all - 1	19.9% (+2.1 / -1.9)	20.0% (+2.5 / -2.3)
2	13.3% (+1.8 / -1.6)	12.9% (+2.1 / -1.9)
3	37.5% (+2.5 / -2.4)	37.5% (+3.0 / -2.9)
4	22.6% (+2.2 / -2.0)	22.9% (+2.6 / -2.4)
I trust them completely - 5	6.8% (+1.4 / -1.2)	6.7% (+1.8 / -1.5)
Average	2.8 (+0.1 / -0.1)	2.8 (+0.1 / -0.1)

Table 17. Overview of responses by gender.

	Men (*n*=704)	**Women (*n*=552)**	**Gender unknown (*n*=247)**
Not at all - 1	20.7% (+3.1 / -2.8)	19.4% (+3.5 / -3.1)	18.6% (+5.3 / -4.4)
2	11.9% (+2.6 / -2.2)	14.7% (+3.2 / -2.7)	14.2% (+4.9 / -3.8)
3	32.4% (+3.5 / -3.4)	42.4% (+4.2 / -4.1)	40.9% (+6.2 / -5.9)
4	27.1% (+3.4 / -3.2)	18.7% (+3.5 / -3.0)	18.2% (+5.3 / -4.3)
I trust them completely - 5	7.8%	4.9%	8.1%

	(+2.2 / -1.8)	(+2.1 / -1.5)	(+4.1 / -2.8)
Average	2.9 (+0.1 / -0.1)	2.8 (+0.1 / -0.1)	2.8 (+0.1 / -0.1)

Table 18. Overview of responses by urban density.

	People in urban areas (*n*=623)	**People in rural areas (*n*=203)**	**People in suburban areas (*n*=649)**	**Urban density unknown (*n*=28)**
Not at all - 1	18.8% (+3.3 / -2.9)	18.2% (+5.9 / -4.7)	21.3% (+3.3 / -3.0)	25.0% (+18.4 / -12.3)
2	13.2% (+2.9 / -2.4)	13.8% (+5.4 / -4.1)	13.4% (+2.8 / -2.4)	10.7% (+16.5 / -7.0)
3	34.2% (+3.8 / -3.6)	45.3% (+6.9 / -6.7)	38.1% (+3.8 / -3.7)	39.3% (+18.3 / -15.7)
4	25.8% (+3.6 / -3.3)	19.2% (+6.0 / -4.8)	20.8% (+3.3 / -2.9)	14.3% (+17.2 / -8.6)
I trust them completely - 5	8.0% (+2.4 / -1.9)	3.4% (+3.5 / -1.8)	6.5% (+2.2 / -1.6)	10.7% (+16.5 / -7.0)
Average	2.9 (+0.1 / -0.1)	2.8 (+0.1 / -0.1)	2.8 (+0.1 / -0.1)	2.8 (+0.5 / -0.5)

Question 2: “To** trust **a model, what** is most important **for you? Scientific models are used to understand things like the weather or the economy.”

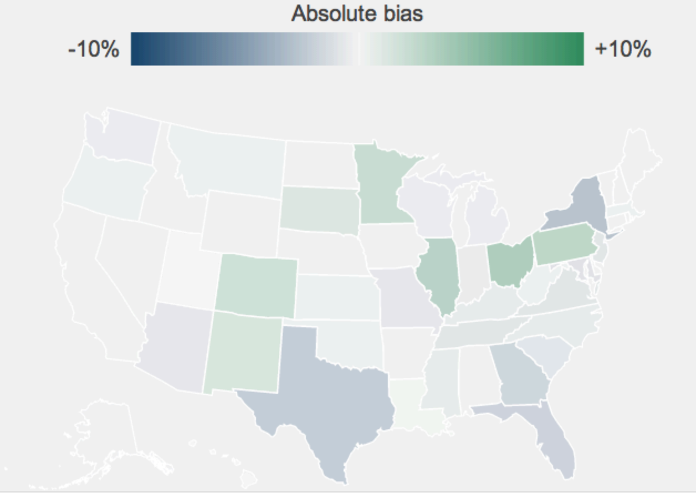
Respondent Demographics

There were 1,511 respondents for this question, of which demographic information was inferred for 1,171 respondents. 55.2% of respondents were reached through news sites, 27.9% of respondents were reached through arts and entertainment sites, and 16.4% of respondents were reached through reference sites. Detailed overviews of respondent demographics compared to Current Population Survey data are available below.

Table 19. Respondent demographics compared to average for the nation.

Group	Response	Non-response	CPS	Bias
Male	57.4%	58.1%	48.4%	9.1%
Female	42.6%	41.9%	51.6%	-9.1%
18-24	10.3%	15.9%	14.8%	-4.5%
25-34	20.4%	22.5%	20.1%	0.3%
35-44	15.1%	15.2%	19.2%	-4.1%
45-54	19.3%	17.4%	19.7%	-0.3%
55-64	21.4%	16.0%	15.4%	6.0%
65+	13.4%	12.9%	10.8%	2.6%
Midwest	30.7%	23.4%	22.2%	8.5%
Northeast	17.2%	18.5%	18.7%	-1.5%
South	25.7%	28.2%	35.5%	-9.7%
West	26.4%	29.9%	23.6%	2.8%

Figure 32. Bias in respondent geographic distributions nationally.



Responses

Table 20. Overview of responses.

	All Respondents (*n*=1511)	With Demographics (*n*=1171)
Model predicts events accurately	25.8% (+2.3 / -2.1)	26.0% (+2.8 / -2.6)
Model is widely used	24.2% (+2.2 / -2.1)	24.3% (+2.8 / -2.6)
Model developed and	20.5% (+2.1 / -2.0)	21.5% (+2.7 / -2.5)

validated by experts		
Model assumptions are understandable	14.8% (+1.9 / -1.7)	14.4% (+2.3 / -2.0)
Model design mirrors reality	14.7% (+1.9 / -1.7)	13.8% (+2.3 / -2.0)

Table 21. Overview of responses by gender.

	Men (*n*=688)	**Women (*n*=517)**	**Gender unknown (*n*=306)**
Model predicts events accurately	29.4% (+3.5 / -3.3)	22.6% (+3.8 / -3.4)	23.2% (+5.0 / -4.4)
Model is widely used	21.9% (+3.2 / -2.9)	25.1% (+3.9 / -3.5)	27.5% (+5.3 / -4.7)
Model developed and validated by experts	18.5% (+3.1 / -2.7)	24.4% (+3.9 / -3.5)	18.6% (+4.7 / -4.0)
Model design mirrors reality	14.7% (+2.8 / -2.4)	13.7% (+3.2 / -2.7)	16.3% (+4.6 / -3.7)
Model assumptions are understandable	15.6% (+2.9 / -2.5)	14.1% (+3.3 / -2.7)	14.4% (+4.4 / -3.5)

Table 22. Overview of responses by urban density.

	Urban areas (*n*=563)	**Rural areas (*n*=216)**	**Suburban areas (*n*=701)**	**Urban Density unknown (*n*=31)**
Model predicts events accurately	26.1% (+3.8 / -3.5)	24.1% (+6.1 / -5.2)	26.1% (+3.4 / -3.1)	25.8% (+17.4 / -12.1)
Model is widely used	21.5% (+3.6 / -3.2)	23.1% (+6.1 / -5.1)	26.8% (+3.4 / -3.1)	19.4% (+16.9 / -10.2)
Model developed and validated by experts	19.9% (+3.5 / -3.1)	24.1% (+6.1 / -5.2)	20.1% (+3.1 / -2.8)	16.1% (+16.5 / -9.0)
Model assumptions are understandable	14.4% (+3.1 / -2.7)	16.2% (+5.5 / -4.3)	14.7% (+2.8 / -2.4)	16.1% (+16.5 / -9.0)
Model design mirrors reality	18.1% (+3.4 / -3.0)	12.5% (+5.1 / -3.8)	12.3% (+2.6 / -2.2)	22.6% (+17.2 / -11.2)

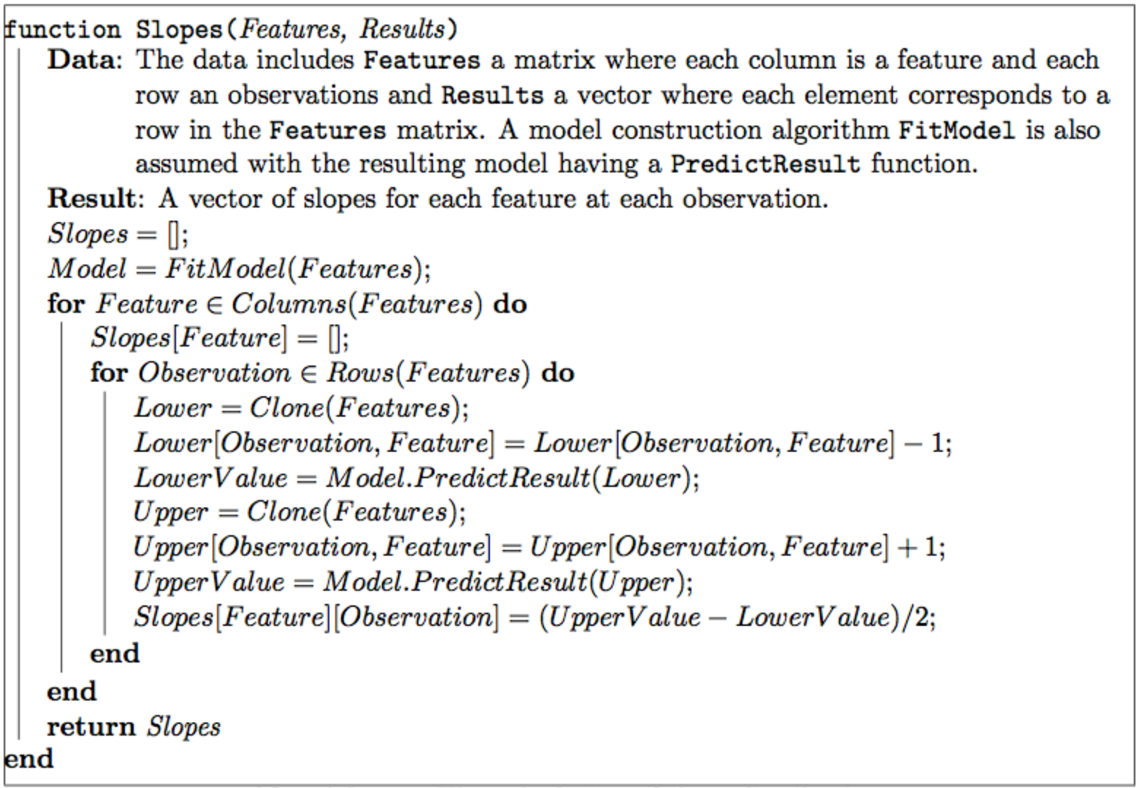
Appendix B: A3 Method Details

This appendix describes the primary algorithms used by the **A3** method. It provides brief narrative descriptions along with algorithmic outlines. For further details, the commented source code in the package itself should be referred to.

Slopes

The **A3** package calculates a measure of slope for each feature that is approximately analogous to the regression coefficient in Linear Regressions (and in fact it reduces to the regression coefficient when applied to Linear Regressions). One common way coefficients are described when discussing Linear Regressions is that they represent how much the dependent variable changes for one unit of change in the independent variable. The slope as reported by the **A3** package is calculated in directly this way and is done so for each point in the data set.

The reason that this apparently crude measure approximation of slope is used, rather than attempting to actually estimate the slope right at each point, is that many models generate non-smooth functions (CART, Random Forests, etc). For instance, in a Random Forest model, the exact slope at a point will either be 0 (if there is not a branch at that point), or infinite (if there is a branch). Thus the straightforward +/- 1 metric is used instead of attempting to precisely estimate the derivative at a point.

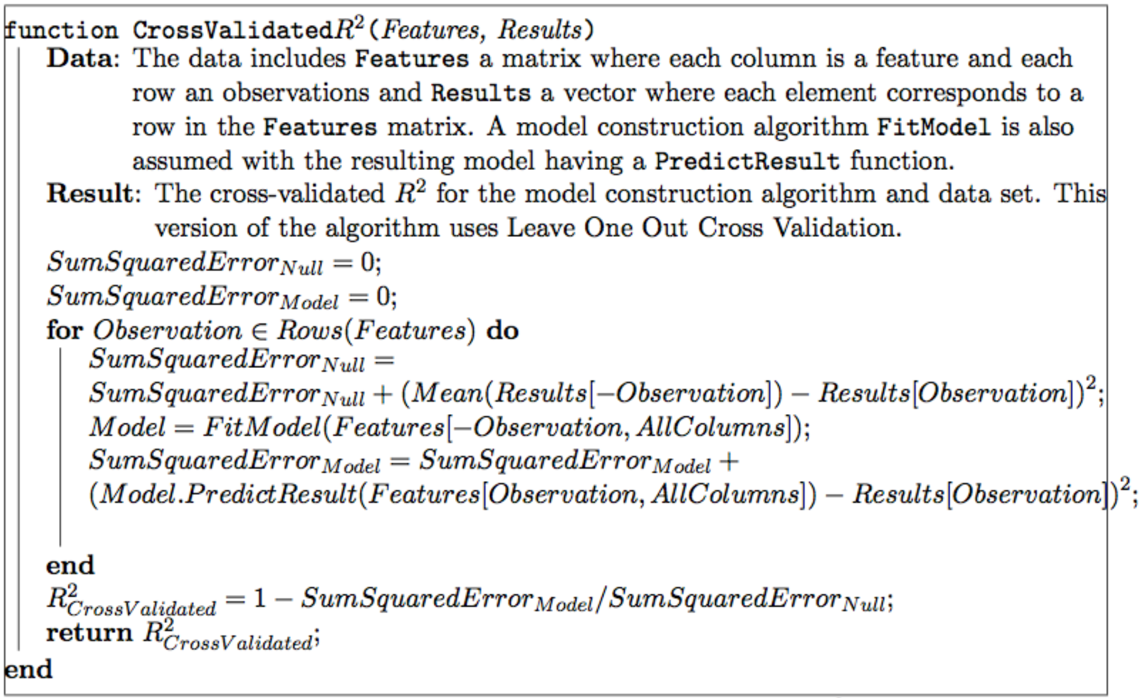


```
function Slopes(Features, Results)
    Data: The data includes Features a matrix where each column is a feature and each
          row an observations and Results a vector where each element corresponds to a
          row in the Features matrix. A model construction algorithm FitModel is also
          assumed with the resulting model having a PredictResult function.
    Result: A vector of slopes for each feature at each observation.
    $Slopes = []$;
    $Model = FitModel(Features)$;
    for $Feature \in Columns(Features)$ do
        $Slopes[Feature] = []$;
        for $Observation \in Rows(Features)$ do
            $Lower = Clone(Features)$;
            $Lower[Observation, Feature] = Lower[Observation, Feature] - 1$;
            $LowerValue = Model.PredictResult(Lower)$;
            $Upper = Clone(Features)$;
            $Upper[Observation, Feature] = Upper[Observation, Feature] + 1$;
            $UpperValue = Model.PredictResult(Upper)$;
            $Slopes[Feature][Observation] = (UpperValue - LowerValue)/2$;
        end
    end
    return $Slopes$
end
```

Algorithm 1: The calculation of slope distributions

Cross-Validated R^2

The calculation of cross-validated R^2 is straightforward. Cross-validation is a widely used technique in which a data set is divided into smaller subsets and where the error for each subset is determined using a model developed without that subset (Stone, 1974). The **A3** package supports *k*-fold cross validation. Algorithm 2 details the calculation of the cross-validated R^2 using Leave One Out Cross Validation (when the *k* in *k*-fold cross-validation is the number of observations).

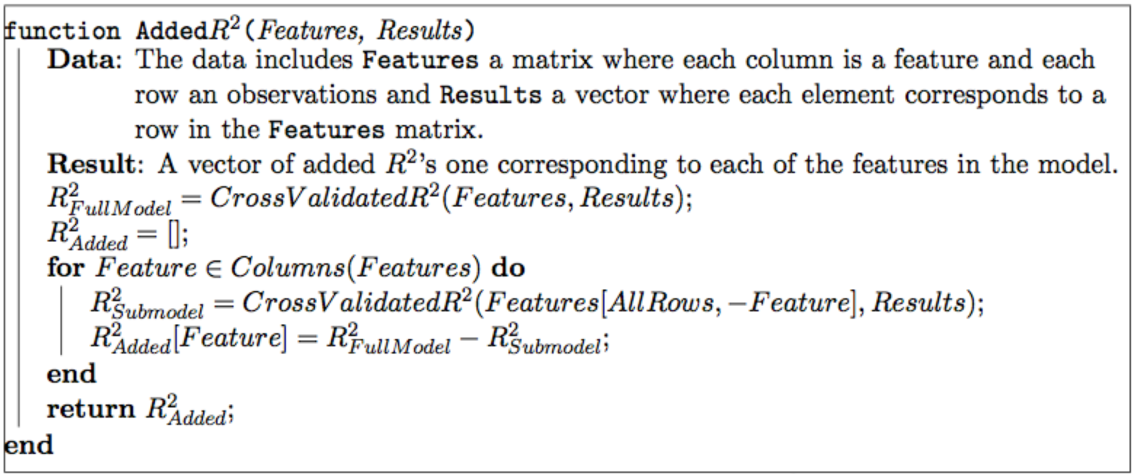


```
function CrossValidated$R^2$(Features, Results)
    Data: The data includes Features a matrix where each column is a feature and each
          row an observations and Results a vector where each element corresponds to a
          row in the Features matrix. A model construction algorithm FitModel is also
          assumed with the resulting model having a PredictResult function.
    Result: The cross-validated $R^2$ for the model construction algorithm and data set. This
            version of the algorithm uses Leave One Out Cross Validation.
    $SumSquaredError_{Null} = 0;$
    $SumSquaredError_{Model} = 0;$
    for $Observation \in Rows(Features)$ do
        $SumSquaredError_{Null} =$
        $SumSquaredError_{Null} + (Mean(Results[-Observation]) - Results[Observation])^2;$
        $Model = FitModel(Features[-Observation, AllColumns]);$
        $SumSquaredError_{Model} = SumSquaredError_{Model} +$
        $(Model.PredictResult(Features[Observation, AllColumns]) - Results[Observation])^2;$

    end
    $R^2_{CrossValidated} = 1 - SumSquaredError_{Model}/SumSquaredError_{Null};$
    return $R^2_{CrossValidated};$
end
```

Algorithm 2: Calculation of cross-validated R^2

Algorithm 3 details the calculation of the Added R^2's in the model. These indicate how much the predictive accuracy of the model is increased when a given feature is added to the model.

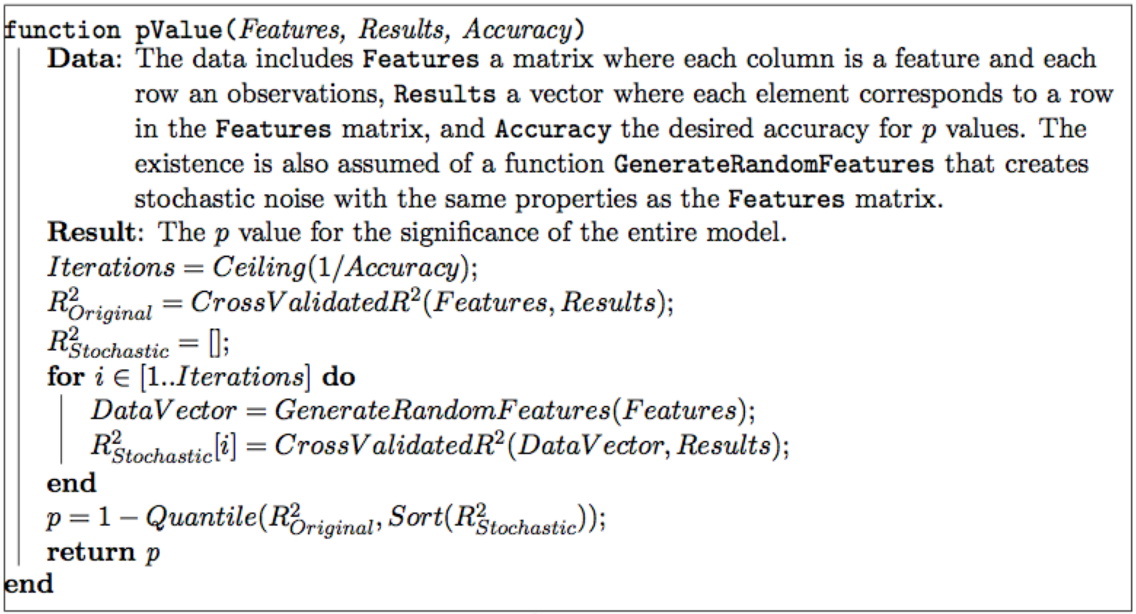


```
function Added$R^2$(Features, Results)
    Data: The data includes Features a matrix where each column is a feature and each
          row an observations and Results a vector where each element corresponds to a
          row in the Features matrix.
    Result: A vector of added $R^2$'s one corresponding to each of the features in the model.
    $R^2_{FullModel} = CrossValidatedR^2(Features, Results);$
    $R^2_{Added} = [];$
    for $Feature \in Columns(Features)$ do
        $R^2_{Submodel} = CrossValidatedR^2(Features[AllRows, -Feature], Results);$
        $R^2_{Added}[Feature] = R^2_{FullModel} - R^2_{Submodel};$
    end
    return $R^2_{Added};$
end
```

Algorithm 3: Calculation of added R^2's

p *Values*

The calculation of *p* values is done using an exact algorithm where the accuracy of the added cross-validated R^2 for a portion of the data is compared to that for randomly generated data. The basic algorithm is detailed in Algorithm 4. The same technique can also be applied to calculated the significance of individual features.



```
function pValue(Features, Results, Accuracy)
    Data: The data includes Features a matrix where each column is a feature and each
          row an observations, Results a vector where each element corresponds to a row
          in the Features matrix, and Accuracy the desired accuracy for $p$ values. The
          existence is also assumed of a function GenerateRandomFeatures that creates
          stochastic noise with the same properties as the Features matrix.
    Result: The $p$ value for the significance of the entire model.
    $Iterations = Ceiling(1/Accuracy)$;
    $R^2_{Original} = CrossValidatedR^2(Features, Results)$;
    $R^2_{Stochastic} = []$;
    for $i \in [1..Iterations]$ do
        $DataVector = GenerateRandomFeatures(Features)$;
        $R^2_{Stochastic}[i] = CrossValidatedR^2(DataVector, Results)$;
    end
    $p = 1 - Quantile(R^2_{Original}, Sort(R^2_{Stochastic}))$;
    return $p$
end
```

Algorithm 4: Calculation of p values

Appendix C: A3 Integration

The *a3* method of the **A3** package relies on a model construction algorithm. The *a3* method, in effect, "wraps" an existing algorithm and generates its statistics based on the output of the method it is wrapping. The technique it uses to carry out this wrapping works for many existing *R* model construction functions. However, in some cases the implementation of the target model construction algorithm may make it fail. In these instances, custom code may be required to bridge the **A3** package and the target model construction algorithm.

The *a3* method assumes the following properties of a model function denoted *f*:

- *f* accepts a *formula* argument that specifies a regression relationship
- *f* accepts a *data* argument that contains a data frame for the specified formula
- *f* returns a model object for which a generic *predict* method has been defined. This *predict* method's first argument should be the regression model and the second argument should be new data from which to generate predictions.

Many, built-in *R* functions conform to these three criteria along with many user contributed packages and functions. However, in cases where a function does not conform to this specification, custom code may be written to allow you to use the function with the *a3* method. As an example, the following general framework could be used:

```
R> # Defined wrapper for custom function
R> function customFunction.wrapper(formula, data){
R>  x <- createModelWithCustomFunction(...)
R>  class(x) <- "a3CustomFunction"
R>  x
R> }
R>
R> # Define predict generic method for result from
wrapper
R> function predict.a3CustomFunction(regression,
new.data){
R>  predictWithCustomFunction(...)
R> }
R>
R> # Run a3 with wrapper
R> a3(y ~ ., data, customFunction.wrapper)
```

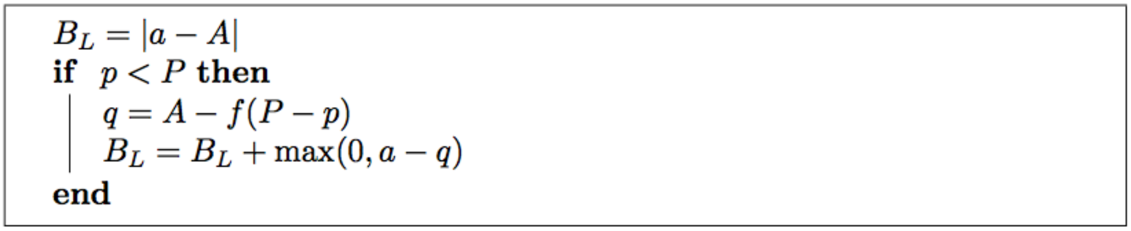
Appendix D: Using the Isopleth Curve to Assess Error

An algorithm is presented here to determine the bounds on the distance (as defined as the symmetric geometric difference between the two original regions) between an area/*p* value pair and a point on the isopleth curve. The following notation is used:

- ***p***: the fraction of the density contained by the point not on the isopleth curve
- ***a***: the area of the point not on the isopleth curve
- ***P***: the fraction of the density contained by the point on the isopleth curve (labeled α in the rest of the chapter)
- ***A***: the area of the point on the isopleth curve
- ***f(x)***: the area on the isopleth curve associated with the *x* fraction of the density (note that *f(P)=A*)

Given this notation, we may now proceed to present the algorithms to calculate the lower and upper bounds.

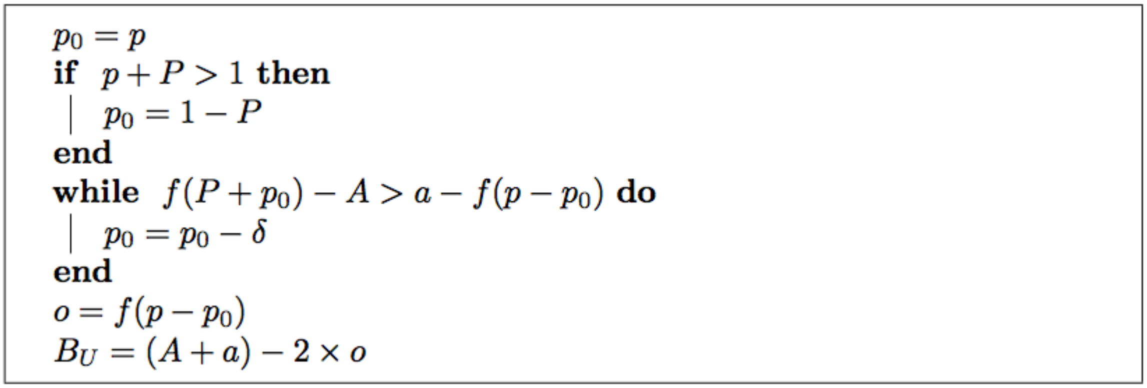
First, we start with the lower bound. Denoting the lower bound, B_L, we carry out the following algorithm to determine this lower bound:



```
$B_L = |a - A|$
if $p < P$ then
    $q = A - f(P - p)$
    $B_L = B_L + \max(0, a - q)$
end
```

The first calculation starts by setting the lower bound to the difference in areas. The logical check after this further narrows the boundary (by increasing the lower bound) by exploiting information embedded in the geometry of the isopleth curve.

Next we turn our attention to the upper bound on the distance. Denoting this upper bound, B_U, we apply the following algorithm:



```
$p_0 = p$
if $p + P > 1$ then
    $p_0 = 1 - P$
end
while $f(P + p_0) - A > a - f(p - p_0)$ do
    $p_0 = p_0 - \delta$
end
$o = f(p - p_0)$
$B_U = (A + a) - 2 \times o$
```

This second algorithm is more complicated than that for the lower bound. It works by identifying the necessary overlap between two regions based on the geometry of the isopleth curve. It then sets the upper bound to the sum of the two regions' areas minus the overlapping parts. In the algorithm, δ in the calculation represents a very small number.

Thus the true distance falls within [B_L, B_U]. Where in this range is the true distance? The answer to this question is not available based solely on the area/*p* plots. However, we can begin to answer it by asking another question: What is the general quality of our candidate estimators?

Imagine we included a candidate estimator that we knew to be a poor estimator. For instance, we could define an estimator which proposed regions that intentionally did not include any of the observations. In this case, it would be safe to assume that that the true distance fell much closer to the upper bound of the range than the lower bound. Conversely, in our case we may assume that our candidate home range estimators are actually quite good. In general the estimators you will employ either have strong statistical theory behind them, extensive practical usage and testing, or both. Given this, it is most likely safe to make the assumption that the true distance for these estimators is nearer the lower bound of the range, rather than the upper bound.

This is an optimistic assumption, but justifiable. Given it, in ICE we in fact take this to its logical conclusion (lacking further data or theory) and use solely the lower bound in estimating distance for estimator selection. The upper bound is ignored in our final selection algorithm and is presented here solely for illustrative purposes.

Appendix E: Source Code

The following subsections contain source code for the key charters in this work. This code can be used to reproduce the key findings in this work.

A3

The following code implements the A3 algorithm. It is additionally published online on CRAN (http://cran.r-project.org/web/packages/A3/index.html).

```
#' A3 Results for Arbitrary Model
#'
#' This function calculates the A3 results for an arbitrary model construction algorithm
(e.g. Linear Regressions, Support Vector Machines or Random Forests). For linear
regression models, you may use the \code{\link{a3.lm}} convenience function.
#'
#' @param formula the regression formula.
#' @param data a data frame containing the data to be used in the model fit.
#' @param model.fn the function to be used to build the model.
#' @param model.args a list of arguments passed to \code{model.fn}.
#' @param ... additional arguments passed to \code{\link{a3.base}}.
#' @return S3 \code{A3} object; see \code{\link{a3.base}} for details
#' @examples
#' \donttest{
#'  ## Standard linear regression results:
#'
#'  summary(lm(rating ~ ., attitude))
#'
#'  ## A3 Results for a Linear Regression model:
#'
#'  # In practice, p.acc should be <= 0.01 in order
#'  # to obtain finer grained p values.
#'
#'  a3(rating ~ ., attitude, lm, p.acc = 0.1)
#'
#'
#'  ## A3 Results for a Random Forest model:
#'
#'  # It is important to include the "+0" in the formula
#'  # to eliminate the constant term.
#'
#'  require(randomForest)
#'  a3(rating ~ .+0, attitude, randomForest, p.acc = 0.1)
#'
#'  # Set the ntrees argument of the randomForest function to 100
#'
#'  a3(rating ~ .+0, attitude, randomForest, p.acc = 0.1, model.args = list(ntree = 100))
#'
#'  # Speed up the calculation by doing 5-fold cross-validation.
#'  # This is faster and more conservative (i.e. it should over-estimate error)
#'
#'  a3(rating ~ .+0, attitude, randomForest, n.folds = 5, p.acc = 0.1)
#'
#'  # Use Leave One Out Cross Validation. The least biased approach,
#'  # but, for large data sets, potentially very slow.
#'
#'  a3(rating ~ .+0, attitude, randomForest, n.folds = 0, p.acc = 0.1)
#'
#'  ## Use a Support Vector Machine algorithm.
#'
#'  # Just calculate the slopes and R^2 values, do not calculate p values.
#'
#'  require(e1071)
#'  a3(rating ~ .+0, attitude, svm, p.acc = NULL)
#'  }

a3 <- function(formula, data, model.fn, model.args = list(), ...){

  model.fn.w.args <- function(y, x){
    dat <- data.frame(cbind(y, x))

    names(dat) <-  c("y", paste("x", 1:ncol(x), sep=""))

    new.model.args = list(formula = y ~ . + 0, data = dat)
    for(n in names(model.args)){
      new.model.args[[n]] = model.args[[n]]
    }

    return(do.call(model.fn, new.model.args))
  }

  simulate.fn <- function(y, x, new.x, ...){
    reg <- model.fn.w.args(y, x)
    new.data <- data.frame(new.x)
    if(ncol(new.data) != ncol(x)){
      new.data <- data.frame(t(new.data))
    }
    names(new.data) <- paste("x", 1:ncol(x), sep="")
    return(predict(reg, new.data))
  }

  a3.base(formula, data, model.fn.w.args, simulate.fn, ...)
}

#' A3 for Linear Regressions
#'
#' This convenience function calculates the A3 results specifically for linear
regressions. It uses R's \code{\link{glm}} function and so supports logistic regressions
and other link functions using the \code{family} argument. For other forms of models you
may use the more general \code{\link{a3}} function.
#'
#' @param formula the regression formula.
#' @param data a data frame containing the data to be used in the model fit.
#' @param family the regression family. Typically 'gaussian' for linear regressions.
#' @param ... additional arguments passed to \code{\link{a3.base}}.
#' @return S3 \code{A3} object; see \code{\link{a3.base}} for details
#' @examples
#' \donttest{
#'  ## Standard linear regression results:
#'
#'  summary(lm(rating ~ ., attitude))
#'
#'  ## A3 linear regression results:
#'
#'  # In practice, p.acc should be <= 0.01 in order
#'  # to obtain fine grained p values.
#'
#'  a3.lm(rating ~ ., attitude, p.acc = 0.1)
#'
#'  # This is equivalent both to:
#'
#'  a3(rating ~ ., attitude, glm, model.args = list(family = gaussian), p.acc = 0.1)
#'
#'  # and also to:
#'
#'  a3(rating ~ ., attitude, lm, p.acc = 0.1)
#'  }

a3.lm <- function(formula, data, family = gaussian, ...){
  a3(formula, data, model.fn = glm, model.args = list(family = family), ...)
}



#'
```

```
#' This function calculates the A3 results. Generally this function is not called
directly. It is simpler to use \code{\link{a3}} (for arbitrary models) or
\code{\link{a3.lm}} (specifically for linear regressions).
#'
#' @param formula the regression formula.
#' @param data a data frame containing the data to be used in the model fit.
#' @param model.fn function used to generate a model.
#' @param simulate.fn function used to create the model and generate predictions.
#' @param n.folds the number of folds used for cross-validation. Set to 0 to use Leave
One Out Cross Validation.
#' @param data.generating.fn the function used to generate stochastic noise for
calculation of exact p values.
#' @param p.acc the desired accuracy for the calculation of exact p values. The entire
calculation process will be repeated \eqn{1/p.acc} times so this can have a dramatic
affect on time required. Set to \code{NULL} to disable the calculation of p values.
#' @param features whether to calculate the average slopes, added \eqn{R^2} and p values
for each of the features in addition to the overall model.
#' @param slope.sample if not NULL the sample size for use to calculate the average
slopes (useful for very large data sets).
#' @param slope.displacement the amount of displacement to take in calculating the slopes.
May be a single number in which case the same slope is applied to all features. May also
be a named vector where there is a name for each feature.
#' @return S3 \code{A3} object containing:
#' \item{model.R2}{The cross validated \eqn{R^2} for the entire model.}
#' \item{feature.R2}{The cross validated \eqn{R^2}'s for the features (if calculated).}
#' \item{model.p}{The p value for the entire model (if calculated).}
#' \item{feature.p}{The p value for the features (if calculated).}
#' \item{all.R2}{The \eqn{R^2}'s for the model features, and any stochastic simulations
for calculating exact p values.}
#' \item{observed}{The observed response for each observation.}
#' \item{predicted}{The predicted response for each observation.}
#' \item{slopes}{Average slopes for each of the features (if calculated).}
#' \item{all.slopes}{Slopes for each of the observations for each of the features (if
calculated).}
#' \item{table}{The A3 results table.}
#'
a3.base <- function(formula, data, model.fn, simulate.fn,  n.folds = 10,
data.generating.fn = replicate(ncol(x), a3.gen.default), p.acc = 0.01, features = TRUE,
slope.sample = NULL, slope.displacement = 1){
  if(! is.null(p.acc)){
    if(p.acc <= 0 || p.acc >=1){
      stop("p.acc must be between 0 and 1. Set p.acc to NULL to disable the calculation
of p values.")
    }
  }
  if(n.folds < 2 && n.folds != 0){
    stop("n.folds must be >= 2. Set n.folds to 0 to use Leave One Out Cross Validation.")
  }

  n.reps <- 0
  if(! is.null(p.acc)){
    n.reps <- ceiling(1/p.acc)
  }

  res <- list()
  mf <- model.frame(formula, data, drop.unused.levels = TRUE)
  x <- model.matrix(formula, mf)
  y <- model.response(mf)

  if(length(data.generating.fn) != ncol(x)){
    stop("data.generating.fn must be a list of functions one for each column in the model
matrix")
  }

  if(n.folds == 0){
    n.folds <- length(y)
  }

  my.apply <- lapply
  if( ! is.null(p.acc) ){
    if( library(pbapply, quietly = TRUE, logical.return = TRUE) == TRUE ){
```

```
    my.apply <- pblapply # Show a progress bar if available
  }
}

# Calculate the groups for cross validation
cv.folds <- split(sample(1:length(y)), rep(1:n.folds, length = length(y)))

# Generate random data series for p values
new.data <- lapply(1:ncol(x), function(c){
  data.generating.fn[[c]](x[,c], n.reps)
})


r2.formater <- function(x){
  signs <- sign(x)
  x <- abs(x)*100
  res <- paste(format(round(x, 1), digits = 3), "%")
  signs <- sapply(signs, function(x){
    if(x == -1){
      return("- ")
    }else{
      return("+ ")
    }
  })
  if(signs[1] == "+ "){
    signs[1] <- "  " # removed the plus sign for the overall model accuracy
  }
  res <- paste(signs, res, sep="")
  return(res)
}

p.formater <- function(x){
  res <- format(x, digits = 4)
  for(i in 1:length(x)){
    if(x[i] == 0){
      res[i] <- paste("<", p.acc)
    }
  }
  return(res)
}

# Setup iterations
# "default" is initial simulation without any randomized data
# Each rep after that has some form of randomized data
iterations <- "default"
if(! is.null(p.acc)){
  iterations <- c(iterations, 1:n.reps)
}

top <- 0
if(features){
  top <- ncol(x)
}

# Iterate through each rep and the default
# outputs[[1]] will have the set of default cases in it
# outputs[[>1]] will have the randomized data cases
outputs <- my.apply(iterations, function(rep){
  # Calculate R2's for the rep
  # We calculate for the model (0) and then for each column of data by numerical index
  out <- lapply(0:top, function(c){

    new.x <- x
    if(rep != "default"){
      # If we aren't on the default case, we add some form of randomization
      if(c==0){
        # We are doing the overall model
        # So randomize all the data

        for(j in 1:ncol(x)){
          new.x[,j] <- new.data[[j]][[as.numeric(rep)]]
          }
        }else{
          # We're looking at a specific column, so just randomize that data
          new.x[,c] <- new.data[[c]][[as.numeric(rep)]]
        }
      }

      # Remove a column of data if we are at the un-randomized case
      if((c != 0) && (rep == "default")){
        if(top == 1){
          return(list(R2=0)); # if there is only one feature column added R^2 should be
full value
        }
        new.x <- as.data.frame(new.x[,-c])
      }

      res <- a3.r2(y, new.x, simulate.fn, cv.folds)
      return(res)

    }
    )
    r2 <- sapply(out, function(x){x$R2})

    return(list( R2 = r2, predicted = out[[1]]$predicted, observed = out[[1]]$observed ))
  })

  predicted <- outputs[[1]]$predicted
  observed <- outputs[[1]]$observed

  outputs <- lapply(outputs, function(x){x$R2})

  if(features){
    get.names <- function(formula, data){
      # <- terms(formula, data=data)
      #l <- attr(t, "term.labels")
      #if(attr(t, "intercept")==1){
      #  l <- c("(Intercept)",l)
      #}
      return(attr(x, "dimnames")[[2]])
    }
    entry.names <- c("-Full Model-", get.names(formula, data = data))

    getSlopes <- function (reg, data){
      slopes <- list()
      for(col in 2:ncol(data)){
        slopes[[as.character(col)]] <- c()
        span <- range(data[,col])
        span <- span[2] - span[1]

        for(row in 1:nrow(data)){

          point <- data[row,]
          at.point <- predict(reg, point)

          if(length(slope.displacement) == 1){
            dist <- slope.displacement
          }else{
            dist <- slope.displacement[entry.names[col]]
          }

          slope <- 0

          #while(TRUE){

            above.point <- point
            above.point[col] <- point[col] + dist
            at.above <- predict(reg, above.point)[[1]]
            below.point <- point
            below.point[col] <- point[col] - dist
            at.below <- predict(reg, below.point)[[1]]
            new.slope <- (at.above - at.below)/(dist*2)
```

```
#
#             if(new.slope == 0){
#               dist <- dist * 2
#               if(slope != 0){
#                 break
#               }
#               if(dist > span){
#                 break
#               }
#             }else{
#               if(abs( (slope-new.slope) / new.slope) < epsilon){
#                 break
#               }
#               dist <- dist / 2
#             }

            slope <- new.slope
          #}
          slopes[[as.character(col)]] <- c(slopes[[as.character(col)]], slope)
        }
      }
      slopes
    }
    # print(model.fn(y, x))
    slope.data <- data.frame(cbind(y, x))
    names(slope.data) <-  c("y", paste("x", 1:ncol(x), sep=""))
       if(! is.null(slope.sample)){
               slope.data <- slope.data[sample(1:nrow(slope.data), slope.sample),]
       }
    res[["all.slopes"]] <- getSlopes(model.fn(y, x), slope.data)
    res[["all.slopes"]] <- lapply(res[["all.slopes"]], function(x){ round(x, digits =
8)})
    res[["slopes"]] = sapply(res[["all.slopes"]], function(x){
      return(median(x))
#       r <- unique(round(range(x), digits=8))
#       if(length(r) == 1){
#         return(r)
#       }else{
#         return(paste(r, collapse = " - "))
#       }
    } )
    names(res[["all.slopes"]]) <- entry.names[-1]
    names(res[["slopes"]]) <- entry.names[-1]

  }else{
    entry.names <- c("-Full Model-")
  }

  # Now take the data and calculate R2 and p values
  res[["predicted"]] <- predicted
  res[["observed"]] <-  observed
  if(! is.null(p.acc)){
    # we did a set of repitions so we should calculate  p value
    r2 <- c()
    p.values <- c()
    res$all.R2 <- list()

    # item 1 is the overall model
    # item > 1 is a specific column
    for(i in 1:(top+1)){
      # Get the R2 for the specific item
      items <- sapply(outputs, function(x){x[i]})

      # check if we are on a column item
      if(i > 1){
        # if so replace the first element of the items list with the value of the overall
model R2
        items[1] <- r2[1]
      }

      names(items) <- c("Base",paste("Rep", 1:n.reps))
      res$all.R2[[paste(entry.names[i])]] <- items

      # rank the items by R2
      dist <- rank(items)

      # find the position of the first item (overall model R2) within the list of
randomly derived p values
      # this is the emprical R2
      p.values <- c(p.values, 1 - (dist[1]-1)/(length(dist)-1))

      # the R2 of the model as generated with random data
      new.null <- mean(items[-1])

      if(i==1){
        # the R2 of the model as generated with random data
        new.null <- mean(items[-1])

        # if the R2 with stochasticity is better than 0, we will use as a baseline to
scale our R2
        #if(new.null > 0){ # Adjust R^2 based on what was observed in stochastic series
XXX reenable?
        #  r2 <- (items[1]-new.null) / (1-new.null)
        #}else{
          r2 <- items[1]
        #}
      }else{
        # get data series again (we overwrote the item[1] position earlier)
        items <- sapply(outputs, function(x){x[i]})

        # see how the results improve compared to the baseline
        r2 <- c(r2, r2[1] - items[1])#max(new.null, items[1])) # Adjust R^2 based on what
was observed in stochastic series XXX reenable?
      }
    }

    res$table <- data.frame(`Average Slope` = c("", res$slopes), `CV R^2` =
r2.formater(r2), `p value` = p.formater(p.values), check.names=F)
    res$model.R2 <- r2[1]
    res$feature.R2 <- r2[-1]
    res$model.p <- p.values[1]
    names(res$model.p) <- entry.names[1]
    res$feature.p <- p.values[-1]
    names(res$feature.p) <- entry.names[-1]
  }else{
    # we didn't do repetitions so no p values

    r2 <- outputs[[1]]
    r2[-1] <- r2[1] - r2[-1] # get delta to overall model R2

    res$table <- data.frame(`Average Slope` = c("", res$slopes), `CV R^2` =
r2.formater(r2), check.names=F)
    res$all.R2 <- r2
    res$model.R2 <- r2[1]
    res$feature.R2 <- r2[-1]
  }

  names(res$model.R2) <- entry.names[1]
  names(res$feature.R2) <- entry.names[-1]

  row.names(res$table) <- entry.names

  class(res) <- "A3"

  return(res)
}

#' Plot A3 Results
#'
#' Plots an 'A3' object results. Displays predicted versus observed values for each
observation along with the distribution of slopes measured for each feature.
#'
```

```
#' @param x an A3 object.
#' @param ... additional options provided to \code{\link{plotPredictions}},
\code{\link{plotSlopes}} and \code{\link{plot}} functions.
#' @method plot A3
#' @examples
#' \donttest{
#'  data(housing)
#'  res <- a3.lm(MED.VALUE ~ NOX + ROOMS + AGE + HIGHWAY + PUPIL.TEACHER, housing, p.acc
= NULL)
#'  plot(res)
#'  }

plot.A3 <- function(x, ...){
  if(class(x) != "A3"){
    stop("'x' must be of class 'A3'.")
  }

  plotPredictions(x, ...)
  old.par <- par(ask=T)
  plotSlopes(x, ...)
  par(old.par)
}

#' Plot Predicted versus Observed
#'
#' Plots an 'A3' object's values showing the predicted versus observed values for each
observation.
#'
#' @param x an A3 object,
#' @param show.equality if true plot a line at 45-degrees.
#' @param xlab the x-axis label.
#' @param ylab the y-axis label.
#' @param main the plot title.
#' @param ... additional options provided to the \code{\link{plot}} function.
#'
#' @examples
#'  data(multifunctionality)
#'  x <- a3.lm(MUL ~ ., multifunctionality, p.acc = NULL, features = FALSE)
#'  plotPredictions(x)

plotPredictions <- function(x, show.equality = TRUE, xlab = "Observed Value", ylab =
"Predicted Value", main = "Predicted vs Observed", ...){
  if(class(x) != "A3"){
    stop("'x' must be of class 'A3'.")
  }

  plot(x$observed, x$predict, xlab=xlab, ylab=ylab, main=main, ...)
  abline(h=0, col="Gray"); abline(v=0, col="Gray");
  if(show.equality){
    abline(coef = c(0, 1), col = "Blue", lty = 2)
  }
}

#' Plot Distribution of Slopes
#'
#' Plots an 'A3' object's distribution of slopes for each feature and observation. Uses
Kernel Density Estimation to create an estimate of the distribution of slopes for a
feature.
#'
#' @param x an A3 object.
#' @param ... additional options provided to the \code{\link{plot}} and
\code{\link{density}} functions.
#'
#' @examples
#' \donttest{
#'  require(randomForest)
#'  data(housing)
#'  x <- a3(MED.VALUE ~ NOX + PUPIL.TEACHER + ROOMS + AGE + HIGHWAY + 0, housing,
randomForest, p.acc = NULL, n.folds = 2)
#'  plotSlopes(x)
#'  }
```

```
plotSlopes <- function(x,  ...){
  if(class(x) != "A3"){
    stop("'x' must be of class 'A3'.")
  }
  size <- length(x$slopes)
  if(size == 0 ){
    stop("no slopes to plot")
  }
  width <- ceiling(sqrt(size))
  height <- floor(sqrt(size))
  if(width*height < size){
    width <- width+1
  }

  old.par <- par(mfrow = c(height, width), mar = .55*c(5, 4, 4, 2) + 0.2)

  for(s in names(x$slopes)){
    if(length(unique(x$all.slopes[[s]]))==1){
      plot(x$slopes[[s]], 0, main = s)
    }else{
      plot(density(x$all.slopes[[s]],...), xlab = "", ylab="", main = s, ...)
      rug(x$all.slopes[[s]], col="Blue")
    }
    abline(h=0, col="Gray")
    abline(v=0, col="Gray")
  }

  par(old.par)
}

#' Print Fit Results
#'
#' Prints an 'A3' object results table.
#'
#' @param x an A3 object.
#' @param ... additional arguments passed to the \code{\link{print}} function.
#' @method print A3
#' @examples
#'  x <- a3.lm(rating ~ ., attitude, p.acc = NULL)
#'  print(x)

print.A3 <- function(x, ...){
  if(class(x) != "A3"){
    stop("'x' must be of class 'A3'.")
  }

  print(x$table, ...)
}

#' Nicely Formatted Fit Results
#'
#' Creates a LaTeX table of results. Depends on the \pkg{xtable} package.
#'
#' @param x an A3 object.
#' @param ... additional arguments passed to the \code{\link{print.xtable}} function.
#' @method xtable A3
#' @examples
#'  x <- a3.lm(rating ~ ., attitude, p.acc = NULL)
#'  xtable(x)

xtable.A3 <- function(x, ...){
  require(xtable)
  if(class(x) != "A3"){
    stop("'x' must be of class 'A3'.")
  }

  data <- x$table
  names(data) <- gsub("p value","Pr(>R^2)", names(data), fixed=TRUE)
  names(data) <- gsub("R^2","$R^2$", names(data), fixed=TRUE)
  print(xtable(data, align=c("l","|", rep("r", ncol(data))), ...),
sanitize.colnames.function = function(x){x}, sanitize.text.function = function(x){
    return(sapply(x, function(x){
      trimmed = gsub("(^ +)|( +$)", "", x)
      if((! is.na(suppressWarnings(as.numeric(x)))) &&
suppressWarnings(as.numeric(x))==trimmed){
        return(paste0("$", x, "$"))
      }else if(substr(trimmed, nchar(trimmed), nchar(trimmed))=="%"){
        return(paste0("$", gsub("%", "\\%", trimmed, fixed=T), "$"));
      }else if(substr(trimmed, 1, 1)=="<"){
        return(paste0("$", trimmed, "$"));
      }else{
        return (gsub("_", "\\_", x, fixed=T));
      }
    }))
  })
}


#' Cross-Validated \eqn{R^2}
#'
#' Applies cross validation to obtain the cross-validated \eqn{R^2} for a model: the
fraction of the squared error explained by the model compared to the null model (which is
defined as the average response). A pseudo \eqn{R^2} is implemented for classification.
#'
#' @param y a vector or responses.
#' @param x a matrix of features.
#' @param simulate.fn a function object that creates a model and predicts y.
#' @param cv.folds the cross-validation folds.
#'
#' @return A list comprising of the following elements:
#' \item{R2}{the cross-validated \eqn{R^2}}
#' \item{predicted}{the predicted responses}
#' \item{observed}{the observed responses}

a3.r2 <- function(y, x, simulate.fn, cv.folds){

  errors <- lapply(cv.folds, function(fold){
    test.y <- y[fold]
    test.x <- x[fold,]
    train.y <- y[-fold]
    train.x <- as.data.frame(x[-fold,])
    new.y <- simulate.fn(train.y, train.x, test.x)

    if(is.factor(new.y)){
      # classification
      return(list( type="classification", correct = (new.y == test.y), predicted = new.y,
observed = test.y ))
    }else{
      # regression
      y.null <- mean(train.y)
      return(list( type="regression", ss.null = sum((test.y-y.null)^2), ss.model =
sum((test.y-new.y)^2), predicted = new.y, observed = test.y ))
    }
  }
  )

  if(errors[[1]]$type == "regression"){
    # regression
    ss.model <- sum(unlist(sapply(errors, function(x){x$ss.model})))
    ss.null <- sum(unlist(sapply(errors, function(x){x$ss.null})))
    return( list(R2 = 1 - ss.model/ss.null, predicted = unlist(sapply(errors,
function(x){x$predicted})), observed = unlist(sapply(errors, function(x){x$observed}))) )
  }else{
    # classification
    corrects <- unlist(sapply(errors, function(x){x$correct}))

    null.count <- max(table(as.factor(y))) # return number of observations in the largest
class
```

```
    return( list( R2 = (sum(corrects)-null.count) / (length(corrects)-null.count),
predicted = unlist(sapply(errors, function(x){x$predicted})), observed =
unlist(sapply(errors, function(x){x$observed}))) )
  }
}

#' Stochastic Data Generators
#'
#' The stochastic data generators generate stochastic noise with (if specified correctly)
the same properties as the observed data. By replicating the stochastic properties of the
original data, we are able to obtain the exact calculation of p values.
#'
#' Generally these will not be called directly but will instead be passed to the
\code{data.generating.fn} argument of \code{\link{a3.base}}.
#'
#' @name a3.gen.default
#' @aliases a3.gen.default a3.gen.bootstrap a3.gen.resample a3.gen.normal a3.gen.autocor
#'
#' @param x the original (observed) data series.
#' @param n.reps the number of stochastic repetitions to generate.
#'
#' @return A list of of length \code{n.reps} of vectors of stochastic noise. There are a
number of different methods of generating noise:
#' \item{a3.gen.default}{The default data generator. Uses \code{a3.gen.bootstrap}.}
#' \item{a3.gen.resample}{Reorders the original data series.}
#' \item{a3.gen.bootstrap}{Resamples the original data series with replacement.}
#' \item{a3.gen.normal}{Calculates the mean and standard deviation of the original series
and generates a new series with that distribution.}
#' \item{a3.gen.autocor}{Assumesa first order autocorrelation of the original series and
generates a new series with the same properties.}
#'
#' @examples
#' \donttest{
#'  # Calculate the A3 results assuming an auto-correlated set of observations.
#'  # In usage p.acc should be <=0.01 in order to obtain more accurate p values.
#'
#'  a3.lm(rating ~ ., attitude, p.acc = 0.1, data.generating.fn =
replicate(ncol(attitude), a3.gen.autocor))
#'  }
#'
#'  ## A general illustration:
#'
#'  # Take x as a sample set of observations for a feature
#'  x <- c(0.349, 1.845, 2.287, 1.921, 0.803, 0.855, 2.368, 3.023, 2.102, 4.648)
#'
#'  # Generate three stochastic data series with the same autocorrelation properties as x
#'  rand.x <- a3.gen.autocor(x, 3)
#'
#'  plot(x, type="l")
#'  for(i in 1:3) lines(rand.x[[i]], lwd = 0.2)

# Default generator, use bootstrap
a3.gen.default <- function(x, n.reps){
  if(length(unique(x))==1){
    #it's a constant, such as an intercept
    return(a3.gen.normal(x, n.reps))
  }
  a3.gen.bootstrap(x, n.reps)
}

# Generates a bootstrap random data series
a3.gen.bootstrap <- function(x, n.reps){
  res <- lapply(1:n.reps, function(r) {sample(x, length(x), replace=TRUE)})
  res$default <- x
  res
}

# Generates a resampled random data series
a3.gen.resample <- function(x, n.reps){
  res <- lapply(1:n.reps, function(r) {sample(x, length(x), replace=FALSE)})
  res$default <- x
  res
}

# Generates a normally distributed random data series
a3.gen.normal <- function(x, n.reps){
  mu <- mean(x)
  sd <- sd(x)
  if(sd == 0){
    sd <- 1
  }
  res <- lapply(1:n.reps, function(r) {rnorm(length(x), mu, sd)})
  res$default <- x
  res
}

# Generates a first order autocorrelated random data series
a3.gen.autocor <- function(x, n.reps){
  mu <- mean(x)
  sd <- sd(x)
  if(sd == 0){
    r <- 1
  }else{
    r <- cor(x[-1], x[-length(x)])
  }
  res <- lapply(1:n.reps,
                function(rep) {
                  dat <- rnorm(length(x), mu, sd)
                  for(i in 2:length(x)){
                    dat[i] <- dat[i-1]*r + dat[i]*(1-r)
                  }
                  return(dat)
                }
  )
  res$default <- x
  res
}
```

ICE

The following implements the ICE method. Code was implemented both in R and, for performance reasons, C++.

R Code

```
if(F){
        load(output.file("bridge.rdata")); b <- add.best.method(bridge);
full.boxplot.test(b); full.method.selection(b)
        load(output.file("starburst.rdata")); s <- add.best.method(starburst);
full.boxplot.test(s); full.method.selection(s)

        xy <- simulate.homerange("starburst", 100)
        nearest.neighbor.cache <- get.nearest.neighbors(xy, 100)
        nearest.neighbor.cache$ids = nearest.neighbor.cache$ids-1;
        save(xy, nearest.neighbor.cache, file="~/Dropbox/hr sample.rData")

        xy <- simulate.homerange("starburst", 100)
        a.thumb(xy)
        res <- jackknife.a.locoh(xy, seq(1, 50, by=5), isopleth = 0.5)
        hr <- LoCoH(xy, a=14, isopleths = c(1, .9, .5))
        plot(hr)

        z <- test(n=10, reps=1, isopleths=.9, seed=1)
        z <- test(n=50, reps=100, isopleths=0.9, seed=1)
        z <- test(n=c(25,50), reps=100, isopleths=c(0.5, 0.8, 0.90, 0.95, 0.99), seed=1)
        z <- test(n=c(100, 200), reps = 30 , isopleths=c(0.5, 0.8, 0.9), seed=1, dists=2,
methods= c("a.locoh.pure", "a.locoh.thumb"))
        plot.test(z, "all", c("a.locoh.pure","a.locoh.thumb"))


        # bridge <- test(n=c(50, 200), reps=30, isopleths=0.8, dists="bridge", methods =
c("kernel.href", "kernel.lscv", "kernel.pure","a.locoh.pure", "a.locoh.thumb"), seed=1)
        # gaussian.1 <- test(n=c(50, 200), reps=30, isopleths=c(0.5, 0.9), dists=1,
methods = c("kernel.href", "kernel.lscv", "kernel.pure", "a.locoh.pure", "a.locoh.thumb"),
seed=1)

        # Paper

        gaussian.1 <- test(n=c(50, 200), reps=40, isopleths=c(0.5, 0.8, 0.9), dists=1,
methods = c("mcp", "kernel.href", "kernel.lscv", "a.locoh.pure", "a.locoh.thumb"),
seed=1)
        gaussian.3 <- test(n=c(50, 200), reps=40, isopleths=c(0.5, 0.8, 0.9), dists=3,
methods = c("mcp", "kernel.href", "kernel.lscv", "a.locoh.pure", "a.locoh.thumb"),
seed=1)

        bridge <- test(n=c(50, 200), reps=40, isopleths=c(0.7, 0.8, 0.9),
err.isopleths=0.8, dists="bridge", methods = c("mcp", "kernel.href", "kernel.lscv",
"a.locoh.pure", "a.locoh.thumb"), seed=1)
        starburst <- test(n=c(50, 200), reps=40, isopleths=c(0.7, 0.8, 0.9),
err.isopleths=0.8, dists="starburst", methods = c("mcp", "kernel.href", "kernel.lscv",
"a.locoh.pure", "a.locoh.thumb"), seed=1)


        walk.20 <- test(n= 200, reps=40, isopleths=c(0.5, 0.8, 0.9), dists="walk", methods
= c("mcp", "kernel.href", "kernel.lscv", "a.locoh.pure", "a.locoh.thumb"), seed=1)
        walk.5 <- test(n= 200, reps=40, isopleths=c(0.5, 0.8, 0.9), dists="walk", methods
= c("mcp", "kernel.href", "kernel.lscv", "a.locoh.pure", "a.locoh.thumb"), seed=1)
        walk.1 <- test(n= 200, reps=40, isopleths=c(0.5, 0.8, 0.9), dists="walk", methods
= c("mcp", "kernel.href", "kernel.lscv", "a.locoh.pure", "a.locoh.thumb"), seed=1)

        if(F){
                bridge <- test(n=c(25, 50, 100, 200), err.n=c(50, 200), reps=40,
isopleths=c(0.7, 0.8, 0.9, 0.99), err.isopleths=0.8, dists="bridge", methods = c("mcp",
"kernel.href", "kernel.lscv", "a.locoh.pure", "a.locoh.thumb"), seed=1)
                save(bridge, file=output.file("bridge.rdata"))

                gaussian.1 <- test(n=c(25, 50, 100, 200), err.n=c(50, 200), reps=40,
isopleths=c(0.45, 0.5, 0.55, 0.75, 0.8, 0.85, 0.9, 0.95), err.isopleths=c(0.5, 0.8, 0.9),
dists=1, methods = c("mcp", "kernel.href", "kernel.lscv", "a.locoh.pure",
"a.locoh.thumb"), seed=1)
                save(gaussian.1, file=output.file("gaussian.1.rdata"))


                starburst <- test(n=c(25, 50, 100, 200), err.n=c(50, 200), reps=40,
isopleths=c(0.7, 0.8, 0.9, 0.99), err.isopleths=0.8, dists="starburst", methods = c("mcp",
"kernel.href", "kernel.lscv", "a.locoh.pure", "a.locoh.thumb"), seed=1)
                save(starburst, file=output.file("starburst.rdata"))

                gaussian.3 <- test(n=c(25, 50, 100, 200), err.n=c(50, 200), reps=40,
isopleths=c(0.45, 0.5, 0.55, 0.75, 0.8, 0.85, 0.9, 0.95), err.isopleths=c(0.5, 0.8, 0.9),
dists=3, methods = c("mcp", "kernel.href", "kernel.lscv", "a.locoh.pure",
"a.locoh.thumb"), seed=1)
                save(gaussian.3, file=output.file("gaussian.3.rdata"))
        }


        # ---

        load(output.file("bridge.large.rdata"))
        load(output.file("bridge.rdata"))

        bridge <- add.best.method(bridge)
        bridge.large <- add.best.method(bridge.large)

        b <- full.summarize.test(bridge, kinds="geo.err")
        b.l <- full.summarize.test(bridge.large, kinds="geo.err")
        b.res <- rbind(b[1:2,], b.l[1:2,])

        row.names(b.res) <- sapply(row.names(b.res), function(x){ strsplit(x,"
")[[1]][1]})
        matplot(b.res)

        # ---

        load(output.file("gaussian.1.large.rdata"))
        load(output.file("gaussian.1.rdata"))

        gaussian.1 <- add.best.method(gaussian.1)
        gaussian.1.large <- add.best.method(gaussian.1.large)

        g <- full.summarize.test(gaussian.1, kinds="geo.err")
        g.l <- full.summarize.test(gaussian.1.large, kinds="geo.err")

        g.res.50 <- rbind(g[c(1,4),], g.l[c(3,3),])[-4,]
        g.res.90 <- rbind(g[c(3,6),], g.l[c(4,4),])[-4,]

        row.names(g.res.50) <- sapply(row.names(g.res.50), function(x){ strsplit(x,"
")[[1]][1]})
        matplot(g.res.50)

        row.names(g.res.90) <- sapply(row.names(g.res.90), function(x){ strsplit(x,"
")[[1]][1]})
        matplot(g.res.90)

        # ----

        load(output.file("kernel.pure.gaussian.1.rdata"))
        gaussian.1 <- add.best.method(gaussian.1)
        full.summarize.test(gaussian.1, kinds="geo.err")

        load(output.file("kernel.pure.test 2.rdata"))
        bridge <-  add.best.method(bridge)
        full.summarize.test(bridge, kinds="geo.err")
}
# install.packages(c("adehabitatHR", "sp","MSBVAR" ,"rgeos", "ggplot2",
"gpclib","raster","spatstat","binom","randomForest","e1071", "A3", "Rcpp", "knitr"),
dependencies = TRUE)

library(adehabitatHR)
library(sp)
library(MSBVAR)
library(rgeos)
library(ggplot2)
library(compiler)
library(gpclib)
library(raster)
library(spatstat)
library(binom)
library(gpclib)
require(Rcpp)
sourceCpp("/alocoh.cpp")

spatstat.options(gpclib=TRUE)

enableJIT(3)

output.file <- function(name){
        return(paste("~/Dropbox/Homerange/output/",name,sep=""))
}

a.thumb <- function(points){
        max(nndist(points, k=nrow(points)-1))
}
```

```
get.hrs <- function(points, isopleth = 0.5, methods = c("mcp", "kernel.href",
"kernel.lscv", "kernel.pure", "a.locoh.pure", "a.locoh.thumb"), truth = NULL ){
        res <- list(points = points, isopleth = isopleth, methods = methods, truth =
truth)

        p <- SpatialPoints(points)

        if("mcp" %in% methods){
                print("Generating MCP:")

                res[["mcp"]] <- list()

                hr.mcp <- mcp(p, percent=isopleth*100)

                hr.mcp.rast <- rasterize.mcp(hr.mcp)

                res[["mcp"]][["hr"]] <- hr.mcp
                res[["mcp"]][["isopleth"]] <- hr.mcp.rast
                if(! is.null(truth)){
                        res[["mcp"]][["geo.err"]] <- grid.error(res[["mcp"]][["isopleth"]],
truth)
                }
                res[["mcp"]][["area"]] <- grid.area(res[["mcp"]][["isopleth"]])

                contained <- 0
                for(i in 1:nrow(points)){
                        hr.mcp <- mcp(p[-i,], percent=isopleth*100)
                        contained <- contained + (! is.na(over(p[i,],hr.mcp))[1])
                }
                res[["mcp"]][["p.est"]] <- contained/nrow(points)
        }

        if("kernel.href" %in% methods){
                print("Generating Kernel h-Ref:")

                h.href <- 1 * 1/2 * ( sd(points[,1]) + sd(points[,2]) )*nrow(points)^(-
1/6) # 0.96 multiplier in my Silvermann book, but discussion with Clement indicates 1 is
the multiplier
                hr.href <- make.hr.grid(p, h = h.href)
                res[["kernel.href"]] <- list()
                res[["kernel.href"]][["h"]] <- h.href
                res[["kernel.href"]][["hr"]] <- hr.href
                res[["kernel.href"]][["isopleth"]] <- binarize(hr.href, isopleth)
                if(! is.null(truth)){
                        res[["kernel.href"]][["geo.err"]] <-
grid.error(res[["kernel.href"]][["isopleth"]], truth)
                }
                res[["kernel.href"]][["area"]] <-
grid.area(res[["kernel.href"]][["isopleth"]])

                contained <- 0
                for(i in 1:nrow(points)){
                        iso <- binarize(make.hr.grid(p[-i,], h="href"), isopleth)
                        coord <- make.coord(unlist(points[i,]))
                        if(coord[1]>0 && coord[1]<=dim(iso)[1] && coord[2]>0 &&
coord[2]<=dim(iso)[2]){
                                contained <- contained + iso[coord[1], coord[2]] # it's ok
if its outside, means it not contained
                        }
                }
                res[["kernel.href"]][["p.est"]] <- contained/nrow(points)
        }



        if("kernel.lscv" %in% methods){
                print("Generating Kernel LSCV:")

                hr.lscv <- make.hr.grid(p, h="LSCV")
                h.lscv <- attr(hr.lscv, "h")
                h.grid <- attr(hr.lscv,"h.grid")
                res[["kernel.lscv"]] <- list()
                res[["kernel.lscv"]][["errs"]] <- attr(hr.lscv,"h.errors")
                res[["kernel.lscv"]][["hs"]] <- h.grid
                res[["kernel.lscv"]][["h"]] <- h.lscv
                res[["kernel.lscv"]][["hr"]] <- hr.lscv
                res[["kernel.lscv"]][["isopleth"]] <- binarize(hr.lscv, isopleth)
                if(! is.null(truth)){
                        res[["kernel.lscv"]][["geo.err"]] <- 
grid.error(res[["kernel.lscv"]][["isopleth"]], truth)
                }
                res[["kernel.lscv"]][["area"]] <- 
grid.area(res[["kernel.lscv"]][["isopleth"]])

                contained <- 0
                for(i in 1:nrow(points)){
                        iso <- binarize(make.hr.grid(p[-i,], h="LSCV"), isopleth)
                        coord <- make.coord(unlist(points[i,]))
                        if(coord[1]>0 && coord[1]<=dim(iso)[1] && coord[2]>0 && 
coord[2]<=dim(iso)[2]){
                                contained <- contained + iso[coord[1], coord[2]] # it's ok 
if its outside, means it not contained
                        }
                }
                res[["kernel.lscv"]][["p.est"]] <- contained/nrow(points)
        }




        if("a.locoh.thumb" %in% methods){
                print("Generating a-LoCoH Thumb:")

                res[["a.locoh.thumb"]] <- list()

                res[["a.locoh.thumb"]][["a"]] <- a.thumb(points) #max distance between any 
two points
                res[["a.locoh.thumb"]][["hr"]] <- LoCoH.binary(points, a = 
res[["a.locoh.thumb"]][["a"]], isopleth = isopleth)
                res[["a.locoh.thumb"]][["isopleth"]] <- res[["a.locoh.thumb"]][["hr"]]

                if(! is.null(truth)){
                        res[["a.locoh.thumb"]][["geo.err"]] <- 
grid.error(res[["a.locoh.thumb"]][["isopleth"]], truth)
                }
                res[["a.locoh.thumb"]][["area"]] <- 
grid.area(res[["a.locoh.thumb"]][["isopleth"]])

                errs <- jackknife.a.locoh(points, res[["a.locoh.thumb"]][["a"]], isopleth)

                res[["a.locoh.thumb"]][["p.est"]] <- errs$p
        }

        if("a.locoh.pure" %in% methods){
                print("Generating a-LoCoH Pure:")

                res[["a.locoh.pure"]] <- list()

                a.bounds <- LoCoH.a.bounds(points)
               res[["a.locoh.pure"]][["as"]]  <-  unique(sort(c(seq(a.bounds[1],a.bounds[2], 
length.out = 100), a.thumb(points))))


                errs <- select.a(points, res[["a.locoh.pure"]][["as"]], isopleth = 
isopleth)

                res[["a.locoh.pure"]][["errs"]] <- errs

                res[["a.locoh.pure"]][["a"]] <- errs$a
                res[["a.locoh.pure"]][["p.est"]] <- errs$p
                res[["a.locoh.pure"]][["iso"]] <- errs$isopleth

                #print(res[["a.locoh.pure"]][["a"]])

                res[["a.locoh.pure"]][["hr"]] <- LoCoH.binary(points, a = 
res[["a.locoh.pure"]][["a"]], isopleth = res[["a.locoh.pure"]][["iso"]])
                res[["a.locoh.pure"]][["isopleth"]] <- res[["a.locoh.pure"]][["hr"]]
                if(! is.null(truth)){
                        res[["a.locoh.pure"]][["geo.err"]] <- 
grid.error(res[["a.locoh.pure"]][["isopleth"]], truth)
                }
                res[["a.locoh.pure"]][["area"]] <- 
grid.area(res[["a.locoh.pure"]][["isopleth"]])
        }


        return(res)
}


make.coord <- function(pt){
        round(c((pt[1]-standard.xs[1])/standard.step, 201-(pt[2]-
standard.ys[1])/standard.step))
}


rasterize.mcp <- function(mcp){
        poly <- attr(mcp,"polygons")[[1]]@Polygons[[1]]@coords
        d <- (standard.xs[2]-standard.xs[1])/standard.step
        hr.mcp.iso <- rep(0, d^2)
        for(r in 1:d){
                for(c in 1:d){
                        if(point.in.hull(c*standard.step+standard.xs[1], 
r*standard.step+standard.ys[1], poly)){
                                hr.mcp.iso[c+(d-r)*d] <- 1
                        }
                }
        }
        return(matrix(hr.mcp.iso, d, d ))
}



# z <- test(n=100, reps=10, isopleths=.9, seed=1, methods = c("a.locoh.pure", 
"a.locoh.thumb"), dists = 1)
# z <- test(n=10, reps=1, isopleths=.9, seed=1, methods = c("kernel.href", "kernel.lscv", 
"a.locoh.pure", "a.locoh.thumb"))
# z <- test(n=100, reps=10, isopleths=.9, seed=1, methods = c("kernel.href", 
"kernel.lscv", "a.locoh.pure", "a.locoh.thumb"))
# z <- test(n=c(10,20), reps=3, isopleths=c(.9,.7), seed=1, methods = c("kernel.href", 
"a.locoh.thumb"))

# obj <- z2[[1]][[1]][["0.5"]][["a.locoh.pure"]][["errs"]]; plot(obj$data$p, 
obj$data$Area); points(obj$p, obj$data$Area[which(obj$data$a==obj$a & obj$data$isopleth 
==  obj$isopleth)], col="red", cex=1.5)
test <- function(n = c(25), err.n = n, reps = 20, isopleths = c(0.5, 0.8, 0.9, 0.95), 
err.isopleths = isopleths, methods = c("mcp", "kernel.href", "kernel.lscv", "kernel.pure", 
"a.locoh.pure", "a.locoh.thumb"), seed=1, dists=1){

        set.seed(seed)

        res <- list()
        for(i in 1:length(n)){
                res[[i]] <- list()
        }
        for(q in 1:reps){
                print(paste("Repetition:", q))
                seeder <- round(runif(1)*10000)
                print(paste("Seeder:", seeder))

                true.dist <- true.grid(dists = dists, seed = seeder)
                p.large <- generate.pts(10000, dists=dists, seed=seeder)

                p <- c()

                for(n.i in 1:length(n)){
                        print(paste("Sample size:", n[n.i]))

                        r <- list()
                        if(n.i == 1){
                                p <- generate.pts(n[n.i], dists=dists, seed=seeder)
                        }else{
                               p <- rbind(p, generate.pts(n[n.i] - n[n.i - 1], dists=dists, 
seed=seeder))
                        }


                        for(iso in isopleths){

                                print(paste("Isopleth:",iso))

                                if(iso %in% err.isopleths){
                                        b.true.dist <- binarize(true.dist, isopleth=iso)
                                }else{
                                        b.true.dist = NULL
                                }


                                r[[as.character(iso)]] <- get.hrs(p, isopleth = iso, 
methods = methods, truth = b.true.dist)



                                for(m in methods){
                                        #print(m)
                                        hr <- r[[as.character(iso)]][[m]][["isopleth"]]
                                        contained <- unlist(sapply(1:nrow(p.large), 
function(i){
                                                coord <- make.coord(unlist(p.large[i,]))
                                                if(coord[1]>0 && coord[1]<=dim(hr)[1] && 
coord[2]>0 && coord[2]<=dim(hr)[2]){
                                                        return(hr[coord[1], coord[2]])
                                                }else{
                                                        return(0) # it's ok if its outside, 
means it is not contained
                                                }
                                        }))
                                        #print(contained)
                                        prob <- sum( contained )/nrow(p.large)
                                        #print(prob)
                                        r[[as.character(iso)]][[m]][["p.true"]] <- prob
                                }

                        }
                        res[[n.i]][[q]] <- r
                }
        }
        attr(res,"isopleths") <- isopleths
        attr(res, "err.isopleths") <- err.isopleths
        attr(res, "err.n") <- err.n
        attr(res,"n") <- n
        attr(res,"reps") <- reps
        attr(res,"seed") <- seed
        attr(res,"dists") <- dists
        attr(res,"methods") <- methods
        return(res)
}

# attr(gaussian.3, "methods")
#  d <- extract.curve(gaussian.3)
```

```
extract.curve <- function(z, n=attr(z, "n"), reps = 1:attr(z, "reps"), p="p.est",
plot=TRUE){

        print("- Extracting Curve")

        methods <- attr(z, "methods")
        isopleths <- attr(z, "isopleth")

        res <- list()

        for(ns in n){
                print(ns)
                n.i <- which(attr(z, "n")==ns)
                print(n.i)

                res.isopleths <- c()
                res.ps <- c()
                res.areas <- c()
                res.methods <- c()
                res.names <- c()
                for(r in reps){
                        print(r)
                        for(iso in isopleths){
                                print(iso)
                                for(method in methods){
                                        print(method)
                                        res.isopleths <- c(res.isopleths, iso)
                                        res.methods <- c( res.methods, method)
                                        res.names <- c( res.names, method.name(method))

        print(z[[n.i]][[r]][[as.character(iso)]][[method]][[p]])
                                        res.ps <- c(res.ps,
z[[n.i]][[r]][[as.character(iso)]][[method]][[p]])
                                        res.areas <- c(res.areas,
z[[n.i]][[r]][[as.character(iso)]][[method]][["area"]])
                                }
                        }
                }
                print(res.isopleths)
                print(res.methods)
                print(res.ps)
                print(res.areas)
                print(res.names)
                d <- data.frame(Isopleth = as.factor(res.isopleths), Method =
as.factor(res.methods), p = res.ps, Area = res.areas, Estimator = as.factor(res.names))


                res[[as.character(ns)]] <- d

                if(plot){
                        g <- ggplot(d, aes(x=p, y=Area,
color=Estimator))+geom_point(aes(shape=Isopleth))
                        g <- g + labs(title= paste0("n=",ns))
                        g <- g + geom_vline(aes(xintercept= isopleths), colour="#999999",
linetype="dashed", data=data.frame(isopleths= isopleths))
                        g <- g + scale_x_continuous(limits = c(0, 1))
                        #g <- g + theme(axis.text.y = element_blank())
                        g <- g + geom_line(aes(x=p, y=Area), colour="#999999",
data=find.envelope(res, ns))
                        print(g)
                }

        }
        attr(res, "n") <- n
        return(res)
}

find.envelope <- function(data, n){
        d <- data[[as.character(n)]]

        #print("--")
        #print(n)
        #print(d)


        if(!(F %in% (d$p==0))){
                print("\n-----")
                print(data)
                print("---")
                print(n)
                print(d)
                print(d$p==0)
                print("+++")
                stop("Only 0's found for envelope probability.")
        }

        return(basic.envelope(d))

        basic <- basic.envelope(d)

        for(i in 2:(nrow(basic)-1)){
                j <- which(d$Area == basic[i,"Area"] & d$p == basic[i, "p"])
                new.env <- basic.envelope(d[-j,])
                area <- find.areas(new.env, basic[i,"p"])
                basic[i,"Area"] <- basic[i,"Area"]-(area-basic[i,"Area"])
        }

        basic.envelope(basic)
}

regress.envelope <- function(n.small, env.small, n.large, env.large, isopleth){

        scales <- c()
        b<- min(isopleth, max(env.small$p))
        for(iso in seq(b*4/5, b, length.out=10)){
                small.area <- find.areas(env.small, iso)
                large.area <- find.areas(env.large, iso)
                scales <- c(scales, regress.area(c(n.small, n.large), c(small.area,
large.area))/large.area)
        }
        env.large$Area <- env.large$Area*mean(scales)
        return(env.large)

        ps <- unique(sort(c(env.small$p, env.large$p)))
        #print(ps)

        end.pt <- (env.small$Area[nrow(env.small)]-env.small$Area[nrow(env.small)-
1])/(env.small$p[nrow(env.small)]-env.small$p[nrow(env.small)-1])*(1-
env.small$p[nrow(env.small)]) + env.small$Area[nrow(env.small)]
        small <- approx(c(env.small$p, 1), c(env.small$Area, end.pt), ps)$y
        end.pt <- (env.large$Area[nrow(env.large)]-env.large$Area[nrow(env.large)-
1])/(env.large$p[nrow(env.large)]-env.large$p[nrow(env.large)-1])*(1-
env.large$p[nrow(env.large)]) + env.large$Area[nrow(env.large)]
        large <- approx(c(env.large$p, 1), c(env.large$Area, end.pt), ps)$y
        #print(small)
        #print(large)

        areas <- c()
        for(i in 1:length(ps)){
                areas <- c(areas, regress.area(c(n.small, n.large), c(small[i],
large[i])))
        }

        pts <- chull(ps, areas)

        start <- which(ps[pts]==0)
        #for(i in start:)

        return(data.frame(p=ps, Area=areas))
        #ids <- chull(ps, areas)
```

```
}

illustrate.envelope <- function(z, rep, iso, n=1){
        obj <- z[[n]][[rep]][[as.character(iso)]][["a.locoh.pure"]][["errs"]];
        plot(obj$data$p, obj$data$Area, main=rep);
        points(obj$p, obj$area, col="red", cex=1.5);
        thumb.ind <-
which(obj$data$a==z[[n]][[rep]][[as.character(iso)]][["a.locoh.thumb"]][["a"]] &
obj$data$isopleth==iso);
        points(obj$data$p[thumb.ind], obj$data$Area[thumb.ind], col="green", cex=1.5);
        abline(v=iso);
        lines(obj$envelope)
        #obj
}
basic.envelope <- function(data){
        min.areas <- c()
        min.ps <- c()

        ind  <- rev(chull(c(0,data$p), c(0,data$Area)))
        d <- rbind(rep(0,ncol(data)), data)[ind,]

        start.i <- which(ind==1)
        iterations <- start.i:nrow(d)
        if(start.i > 1){
                iterations <- c(iterations, 1:(start.i-1))
        }

        for(i in iterations){
                if(length(min.ps)==0 || d$p[i] > max(min.ps)){
                        min.ps <- c(min.ps, d$p[i])
                        min.areas <- c(min.areas, d$Area[i])
                }else if(d$p[i] == max(min.ps) && d$Area[i] <
min.areas[length(min.areas)]){
                        min.areas[length(min.areas)] <- d$Area[i]
                }
        }

        data.frame(p=min.ps, Area=min.areas)
}

find.areas <- function(envelope, ps){
        areas <- c()
        #print("XXX")
        #print(envelope)
        #print(ps)
        for(p in ps){
                found <- FALSE
                for(i in 1:(nrow(envelope)-1)){
                        if(envelope[i,"p"]<=p & envelope[i+1,"p"]>=p){
                                areas <- c(areas, (p-envelope[i,"p"])/(envelope[i+1,"p"]-
envelope[i,"p"])*(envelope[i+1,"Area"]-envelope[i,"Area"])+envelope[i,"Area"])
                                found <- TRUE
                                break
                        }
                }
                if(found==FALSE){
                        i <- nrow(envelope) - 1
                        areas <- c(areas, (p-envelope[i+1,"p"])/(envelope[i+1,"p"]-
envelope[i,"p"])*(envelope[i+1,"Area"]-envelope[i,"Area"])+envelope[i+1,"Area"])
                }
        }
        areas
}

regress.area <- function(n, area){
        #data <- data.frame(n = 1/n, area = area)
        #print(data)
        #reg <- lm(area ~ n)

        #summary(reg)$coefficients
        slope <- (area[1]-area[2])/(1/n[1]-1/n[2])
        #print(slope)

        tarea <- area[2]-(1/n[2])*slope

        return(max(min(tarea, area[2]), min(area)/2))
}
if(! exists("dist.plots.data")){
        dist.plots.data <- c()
}
illustrate.distances <- function(isopleth = 0.8, res = 150){
        envelope.p <- seq(0, 1, by=0.02)
        envelope <- data.frame(p = envelope.p, Area = envelope.p ^2*100)

        p.grid <- seq(0, 1, length.out=res)
        area.grid <- seq(0, max(envelope$Area), length.out=res)

        ps <- c()
        areas <- c()
        for(p in p.grid){
                area <- find.areas(envelope, p)
                for(a in area.grid){
                        if(a>=area){
                                ps <- c(ps, p)
                                areas <- c(areas, a)
                        }
                }
        }
        #ps <- rep(p.grid, length(area.grid))
        #areas <- rep(area.grid, each=length(p.grid))


        d <- data.frame(p=ps, Area=areas)

        dists <- distances(d, envelope, isopleth)

        d$maxs <- dists$maxs
        d$mins <-dists$mins


        min.val <- min(c(d$maxs, d$mins))
        max.val <- max(c(d$maxs, d$mins))
        range.val <- diff(range(c(d$maxs, d$mins)))

        max.mat <- matrix(nrow=length(p.grid), ncol=length(area.grid))
        min.mat <- matrix(nrow=length(p.grid), ncol=length(area.grid))

        for(p in p.grid){
                for(a in area.grid){
                        x <- d[d$p==p & d$Area==a,]
                        #print(x);
                        if(length(x$maxs)==1){

                        #print(x)
                        #print(length(x))
                        #print(length(x$maxs))
                                max.mat[which(p.grid==p), which(area.grid==a)] <- x$maxs
                                min.mat[which(p.grid==p), which(area.grid==a)] <- x$mins
                        }
                }
        }


        dist.plots.data <<- list(isopleth = isopleth, envelope = envelope, d = d, max.mat
= max.mat, min.mat = min.mat, min.val = min.val, range.val = range.val, max.val = max.val,
p.grid = p.grid, area.grid = area.grid)

        make.distances.plots(dist.plots.data)
}

make.distances.plots <- function(data){
        d <- data$d


        FUN <- function(x){rgb(colorRamp(c("green","blue","red"))(x), maxColorValue=256)}

        #plot(d$p, d$Area, pch=16, col= FUN((d$mins-data$min.val)/data$range.val),
main="Min Error", cex=1.5)
        #lines(data$envelope)
        #abline(v = data$isopleth)

        #plot(d$p, d$Area, pch=16,col= FUN((d$maxs-data$min.val)/data$range.val),
main="Max Error", cex=1.5)
        #lines(data$envelope)
        #abline(v = data$isopleth)

        #plot(d$p, d$Area, pch=16,col= FUN(((d$maxs+d$mins)/2-
data$min.val)/data$range.val), main="Average Error", cex=1.5)
        #lines(data$envelope)
        #abline(v = data$isopleth)

        opar <-  par(mfrow=c(1,3), mar=c(5,1.1,2.2,1), oma=c(0,1.5,0,0))
        image(data$p.grid, data$area.grid, data$min.mat, main="Minimum",
col=topo.colors(50), xlab="", ylab="", yaxt='n')
        #contour(data$p.grid, data$area.grid, data$min.mat, add=T)
        lines(data$envelope, lwd = 4)
        abline(v = data$isopleth, lwd=3, lty=2, col ="darkgrey")

        mtext("Distance (Error)",2, .9, cex=.8)

        image(data$p.grid, data$area.grid, data$max.mat, main="Maximum",
col=topo.colors(50), xlab="", ylab="", yaxt='n')
        #contour(data$p.grid, data$area.grid, data$max.mat, add=T)
        lines(data$envelope, lwd = 4)
        abline(v = data$isopleth, lwd=3, lty=2, col ="darkgrey")


        mtext("Fraction of Density (p)",1, 3, cex=.8)

        image(data$p.grid, data$area.grid, (data$max.mat+data$min.mat)/2,
main="(Maximum+Minimum)/2", col=topo.colors(50), xlab="", ylab="", yaxt='n')
        #contour(data$p.grid, data$area.grid, (data$max.mat+data$min.mat)/2, add=T)
        lines(data$envelope, lwd = 4)
        abline(v = data$isopleth, lwd=3, lty=2, col ="darkgrey")


        par(opar)

}

distances <- function(d, envelope, isopleth){
        area <- find.areas(envelope, isopleth)

        mins <- c()
        maxs <- c()
        for(i in 1:nrow(d)){
                a <-  d[i, "Area"]
                p <- d[i,"p"]

                #min.val <-  max(a - area, 0)  # above the target area
                #
                #
                #if(p < isopleth){
                #       # now we do below the target area
                #
                #       min.val <- min.val + max(area-a, 0)
                #
                #       max.region <- area - find.areas(envelope, isopleth-p)
                #
                #       min.val <- min.val+ max(0, a-max.region)
                #}
                min.val = abs(a - area)
                if(p < isopleth){
                        max.region = area - find.areas(envelope, isopleth - p)
                        min.val = min.val + max(0, a - max.region)
                }

                mins <-  c(mins, min.val)


                p.o <- p
                if(p.o + isopleth > 1){
                        p.o <- 1 - isopleth
                }

                while( round(find.areas(envelope,  isopleth + p.o) - area, digits=6) >
round(a - find.areas(envelope, p - p.o), digits=6)){
                        if(p.o == 0){
                                out.data <<- data
                                print(d[i,])
                                print(p.o)
                                stop("Error with distances")
                        }
                        p.o <- max(0, p.o - 0.005)
                }
                overlap <- find.areas(envelope, p - p.o)
                maxs <- c(maxs, (area + a) - 2 * overlap)
        }
        list("mins" = mins, "maxs" = maxs)
}

out.data <- c()

select.a <- function(xy, a, isopleth){
        data <- list()
        data[[as.character(nrow(xy))]] <- rbind(jackknife.a.locoh(xy, a, isopleth =
max(0,isopleth/2)), jackknife.a.locoh(xy, a, isopleth = max(0,isopleth-0.1)),
jackknife.a.locoh(xy, a, isopleth = isopleth), jackknife.a.locoh(xy, a, isopleth =
min(.99,isopleth+.1)), jackknife.a.locoh(xy, a, isopleth = 1))

        nxy <- xy[sample(1:nrow(xy), floor(nrow(xy)/2)),]
        data[[as.character(nrow(nxy))]] <- rbind(jackknife.a.locoh(xy, a, isopleth =
max(0,isopleth/2)), jackknife.a.locoh(nxy, a, isopleth = max(0,isopleth-0.1)),
jackknife.a.locoh(nxy, a, isopleth = isopleth), jackknife.a.locoh(nxy, a, isopleth =
min(.99,isopleth+0.1)), jackknife.a.locoh(nxy, a, isopleth = 1))
        envelope.2 <- find.envelope(data, nrow(nxy))

        #nxy <- xy[sample(1:nrow(xy), floor(nrow(xy)/2)),]
        #data[[as.character(nrow(nxy))]] <- rbind(jackknife.a.locoh(nxy, a, isopleth =
max(0,isopleth-0.1)), jackknife.a.locoh(nxy, a, isopleth = isopleth),
jackknife.a.locoh(nxy, a, isopleth = min(.99,isopleth+0.1)), jackknife.a.locoh(nxy, a,
isopleth = 1))
        #area2.2 <- find.areas(find.envelope(data, nrow(nxy)), isopleth)

        #nxy <- xy[sample(1:nrow(xy), floor(nrow(xy)/2)),]
        #data[[as.character(nrow(nxy))]] <- rbind(jackknife.a.locoh(nxy, a, isopleth =
max(0,isopleth-0.1)), jackknife.a.locoh(nxy, a, isopleth = isopleth),
jackknife.a.locoh(nxy, a, isopleth = min(.99,isopleth+0.1)), jackknife.a.locoh(nxy, a,
isopleth = 1))
        #area2.3 <- find.areas(find.envelope(data, nrow(nxy)), isopleth)

        #area2 <- (area2.1+area2.2+area2.3)/3

        envelope <- find.envelope(data, nrow(xy))

        envelope <- regress.envelope(nrow(nxy), envelope.2, nrow(xy), envelope,  isopleth)

        #print("ENVELOPE")
        #print(envelope)
        #plot(envelope)

        #print(paste("Full area:", area1))
        #print(paste("Reduced area:", area2))

        area  <- find.areas(envelope, isopleth)

        d <- data[[as.character(nrow(xy))]]
        #print(d)

        dists <- distances(d, envelope,  isopleth)
        i <- which.min(dists$mins)# + dists$maxs)

        list(a = d[i, "a"], p = d[i, "p"], isopleth = d[i, "isopleth"], area = d[i,
"Area"], dists = dists, data = d, scale = scale, envelope = envelope, half.data =
data[[as.character(nrow(nxy))]])

}

# load(output.file("bridge.rdata")); z <- add.best.method(bridge); full.boxplot.test(z)
# z <- add.best.method(starburst)
# full.boxplot.test(z)
add.best.method <- function(z){
        empirical <- 0
        if(is.null(attr(z, "reps"))){
                empirical <- 1
                q <- z
                z <- q
                attr(z[[2]],"sets") <- attr(z, "sets")
                attr(z[[2]],"methods") <- attr(z, "methods")
                attr(z[[2]],"isopleths") <- attr(z, "isopleths")
                attr(z,"reps") <- length(attr(z, "sets"))
                attr(z, "isopleths") <- attr(z, "isopleths")
                attr(z, "err.isopleths") <- attr(z, "isopleths")
                attr(z, "methods") <- attr(z, "methods")
                attr(z, "n") <- c(1, 2)
                attr(z, "err.n") <- c(2)
        }

        ns <- attr(z, "n")
        err.n <- attr(z, "err.n")

        isopleths <- attr(z, "err.isopleths")
        reps <- attr(z, "reps")
        methods <- attr(z, "methods")
        if("pure" %in% methods){
                methods <- methods[-length(methods)]
        }

        #print("------")
        #print(err.n)
        #print(isopleths)
        #print(reps)
        for(n in err.n){
                print(paste("n", n))
                q <- which(ns==n)

                for(r in 1:reps){
                        if(empirical){
                                print(paste("Set:", attr(z,"sets")[r]))
                        }
                        data <- extract.curve(z,  reps = r, p = "p.est", plot = FALSE)
                        #print("data:")
                        #print(data)
                        d <- data[[as.character(n)]]

                        for(isopleth in isopleths){

                                print(paste("isopleth", isopleth))

                                small.n <- ns[which(ns==n)-1]
                                envelope <- regress.envelope(small.n, find.envelope(data,
small.n), n, find.envelope(data, n), isopleth)
```

```
                                pure <- list(method="none", geo.err = NA, area = NA, p.est
= NA, p.true = NA)
                                dists <- distances(d, envelope,  isopleth)
                                i <- which.min(dists$mins)# + dists$maxs)

                                #print(methods)
                                #print(dists)

                                min.method <- as.character(d[i, "Method"])
                                min.iso <- as.character(d[i, "Isopleth"])
                                pure[["method"]] <- min.method
                                pure[["iso"]] <- min.iso
                                print(min.method)
                                print(min.iso)
                                if(! empirical){
                                        if(min.iso == as.character(isopleth)){
                                                # we have the error stored
                                                pure[["geo.err"]] <-
z[[q]][[r]][[min.iso]][[min.method]][["geo.err"]]
                                        }else{
                                                # we need to calculate the error
                                                print("Calculating Error")
                                                pure[["geo.err"]] <-
grid.error(z[[q]][[r]][[min.iso]][[min.method]][["isopleth"]],
z[[q]][[r]][[as.character(isopleth)]][["truth"]])
                                        }
                                }
                                pure[["area"]] <-
z[[q]][[r]][[min.iso]][[min.method]][["area"]]
                                pure[["p.est"]] <-
z[[q]][[r]][[min.iso]][[min.method]][["p.est"]]
                                pure[["p.true"]] <-
z[[q]][[r]][[min.iso]][[min.method]][["p.true"]]
                                pure[["isopleth"]] <-
z[[q]][[r]][[min.iso]][[min.method]][["isopleth"]]

                                z[[q]][[r]][[as.character(isopleth)]][["pure"]] <- pure
                        }

                }
        }

        if(empirical){
                if(! ("pure" %in% attr(z[[2]], "methods"))){
                        attr(z[[2]], "methods") <- c(attr(z, "methods"), "pure")
                }
                for(i in 1:reps){
                        if(! ("pure" %in% attr(z[[1]][[i]], "methods"))){
                                attr(z[[2]][[i]], "methods") <- c(attr(z, "methods"),
"pure")
                        }
                }
                return(z[[2]])
        }else{

                if(! ("pure" %in% attr(z, "methods"))){
                        attr(z, "methods") <- c(attr(z, "methods"), "pure")
                }
                return(z)
        }

}


# plot.test(res.100, "all", c("href.err","pure.err","lscv.err"))

plot.test <- function(z,  mode = c("all","diff"), kind = "geo.err", q = 1){

        iso <- c()
        error <- c()
        type <- c()
        for(i in attr(z, "isopleths")){

                print(paste("Isopleth:",i))
                pure.err <- c()
                lscv.err <- c()

                for(r in 1:attr(z, "reps")){
                                if(mode[1]=="all"){
                                        iso <- c(iso, rep(i,length(kinds)))
                                        type <- c(type, kinds)

                                        for(k in attr(z, "methods")){
                                                error <- c(error,
z[[q]][[r]][[as.character(i)]][[k]][[kind]])
                                        }
                                }else{
                                        iso <- c(iso, i)
                                        type <- c(type, "Improvement")
                                        error <-c(error,
z[[q]][[r]][[as.character(i)]][["true.err.kernel.pure"]]-
z[[q]][[r]][[as.character(i)]][["true.err.lscv"]])
                                }

                                #pure.err <- c(pure.err,
z[[q]][[r]][[as.character(i)]][["true.err.kernel.pure"]])
                                #lscv.err <- c(lscv.err,
z[[q]][[r]][[as.character(i)]][["true.err.lscv"]])
                        }


                #print(t.test(pure.err, lscv.err, pair=TRUE, conf.level=.99))
        }

        data <- data.frame(Isopleth = iso, Error = error, Type=type)


        print(ggplot(data, aes(x=Isopleth, y=Error, color=Type))+stat_summary(fun.data =
"mean_cl_normal", geom="errorbar", position = position_dodge(.01),
lty=1)+stat_summary(fun.y = mean, geom="line", position = position_dodge(.01)))

        return(data)
}

join.test <- function(z1, z2){
        z <- z1
        base.r <- attr(z1, "reps")
        for(r in 1:attr(z2, "reps")){
                z[[r+base.r]] <- z2[[r]]
        }
        attr(z, "reps") <-base.r+attr(z2, "reps")
        return(z)

}

make.cov <- function(s,rho){
        return(matrix(c(s[1]^2, rho*s[1]*s[2], rho*s[1]*s[2], s[2]^2),2,2))
}

dmultnorm <- function(x,y,loc, s, rho)  {
        mu1 <- loc[1]
        mu2 <- loc[2]

        nlizer <- 1/(2*pi*prod(s)*sqrt(1-rho^2))
        e.term1 <- (x - mu1)/s[1]
        e.term2 <- (y - mu2)/s[2]
        like <- exp( -1/(2*(1-rho^2)) * (e.term1^2 + e.term2^2 - 2*rho*e.term1*e.term2) )
        nlizer*like
}
```

```
standard.step <- 0.05
standard.xs <- c(-5,5)
standard.ys <- c(-5,5)

# z<-make.hr.grid(generate.pts(100,2,3), "LSCV")
# z<-make.hr.grid(generate.pts(100,2,3), 1)


make.hr.grid <- function(pts, h){

        fakeda <- data.frame(X = standard.xs+c(standard.step/2,-standard.step/2), Y =
standard.ys+c(standard.step/2,-standard.step/2))
        coordinates(fakeda) <- c("X","Y")

        grid <- ascgen(fakeda, cellsize=standard.step)

        hr <- kernelUD(SpatialPoints(pts), h, grid = grid)
        rast <- as.matrix(hr[,1])
        attr(rast, "h") <- hr@h$h
        try(attr(rast, "h.grid") <- hr@h$CV[,1], T)
        try(attr(rast, "h.errors") <- hr@h$CV[,1], T)
        rast
}

ascgen <- function (xy, cellsize = NULL, nrcol = NULL, count = TRUE)
{
    if (!inherits(xy, "SpatialPoints"))
        stop("xy should inherit the class SpatialPoints")
    if (is.null(cellsize) & is.null(nrcol))
        stop("One of the parameters cellsize or nrcol should be specified")
    if (!is.null(cellsize) & !is.null(nrcol))
        warning("cellsize and nrcol specified.\nOnly cellsize is taken into account")
    if (ncol(coordinates(xy)) > 2)
        stop("xy should be defined in two dimensions")
    pxy <- proj4string(xy)
    xy <- coordinates(xy)
    xl <- c(min(xy[, 1]), max(xy[, 1]))
    yl <- c(min(xy[, 2]), max(xy[, 2]))
    rx <- xl[2] - xl[1]
    ry <- yl[2] - yl[1]
    u <- rx
    ref <- "x"
    if (ry > rx) {
        u <- ry
        ref <- "y"
    }
    xll <- xl[1]
    yll <- yl[1]
    if (is.null(cellsize)) {
        cellsize <- u/nrcol
    }
    cx <- ceiling(rx/cellsize) + 1
    cy <- ceiling(ry/cellsize) + 1
    xx <- seq(xl[1], xl[2], by = cellsize)
    yy <- seq(yl[1], yl[2], by = cellsize)
    grid <- expand.grid(yy, xx)[, 2:1]
    asc <- data.frame(x = rep(0, nrow(grid)))
    coordinates(asc) <- grid
    gridded(asc) <- TRUE
    if (count) {
        uu <- overlay(asc, SpatialPoints(xy))
        uu <- table(uu)
        asc <- asc[[1]]
        asc[as.numeric(names(uu))] <- uu
        asc <- data.frame(x = asc)
        coordinates(asc) <- grid
        gridded(asc) <- TRUE
    }
    if (!is.na(pxy))
        proj4string(asc) <- CRS(pxy)
    return(asc)
}



grid.error <- function(est, true){
        return(sum(abs(est-true))/sum(true))
}

grid.area <- function(grid){
        return(sum(grid))
}


binarize <- function(grid, isopleth=.95, step = standard.step){
        sorted <- sort(grid, decreasing=T, method="quick")
        #print(head(sorted))
        cum.sorted <- cumsum(sorted)*step*step
        #print(head(cum.sorted))
        cutoff <- sorted[which(cum.sorted>isopleth)[1]]
        if(is.na(cutoff)){
                cutoff <- 0
        }
        pmin(pmax(ceiling(grid-cutoff), 0),1)
        #ifelse(grid>cutoff,1,0)
}

fast.ud <- .kernelUDs <- function(xy, h="href",
xg=seq(standard.xs[1]+standard.step/2,standard.ys[2]-standard.step/2, standard.step),
yg=seq(standard.xs[1]+standard.step/2,standard.ys[2]-standard.step/2, standard.step))
{


     toto<-.C("kernelhr", double(length(xg)*length(yg)),as.double(xg),
                  as.double(yg),
                  as.integer(length(yg)), as.integer(length(xg)),
                  as.integer(nrow(xy)), as.double(h),
                  as.double(xy[,1]), as.double(xy[,2]),
                  PACKAGE="adehabitatHR")


        return(t(apply(matrix(toto[[1]], length(xg),length(yg), byrow=T),1,rev)))

}

# xy <- generate.pts(100)
# jackknife.kernel(xy, seq(0.2, 1, by=0.1), 0.8)
jackknife.kernel <- function(xy, h, isopleth){

        #res <- sapply(h, function(x){
        #       jackknife.h(xy, x, isopleth)
        #})

        npoints <- nrow(xy)
        valid <- c()
        p <- c()
        out.h <- c()

        for(i in 1:length(h)){
                out.h <- c(out.h, h[i])
                res <- jackknife.h(xy, h[i], isopleth)

            ci <- binom.confint(res, npoints, methods = "wilson", conf.level = 0.7)
            #print(ci)

            p <- c(p, res[i]/npoints)
            if(ci$lower < isopleth && ci$upper > isopleth){
                valid <- c(valid, TRUE)
                break;
            }else{
                valid <- c(valid, FALSE)
            }
        }

        return(data.frame(h=out.h, p=p, valid=valid))
}

jackknife.h <- function(points, h, isopleth){
        print( paste( "Jackknifing h:", h ) )


        count <- 0
        for(i in 1:nrow(points)){
                coord <- make.coord(unlist(points[i,]))
                ud <- binarize(fast.ud(points[-i,], h), isopleth = isopleth)
                count <- count + ud[coord[1], coord[2]]
        }
        return( count )


}

# LoCoH Code
# xy <- generate.pts(50, "bridge")
# res <- jackknife.a.locoh(xy, seq(1, 50, by=5), isopleth = 0.5)

jackknife.a.locoh <- function(xy, a, isopleth = 0.9){

        nearest.neighbors <- get.nearest.neighbors(xy, max(a)*1000)
        nearest.neighbors$ids <- nearest.neighbors$ids-1;
        res <- testA(as.matrix(xy), a, isopleth, as.matrix(nearest.neighbors$ids),
as.matrix(nearest.neighbors$distances))

        #print(res)
        #return(res)

        npoints <- nrow(xy)
        ps <- c()
        areas <- c()
        areas <- rep(1e10, length(a))

        for(i in 1:length(a)){
            ps <- c(ps, res$contained[i]/npoints)
            entry <- as.character(i)
            count <- res[[entry]][["count"]]
            #print(i)
            #print(entry)
            #print(count)
            #print(res[[entry]])

            hulls <- list()
            if(length(count)==0){
                print("Error occurred with areas, setting area to high value")
            }else if(count==0){
                        areas[i] <- 0
            }else{

                # for(j in 1:count){
                #       hulls[[j]] <- as(data.frame(x=res[[entry]][[as.character(j)]]$x,
y=res[[entry]][[as.character(j)]]$y), "gpc.poly")
                # }
                #
                # iso <- punion(hulls)
                        # areas <- c(areas, area.poly(iso))

                        for(j in 1:count){
                                x <- res[[entry]][[as.character(j)]]$x
                                y <- res[[entry]][[as.character(j)]]$y
                        hulls[[j]] <- Polygon(cbind(c(x, x[1]), c(y, y[1])))
                }

                        tryCatch(
                                {
                                spolys <- SpatialPolygons(list(Polygons(hulls, "hulls")))
                                area <- gArea(gUnionCascaded(spolys))
                                        areas[i] <- area
                                },
                        error = function(e) {print("Error occurred with areas, setting area 
to high value")}
                        )
                }
        }

        return(data.frame(a=a, p=ps, Area=areas, isopleth = rep(isopleth, length(a))))
}

#tryCatch({
#                       spolys <- SpatialPolygons(list(Polygons(hulls, "hulls")))
#                       area <- gArea(gUnionCascaded(spolys))
#                               areas <- c(areas, area)
#                               },
#                               error=function(e){
#                                       areas <- c(areas, 1e10)
#                               }
#                       )

jackknife.a.locoh.slow <- function(xy, a, isopleth = 0.9){
        xy.simple <- as.matrix(xy)

        nearest.neighbor.cache <- get.nearest.neighbors(xy.simple, max(a))

        res <- sapply(a, function(a){
                print(paste("Jackknifing a:",a))
                jackknife.a(xy.simple, a = a, isopleth = isopleth, 
nearest.neighbor.cache=nearest.neighbor.cache)
        })
        res
}

jackknife.a <- function(xy, a, isopleth = 0.9, nearest.neighbor.cache = NULL){
        npoints <- nrow(xy)

        if(is.null(nearest.neighbor.cache)){
                nearest.neighbor.cache <- get.nearest.neighbors(xy, a)
        }

        nearest.neighbors <- nearest.neighbor.cache$ids
        nearest.neighbors.distance <- nearest.neighbor.cache$distances

        contained <- 0 #just one isopleth


        for(p in 1:npoints){

        hull.count <- 0
        hull.points <- list()
        hull.areas <- c()
        hull.ids <- list()

        for (i in (1:npoints)[-p]) {
                nearest.neighbors.distances <- cumsum(nearest.neighbors.distance[i, 
c(nearest.neighbors[i, ] != p)])
                maxi <- rev(which(nearest.neighbors.distances <= a))[1]

                ##if(p==11){
                #       print(nearest.neighbors.distance[i,])
                #       print(nearest.neighbors.distances)
                #}
                if(is.na(maxi)){
                        maxi <- 0
                }
                        maxi <- max(maxi,2) #FIXME SFR
                if (maxi >= 2) {
                        #if(p==11){
                        #      print(paste("p:", p, "i:",i, "maxi:",maxi))
                        #      stop()
                        #}
                        hull.count <- hull.count + 1
                        hull.point.ids <- c(i, nearest.neighbors[i,
c(nearest.neighbors[i, ] != p)][1:maxi])
                        xytmp <- xy[hull.point.ids, ]
                        hp <- chull(xytmp[, 1], xytmp[, 2])
                        hull.points[[hull.count]] <- xytmp[hp,]
                        hull.ids[[hull.count]] <- hull.point.ids
                        hull.areas <- c(hull.areas,
calc.hull.area(hull.points[[hull.count]]))
                }
        }

        if (! hull.count > 0){
                stop("No hulls were created; try again with a larger value for a.")
        }
        #if(p==11){
            #    for(k in 1:hull.count){
            #          print("--");
            #          print(paste("Area:", hull.areas[k]))
            #          print(paste("Count:", length(hull.ids[[k]])))
            #          print(paste("Density:", length(hull.ids[[k]])/hull.areas[k]))
            #    }
            #
            #    stop("end test")
        #}

        densities <- sapply(hull.ids, function(x){length(x)})/hull.areas

        size.order <- rev(order(densities)) # order by density

        ids <- c()
                for(q in 1:hull.count){
                        k <- size.order[q]

                        #if( p==11){
                        #      print("missed")
                        #      print(hull.points[[k]]);
                        #      print(xy[p,1:2])
                        #}


                if(point.in.hull(xy[p,1], xy[p,2], hull.points[[k]])){
                        contained <- contained+1

                        #print(paste("IN HULL k-",k,"p-",p));

                        #print(hull.points[[k]]);
                               #print(xy[p,1:2])

                        break
                }

                if(q>=hull.count*isopleth){           #experimental create by hull count
                        break
                }
        }
    }

    #return(contained)
    #return(abs(contained/npoints-isopleth))

    ci <- binom.confint(contained, npoints, methods="wilson", conf.level = 0.95)
    print(ci)
    if(ci$lower < isopleth && ci$upper > isopleth){
        return(1)
    }else{
        return(0)
    }

}

point.in.hull <- function(x, y, hull){
        npoints <- nrow(hull)
        inPoly <- FALSE

        vertx <- hull[,1]
        verty <- hull[,2]
        j <- npoints
        for(i in 1:npoints){
                if( ((verty[i]>y) != (verty[j]>y)) && (x < (vertx[j]-vertx[i]) * (y-
verty[i]) / (verty[j]-verty[i]) + vertx[i])){
                        inPoly <- (! inPoly)

                }
                j <- i
        }
        return(inPoly)
}


rectangle <- function(w,h,cx,cy){
        owin(c(cx-w/2,cx+w/2), c(cy-h/2, cy+h/2))
}

homerange.boundaries <- function(type=0){
        boundaries <- list()

        if(type==0){ #three concentric circles
                boundaries[["100"]] = disc(radius=3,npoly=1000)
                boundaries[["95"]] = disc(radius=2.5, npoly=1000)
                boundaries[["50"]] = disc(radius=1, npoly=1000)
        }else if(type==1){ #circle, square, square
                boundaries[["50"]] = disc(radius=1, center=c(1,0), npoly=1000)
                boundaries[["95"]] = rectangle(3,3, 0,0)
                boundaries[["100"]] = rectangle(4,4,0,0)
        }else if(type=="circles"){ #three concentric circles with a whole in the middle
                boundaries[["100"]] = disc(radius=3, npoly=1000)
                boundaries[["80"]] = disc(radius=2,  npoly=1000)
                boundaries[["20"]] = disc(radius=.75, center=c(1,0), npoly=1000)
                boundaries <- add.hole(boundaries, disc(radius=0.5, centre=c(.75,0),
npoly=1000))
        }else if(type=="starburst"){
                boundaries[["100"]] <- owin(poly=list(x=c(0,-3,-3,1,4,2),y=c(0,3,-4,-2,-
4,0) ) )
                boundaries[["80"]] <- owin(poly=list(x=c(0,-3,1,4,2),y=c(0,-4,-2,-4,0) ) )
                boundaries <- add.hole(boundaries, rectangle(1.5,1,-.5,-1.2))

                #boundaries[["100"]] <- owin(poly=list(x=c(0,2,2,10,2,2,0,-7,-2,-
10)*.5,y=c(-3,-8,-2,-3,1,10,3,8,2,0)*.5))
                #boundaries[["80"]] <- owin(poly=list(x=c(0,2,2)*.5,y=c(-3,-8,-2)*.5))
                #boundaries <- add.hole(boundaries, rectangle(1.5,3,.25,0))
        }else if(type=="bridge"){
                boundaries[["100"]] <- disc(radius=2*.2, centre=c(0-2.5*5,0)*.2)
                boundaries[["100"]] <- union.owin(boundaries[["100"]], rectangle(14*.2,
1.5*.2, 8*.2-2.5, 0))
                boundaries[["80"]] <- disc(radius=5*.2, centre=c(20-2.5*5,0)*.2)
                boundaries <- add.hole(boundaries, disc(radius=3*.2, centre=c(20-
2.5*5,0)*.2))
        }else if(type=="walk"){
                boundaries[["100"]] <- disc(radius = (random.walk.steps +
random.walk.start)*random.walk.scale(), centre=c(0, 0))
               boundaries[["95"]] <- disc(radius = random.walk.boundary(0.95), centre=c(0,
0))
                boundaries[["90"]] <- disc(radius = random.walk.boundary(0.9), centre=c(0,
0))
                boundaries[["80"]] <- disc(radius = random.walk.boundary(0.8), centre=c(0,
0))
```

```
                boundaries[["50"]] <- disc(radius = random.walk.boundary(0.5), centre=c(0,
0))
        }else{
                stop("unrecognized type")
        }
        class(boundaries) <- "Homerange.Templates"
        boundaries
}

# p <- generate.pts(10000,"walk")
# b <- homerange.boundaries("walk")
# sum(sapply(1:nrow(p), function(i){inside.owin(p[i,1],p[i,2], b[[3]])}))/nrow(p) # 0.90
# sum(sapply(1:nrow(p), function(i){inside.owin(p[i,1],p[i,2], b[[5]])}))/nrow(p) # 0.50

if(F){
        iso <- z3[[1]][[2]][["0.8"]][["a.locoh.pure"]][["isopleth"]]
        sum(sapply(1:nrow(p), function(i){
                c <- make.coord(unlist(p[i,]))
                iso[c[1], c[2]]
        }))/nrow(p)
}



add.hole<-function(isopleths, hole){
         for(i in 1:length(isopleths)){
                isopleths[[i]] <- setminus.owin(isopleths[[i]], hole, eps=.01)
         }
         return(isopleths)
}



calc.hull.area <- function(x, y=NULL)
{
         if(is.null(y)) {
                 if(is.matrix(x) && ncol(x) == 2.) {
                        #print("A")
                         y <- x[, 2.]
                         x <- x[, 1.]
                 }
                 else if(!is.null(x$X) && !is.null(x$Y)) {
                        #print("B")
                         y <- x$Y
                         x <- x$X
                 }else{
                        stop("Bad hull format");
                 }
         }
         x <- c(x, x[1.])
         y <- c(y, y[1.])
         i <- 2.:length(x)
         return(0.5 * sum(x[i] * y[i - 1.] - x[i - 1.] * y[i]))
}


plot.LoCoH <- function(homerange,  border = NA, gr =
rev(heat.colors(length(homerange$isopleths))), add.points=TRUE, ...){


        opar <- par(mfrow = n2mfrow(length(homerange$a)), mar = c(0, 0, 2, 0))
         on.exit(par(opar))
        xxx <-  homerange$xy
                rx <- range(xxx[, 1])
        ry <- range(xxx[, 2])
                for(a in homerange$a){
                        plot(homerange$xy, ty = "n", asp = 1, main=a, xlim = rx, ylim = ry,
axes = FALSE, ...)

                        if(is.null(homerange[[as.character(a)]]$isopleth.polygons)){
                                homerange[[as.character(a)]]<-
create.isopleth.polygons(homerange[[as.character(a)]])
                        }
                    for(i in 1:length(homerange[[as.character(a)]]$isopleth.polygons)){

        plot(noholes.poly(homerange[[as.character(a)]]$isopleth.polygons[[i]]), poly.args
= list(col = gr[i],  border = border), add = TRUE)
                }
                if(add.points){
                        points(homerange$xy)
                }
                box()
            }

}

LoCoH.a.bounds <- function(xy){
        require(spatstat)
        npoints <- nrow(xy)

        lower.bound <- min(nndist(xy, k=2))
        upper.bound <- a.thumb(xy)*8#quantile(rowSums(nndist(xy, k=1:(npoints-1))),
0.50)/2
        return(c(lower.bound, upper.bound))
}

get.nearest.neighbors <- function(xy, max.a){
        require(spatstat)
        npoints <- nrow(xy)
        #locate nearest neighbors
        nearest.neighbor.search.k <- 15
        if(nearest.neighbor.search.k>(npoints-1)){
          nearest.neighbor.search.k = npoints-1;
        }

    nearest.neighbors.distance <- nndist(xy, k=1:nearest.neighbor.search.k)
    row.summations <- rowSums(nearest.neighbors.distance)
    while(length(row.summations[row.summations>max.a]) != npoints &&
nearest.neighbor.search.k<npoints-1){ #greater than rather than greater than or equal so
we can be sure to have an extra point for jackknifing

       nearest.neighbor.search.k <- nearest.neighbor.search.k*2;
       if(nearest.neighbor.search.k>(npoints-1)){
         nearest.neighbor.search.k = npoints-1;
       }
       nearest.neighbors.distance <- nndist(xy, k=1:nearest.neighbor.search.k);
       row.summations <- rowSums(nearest.neighbors.distance)
    }

    nearest.neighbors <- nnwhich(xy, k=1:nearest.neighbor.search.k)

    cache <- list(ids=nearest.neighbors, distances = nearest.neighbors.distance)
    return(cache)
}

create.isopleth.polygons <- function(homerange){
        require(gpclib)
        if(is.null(homerange$isopleth.polygons)){
                isopleth.polygons <- list()
        hulls <- sapply(homerange$hull.points, function(x) {
                        hull.points <- homerange$xy[x,]
                        convex.hull.ids <- chull(hull.points)
                        as(hull.points[convex.hull.ids,], "gpc.poly")
        })

        used.hulls <-c()
        poly.base <- NULL
        for (i in length(homerange$isopleths):1) {
                new.hulls <- unique(unlist(homerange$isopleth.hulls[[i]]), used.hulls)
                used.hulls <- c(new.hulls, used.hulls)
                hull.polys <- unlist(hulls[new.hulls])
                if(! is.null(poly.base)){
                        hull.polys     <- c(poly.base, hull.polys)
                }
                isopleth.polygons[[i]] <- punion(hull.polys)
                poly.base <- isopleth.polygons[[i]]
         }
         homerange$isopleth.polygons <- isopleth.polygons
        }
        return(homerange)
}

punion <- function(polygons)
{
        if (!all(sapply(polygons, is, "gpc.poly")))
                stop("'...' must all be 'gpc.poly'")
        ## avoid method look-up
        to_numeric <- selectMethod(coerce, c("gpc.poly", "numeric"))
        vec <- to_numeric(polygons[[1]])
        for (p in polygons[-1]) {
                #print(p)
                #plot(as(vec, "gpc.poly"))
                #plot(p)
                #print(get.pts(as(vec, "gpc.poly")))
                #print(get.pts(p))
                clip <- to_numeric(p)
                vec <- .Call("Rgpc_polygon_clip", vec, clip, 3, PACKAGE="gpclib")
        }
        if (identical(vec, 0))
                new("gpc.poly")
        else
                as(vec, "gpc.poly")
}



LoCoH.binary <- function(xy, a, isopleth){
        print("Creating LoCoH binary...")
        hr <- LoCoH(xy, a, isopleths = c(isopleth), verbose = F)
        item <- hr[[as.character(hr$a)]]
        rasterize.owin( as.owin(item$isopleth.polygons[[1]]) )
}

rasterize.owin <- function(poly){
        r <- raster(ncols = (standard.xs[2]-standard.xs[1])/standard.step, nrows = 
(standard.ys[2]-standard.ys[1])/standard.step, xmn = standard.xs[1], xmx = standard.xs[2], 
ymn = standard.ys[1], ymx = standard.ys[2] )
        mat <- as.matrix(rasterize(owin2SP(poly), r))
        mat[is.na(mat)] <- 0
        return(apply(apply(mat,1,rev),2,rev))
}

LoCoH <- function(xy, a, isopleths = c(1.00, 0.95, 0.50), cache = NULL, 
create.isopleth.polygons=TRUE, verbose=TRUE){
        names(xy)<-c("X", "Y")

        #set things up
        a <- vectorize.parameter(a)
        npoints <- nrow(xy)
        nearest.neighbor.cache <- cache

        #prepare output
        results <- list()
        results[["a"]] <- a
        results[["xy"]] <- xy
        results[["isopleths"]] <- isopleths

        if(is.null(nearest.neighbor.cache)){
                nearest.neighbor.cache <- get.nearest.neighbors(xy, max(a))
        }
        nearest.neighbors <- nearest.neighbor.cache$ids
        nearest.neighbors.distance <- nearest.neighbor.cache$distances
```

```
    for(aVal in a){
        if(verbose){
                print(paste("Analyzing for 'a' of", aVal))
        }
        hull.count <- 0
        hull.points <- list()
        hull.area <- list()



        #calculate hulls
        for (i in 1:npoints) {
                nearest.neighbors.distances <- cumsum(nearest.neighbors.distance[i, ])
                maxi <- rev(which(nearest.neighbors.distances <= aVal))[1]

                if(is.na(maxi)){
                        maxi <- 0
                }
                        maxi <- max(maxi,2) #FIXME XXX
                if (maxi >= 2) {
                        hull.count <- hull.count + 1
                        hull.point.ids <- c(i, nearest.neighbors[i, 1:maxi])
                        xytmp <- xy[hull.point.ids, ]

                        convex.hull.ids <- chull(xytmp[, 1], xytmp[, 2])
                        hull.area[[hull.count]] <- calc.hull.area(xytmp[convex.hull.ids, ])
                        hull.points[[hull.count]] <-  hull.point.ids
                }


        }

        if (!length(hull.points) > 0){
                stop("No hulls were created; try again with a larger value for a.")
        }

        #sort hulls
        numpoints <- sapply(hull.points, length)
        density <- unlist(numpoints)/unlist(hull.area)
        ord <- rev(order(density))
        hull.points <- hull.points[ord]
        hull.area <- hull.area[ord]


        #merge hulls to make isopleths
        isopleth.points <- vector("list", length(isopleths))
        isopleth.hulls <- vector("list", length(isopleths))

                if(F){#standard locoh create isopleths by point count
         new.points <- c()
         next.points <- c()
         for (i in 1:length(hull.points)) {
                new.points <- unique(c(new.points, hull.points[[i]]))

            if(length(hull.points)>i){
                next.points <- unique(c(new.points, hull.points[[i+1]]))
            }

            for(j in 1:length(isopleths)){
                if(((length(next.points)/npoints > isopleths[j]) ||
length(hull.points)==i) && is.null(isopleth.points[[j]])){
                        isopleth.hulls[[j]] <- 1:i
                                isopleth.points[[j]] <- new.points
                        }
                }
        }
        }

        if(T){ #experimental create isopleths by hull count not point ocunt
        for( i in 1:length(isopleths)){
               print(isopleths[i])
               isopleth.hulls[[i]] <- 1:ceiling(length(hull.area)*isopleths[i])
               p <- c()
               for(j in 1:ceiling(length(hull.area)*isopleths[i])){
                      p <- c(p, hull.points[[j]])
               }
               isopleth.points[[i]] <- unique(p)
        }
        }


        results[[as.character(aVal)]] <-  list( "isopleth.points" = isopleth.points,
"isopleth.hulls" = isopleth.hulls, "hull.points" = hull.points, "isopleths" = isopleths,
"xy" = xy)

        if(create.isopleth.polygons){
               results[[as.character(aVal)]]<-
create.isopleth.polygons(results[[as.character(aVal)]])
        }
        class(results[[as.character(aVal)]])<-"LoCoH.result"
    }


    results[["cache"]] <- nearest.neighbor.cache

    class(results) <-  "LoCoH"
    return(results)
}



vectorize.parameter<-function (x)
{
    if (is.null(x))
        return(x)
    if (is.character(x))
        x <- as.numeric(unlist(strsplit(x, " *, *")))
    return(sort(x))
}
owin.gpc.poly <- function(window)
{
if(is.null(window$bdry))
   return(as(cbind(c(window$xrange,rev(window$xrange)),
           c(rep(window$yrange[1],2),rep(window$yrange[2],2))),
           "gpc.poly"))
else
    return(as(cbind(window$bdry[[1]]$x,window$bdry[[1]]$y),"gpc.poly"))
}

noholes.poly <- function (x)
{
    pts <- get.pts(x)
    xvals <- lapply(pts, function(x) x$x)
    yvals <- lapply(pts, function(x) x$y)
    holes <- unlist(lapply(pts, function(x) x$hole))
    container <- array(dim = length(holes))
    hp <- which(holes == TRUE)
    pp <- which(holes == FALSE)
    for (k in hp) {
        for (i in pp) {
            pa <- as(data.frame(x = xvals[[k]], y = yvals[[k]]),
                "gpc.poly")
            pb <- as(data.frame(x = xvals[[i]], y = yvals[[i]]),
                "gpc.poly")
            if (area.poly(union(pa, pb)) - (area.poly(pa) + area.poly(pb)) <
                0) {
                container[k] <- i
                break
            }
        }
    }
    newPoly <- new("gpc.poly")
    for (v in 1:length(pp)) {
        i = pp[v]
        pxs <- unlist(xvals[[i]])
        pys <- unlist(yvals[[i]])
        hp <- which(container == i)
        while (length(hp) > 0) {
            sizeP <- length(pxs)
            xs <- xvals[hp]
            ys <- yvals[hp]
            sizeH <- sapply(xs, function(x) length(x))
            sumSizeH <- c(0, cumsum(sizeH))
            holeIDs <- array(dim = 0)
            for (k in 1:length(sizeH)) {
                holeIDs <- c(holeIDs, rep(k, sizeH[k]))
            }
            allxs <- unlist(xs)
            allys <- unlist(ys)
            dists <- matrix(ncol = sizeP, nrow = length(allxs))
            mins <- array(dim = sizeP)
            minsIndex <- array(dim = sizeP)
            for (k in 1:sizeP) {
                for (j in 1:length(allxs)) {
                  dists[j, k] <- ((pxs[k] - allxs[j])^2 + (pys[k] -
                    allys[j])^2)^0.5
                }
                minsIndex[k] <- which.min(unlist(dists[, k]))
                mins[k] <- unlist(dists[, k])[minsIndex[k]]
            }
            minP <- which.min(mins)
            minH <- minsIndex[minP]
            holeID <- holeIDs[minH]
            minH <- minH - sumSizeH[holeID]
            holeID <- hp[holeID]
            xpts <- unlist(xvals[holeID])
            ypts <- unlist(yvals[holeID])
            pxs <- c(pxs[1:minP], xpts[minH:length(xpts)], xpts[1:minH],
                pxs[minP:length(pxs)])
            pys <- c(pys[1:minP], ypts[minH:length(xpts)], ypts[1:minH],
                pys[minP:length(pys)])
            hp <- setdiff(hp, holeID)
        }
        newPoly <- append.poly(newPoly, as(data.frame(x = pxs,
            y = pys), "gpc.poly"))
    }
    return(newPoly)
}

owin2Polygons <- function(x, id="1") {
  stopifnot(is.owin(x))
  x <- as.polygonal(x)
  closering <- function(df) { df[c(seq(nrow(df)), 1), ] }
  pieces <- lapply(x$bdry,
                   function(p) {
                     Polygon(coords=closering(cbind(p$x,p$y)),
                             hole=is.hole.xypolygon(p))  })
  z <- Polygons(pieces, id)
  return(z)
}

owin2SP <- function(x) {
  stopifnot(is.owin(x))
  y <- owin2Polygons(x)
  z <- SpatialPolygons(list(y))
  return(z)
}


true.grid <- function(dists = 1, seed= 1, step=standard.step, xs=standard.xs,
ys=standard.ys){
        if(is.numeric(dists)){
                ncol <- (ys[2]-ys[1])/step
                nrow <- (xs[2]-xs[1])/step
                grid <- matrix(0, nrow = nrow, ncol = ncol)

                locs = list()
                ss = list()
                rhos = list()
                for(i in 1:dists){
                        set.seed(seed)
                        locs[[i]] = c(rnorm(1,sd=.7), rnorm(1,sd=.7))
                        ss[[i]] = runif(2, 0.6, 1)
                        rhos[[i]] = runif(1, -0.9, 0.9)
                        seed <- runif(1)*100000
                }

                for(x in 1:nrow){
                        for(y in 1:ncol){
                                for(i in 1:dists){
                                        grid[x,1+nrow-y] <- grid[x,1+nrow-
y]+dmultnorm(x*step+xs[1]-step/2, y*step+ys[1]-step/2, locs[[i]], ss[[i]], rhos[[i]])
                                }
                        }
                }


                return(grid/dists)
        }else{

                boundaries <- homerange.boundaries(type=dists)

                levels <- sort(as.numeric(names(boundaries)))
                level.strings <- as.character(levels)
                levels <- levels/100


                base <- rasterize.owin(as.owin(boundaries[[level.strings[1]]]))
                base <- base/(sum(base)*standard.step^2)*levels[1]

                for(i in 2:length(levels)){
                        #print(i)

                        win <- boundaries[[level.strings[i]]]
                        for(k in (i-1):1){
                                win <- setminus.owin(win, boundaries[[level.strings[k]]],
eps=.1)
                        }

                        base2 <- rasterize.owin(win)
                        base2 <- base2/(sum(base2)*standard.step^2)*(levels[i]-levels[i-1])
                        base <- base+base2
                }

                return(base)
        }

}


generate.pts <- function(n, dists = 1, seed = 1){
        if(is.numeric(dists)){
                pts <- data.frame()
                each.n <- rep(n/dists, dists)
                each.n <- floor(each.n)
                if(dists>1){
                        each.n[1] = each.n[1]+sum(n/dists-each.n)
                }

                new.seed <- runif(1)*100000

                for(i in 1:dists){
                        set.seed(seed)
                        loc = c(min(rnorm(1,sd=.7), 2.5), min(rnorm(1,sd=.7), 2.5))
                        cov = make.cov(runif(2, 0.5, 0.95), runif(1, -0.8, 0.8))
                        seed <- runif(1)*100000
                        set.seed(new.seed)
                        new.pts <- rmultnorm(each.n[i], matrix(loc,2,1), 
vmat=matrix(cov,2,2))
                        while((T %in% (new.pts[,1]<standard.xs[1])) | (T %in% 
(new.pts[,1]>standard.xs[2])) | (T %in% (new.pts[,2]<standard.ys[1])) | (T %in% 
(new.pts[,2]>standard.ys[2]))){
                                print("Point boundary violation - regenerating")
                                loc = c(min(rnorm(1,sd=.7), 2), min(rnorm(1,sd=.7), 2))
                                cov = make.cov(runif(2, 0.5, .9), runif(1, -0.8, 0.8))
                                new.pts<-rmultnorm(each.n[i], matrix(loc,2,1), 
vmat=matrix(cov,2,2))
                        }
                        pts<-rbind(pts, new.pts)
                        new.seed <- runif(1)*100000
                }
                return(pts)
        }else if(dists=="walk"){
                return(random.walk(n/random.walk.reps))
        }else{
                return(simulate.homerange(type = dists, n = n))
        }
}


random.walk <- function(n, mu = c(0, 0), a = c(0.1, 0.1)){
        xs <- c(0)
        ys <- c(0)

        for(i in 1:n){
                xxx
        }
        data.frame(x=xs, y=ys)*random.walk.scale()
}


simulate.homerange <- function(type = 0, n = 500){
        npoints <- n
        boundaries <- homerange.boundaries(type=type)

        levels <- sort(as.numeric(names(boundaries)))
        level.strings <- as.character(levels)
        levels <- levels/100
        points <- list()
        for(i in 1:length(levels)){
                already.plotted<-0

                win <- boundaries[[level.strings[i]]];
                if(i>1){
                        for(k in (i-1):1){
                                win <- setminus.owin(win, boundaries[[level.strings[k]]], 
eps=.1)
                        }
                        already.plotted <- levels[i-1]
                }
                sample.points <- rpoint(20000, win = win); #generate a large number so 
that we have consistancy for random number generation
                points[[level.strings[i]]] <- sample.points[1:round(npoints*(levels[i]-
already.plotted))];
        }
        combined <- points[[1]]
        if(length(levels)>1){
                for(i in 2:length(levels)){
                        combined <- superimpose(combined, points[[i]], W = 
boundaries[["100"]], check=FALSE)
                }
        }
        data.frame(X = combined$x, Y = combined$y)
}
```

```
real.data <- list()
load.real <- function(){
        sets <- c()
        real.data <<- list()

        base <- "~/Dropbox/Homerange/Empirical Data/"

        files <- list.files(base)
        for(file in files){
                l <- nchar(file)
                #print(substr(file, l-3, l))
                if(substr(file, l-3, l) == ".txt"){
                        n <- substr(file, 1, l-4)
                        sets <- c(sets, n)
                        real.data[[paste(n,"Orig")]] <<- read.delim(paste(base, file, 
sep=""))
                        real.data[[n]] <<- scale.real(real.data[[paste(n,"Orig")]])
                }
        }
        attr(real.data, "sets") <<- sets
}

illustrate.methods <- function(data){
        o.par <- par(mfrow=c(1,2), oma = c( 0, 0, .5, 0 ), mai=c(0.1,0.1,.3,0.1))
        p <- SpatialPoints(data)

        plot(data,  xlab="", ylab="", cex.axis=0.0001, tck=0, main="Minimum Convex 
Polygon", pch=20, asp=1, cex.main=0.8)

        hr.mcp <- mcp(p, percent = 100)
        plot(hr.mcp, add=TRUE, lty=2)
        hr.mcp <- mcp(p, percent = 80)
        plot(hr.mcp, add=TRUE, lty=2)
        hr.mcp <- mcp(p, percent = 50)
        plot(hr.mcp, add=TRUE, lty=2)

        hr.ud <- kernelUD(p, h = "LSCV", grid = 120)
        plot(data,  xlab="", ylab="", cex.axis=0.0001, tck=0, main="Kernel Density 
Estimation", pch=20, asp=1, cex.main=0.8)
        plot(getverticeshr(hr.ud, 95), add=TRUE, lty=2)
        plot(getverticeshr(hr.ud, 50), add=TRUE, lty=2)
        plot(getverticeshr(hr.ud, 20), add=TRUE, lty=2)

        #hr.locoh <- LoCoH(data, a = a.thumb(data), isopleths=c( 0.99, 0.80, .50))
        #plot(data)
        #plot(hr.locoh, add=TRUE)

        par(o.par)
}

scale.real <- function(xy){
        nxy <- xy


        xrange <- max(xy[,1]) - min(xy[,1])

        yrange <- max(xy[,2]) - min(xy[,2])

        nxy[,1] <- nxy[,1] - min(nxy[,1]) - xrange/2
        nxy[,2] <- nxy[,2] - min(nxy[,2]) - yrange/2



        nxy[,1] <- nxy[,1] * (standard.xs[2]-standard.xs[1]) / xrange * 0.7
        nxy[,2] <- nxy[,2] * (standard.ys[2]-standard.ys[1]) / yrange * 0.7

        nxy
}

load.real()
```

```
# real.fits <- analyze.real(c(0.5, 0.9), sets = c("Black Bear", "Warbler"), methods =
c("kernel.href", "kernel.lscv",  "a.locoh.pure", "a.locoh.thumb"))
# real.fits <- analyze.real(c(0.5, 0.7, 0.9), sets = c("Black Bear", "Blandings Turtle",
"Elk", "Marsh Hawk", "Warbler"), methods = c("mcp", "kernel.href", "kernel.lscv",
"a.locoh.pure", "a.locoh.thumb"));
# real.fits2 <- add.best.method(real.fits)
# load(output.file("real.fits.rdata")); real.fits <- add.best.method(real.fits)

analyze.real <- function(isopleths = c(0.5, 0.8, 0.9), sets = attr(real.data, "sets"),
methods = c("kernel.href", "kernel.lscv", "kernel.pure", "a.locoh.pure",
"a.locoh.thumb")){
        res <- list("lower" = list(), "upper" = list())

        for(s in sets){
                print(paste("Analyzing Dataset:", s))

                p <- real.data[[s]]
                p2 <- p[sample(1:nrow(p), floor(nrow(p)/2)),]


                ires <- list(points = p)

                attr(ires, "isopleths") <- isopleths
                attr(ires, "set") <- s
                attr(ires, "methods") <- methods

                ires2 <- list(points = p2)

                attr(ires2, "isopleths") <- isopleths
                attr(ires2, "set") <- s
                attr(ires2, "methods") <- methods

                for(iso in isopleths){
                        print(paste("Analyzing Isopleth:", iso))
                        hrs <- get.hrs(p, isopleth = iso, methods = methods)
                        ires[[as.character(iso)]] <- hrs
                        hrs2 <- get.hrs(p2, isopleth = iso, methods = methods)
                        ires2[[as.character(iso)]] <- hrs2
                }
                res[["lower"]][[s]] <- ires2
                res[["upper"]][[s]] <- ires
        }
        attr(res, "isopleths") <- isopleths
        attr(res, "methods") <- methods
        attr(res, "sets") <- sets
        attr(res, "n") <- c(1, 2)
        res
}

# plot.reals(real.fits, isopleths=c(0.5, 0.7, 0.9), methods="pure")
# png("~/Dropbox/PHStats/Empirical.png", width=1000, height=1600, pointsize=42);
plot.reals(real.fits, isopleths=c(0.5, 0.7, 0.9), methods="pure", lwd=5); dev.off()
plot.reals <- function(x, sets = attr(x, "sets"), isopleths = attr(x, "isopleths"),
methods = attr(x, "methods"), lwd=1.5){

        o.par <- par(mfrow = c(length(sets), length(isopleths)), oma = c( 0, 2, 1.5, 0 ),
mai=c(0.06,0.1,0.06,0.1))

        for(s in sets){
                for(m in methods){
                        plot.real(x[[s]], method = m, isopleths = isopleths,
plot.iso=(s==sets[1]), lwd=lwd)
                }
        }

        par(o.par)
}
```

```
plot.real <- function(x, method, isopleths = attr(x, "isopleths"), plot.iso=TRUE,
lwd=1.5 ){
        lty <- 1
        col <- "darkgrey"
        dl <- FALSE
        iso <- isopleths[length(isopleths)]


        p <- x[["points"]]
        p[,1] <- (p[,1]-standard.xs[1])/(standard.xs[2]-standard.xs[1])
        p[,2] <- (p[,2]-standard.ys[1])/(standard.ys[2]-standard.ys[1])

        for( i in length(isopleths):1){
                iso <- isopleths[i]


                plot(p,  xlab="", ylab="", cex.axis=0.0001, tck=0, xlim=c(0,1),
ylim=c(0,1), pch=20, cex=.8, col=grey(.7), asp=1)
                if(plot.iso){
                        mtext( paste0(iso*100, "%-Isopleth"), side=3, cex=.8, line=.5,
font=2)
                }
                if(i==length(isopleths)){
                        mtext(attr(x, "set"), side=2, cex=.8, line=1, font=2)
                }
                #print(x[[as.character(iso)]][[method]][["isopleth"]])
                contour(t(apply(x[[as.character(iso)]][[method]][["isopleth"]], 1, rev)),
labels = paste(iso*100, "%"),  drawlabels = dl, nlevels=1, lwd=lwd, lty=lty, col=grey(.1),
add=T)
        }

        if(length(method)>1){
        #       title( method.name(paste(attr(x, "set"), "–", method)), outer = TRUE )
        }else{
        #       title(attr(x, "set"), outer = TRUE )
        }

}

method.name <- function(s){
        s <- gsub("kernel\\.href", "Reference KDE", s)
        s <- gsub("kernel\\.lscv", "LSCV KDE", s)
        s <- gsub("kernel\\.pure", "ICE KDE", s)
        s <- gsub("a\\.locoh\\.pure", "ICE a-LoCoH", s)
        s <- gsub("a\\.locoh\\.thumb", "Thumb a-LoCoH", s)
        s <- gsub("pure", "Full ICE", s)
        s <- gsub("mcp", "MCP", s)
        s
}

# png(filename="~/Dropbox/PHStats/patterns.png", width=960, height=960, pointsize=24)
# illustrate.simulated()
# dev.off()

illustrate.simulated <- function(){
        o.par <- par(mfrow=c(2,2), mai=c(0.1,0.1,.3,0.1))

        set.seed(1)
        plot(generate.pts(200,dists=1,seed=2), main="Bivariate Normal", xlab="", ylab="",
cex.axis=0.0001, tck=0, asp=1)
       plot(generate.pts(200,dists=3), main="Mixture of Three Bivariate Normals", xlab="",
ylab="", cex.axis=0.0001, tck=0, asp=1)
        plot(generate.pts(200,dists="bridge"), main="Polygonal Isopleths A", xlab="",
ylab="", cex.axis=0.0001, tck=0, asp=1)
        plot(generate.pts(200,dists="starburst"), main="Polygonal Isopleths B", xlab="",
ylab="", cex.axis=0.0001, tck=0, asp=1)

        #plot(generate.pts(200,dists="walk"), main="Fourty Short Random Walks", xlab="",
ylab="", cex.axis=0.0001, tck=0, asp=1)
```

```
        #plot(generate.pts(200,dists="walk"), main="Ten Long Random Walks", xlab="",
ylab="", cex.axis=0.0001, tck=0, asp=1)

        par(o.par)
}

# load.simulated(); add.best.methods(); full.summarize()
simulated.names <- c("gaussian.1", "gaussian.3", "bridge", "starburst")

load.simulated <- function(){
        for(s in simulated.names){
                load(output.file(paste(s,".rdata", sep="")), .GlobalEnv)
        }
}

data <- list()
normal.data <- list()
walk.data <- list()
polygon.data <- list()
add.best.methods <- function(){
        gaussian.1 <<- add.best.method(gaussian.1)
        gaussian.3 <<- add.best.method(gaussian.3)
        bridge <<- add.best.method(bridge)
        starburst <<- add.best.method(starburst)

        data <<- list("Bivariate Normal" = gaussian.1, "Three Bivariate Normals" =
gaussian.3, "Polygonal Isopleths A" = bridge, "Polygonal Isopleths B" = starburst)

        normal.data <<- list("Bivariate Normal" = gaussian.1, "Three Bivariate Normals" =
gaussian.3)


        polygon.data <<- list("Polygonal Isopleths A" = bridge, "Polygonal Isopleths B" =
starburst)
}


full.summarize <- function( methods = attr(gaussian.1, "methods"), points = attr(z, "n"),
kinds = c("geo.err")){

        print("Gaussian 1")
        s <- full.summarize.test(gaussian.1, methods = methods, kinds = kinds, points =
points)
        t <- s[nrow(s),]
        m <- s[nrow(s)-1,]
        print(s)

        print("Gaussian 3")
        s <- full.summarize.test(gaussian.3, methods = methods, kinds = kinds, points =
points)
        t <- t + s[nrow(s),]
        m <- m + s[nrow(s)-1,]
        print(s)

        print("Bridge")
        s <- full.summarize.test(bridge, methods = methods, kinds = kinds, points =
points)
        t <- t + s[nrow(s),]
        m <- m + s[nrow(s)-1,]
        print(s)

        print("Starburst")
        s <- full.summarize.test(starburst, methods = methods, kinds = kinds, points =
points)
        t <- t + s[nrow(s),]
        m <- m + s[nrow(s)-1,]
        print(s)

        if(points != "50"){

                print("Walk 5")
                s <- full.summarize.test(walk.5, methods = methods, kinds = kinds, points
= points)
                t <- t + s[nrow(s),]
                m <- m + s[nrow(s)-1,]
                print(s)

                print("Walk 20")
                s <- full.summarize.test(walk.20, methods = methods, kinds = kinds, points
= points)
                t <- t + s[nrow(s),]
                m <- m + s[nrow(s)-1,]
                print(s)

        }

        print("Total Orders:")
        print(t)

        print("Total Means:")
        print(m)
}

if(F){
        load("~/HR July 12/bridge.full.rdata");
        load("~/HR July 12/starburst.full.rdata");
        load("~/HR July 12/gaussian.1.full.rdata");
        load("~/HR July 12/gaussian.3.full.rdata");

        data <- list("Bivariate Normal" = gaussian.1, "Three Bivariate Normals" =
gaussian.3, "Polygonal Isopleths A" = bridge, "Polygonal Isopleths B" = starburst)
        full.boxplot(data)

        s <- full.latex(data)
        cat(s)
}
full.latex <- function(data, kinds="geo.err"){
        res <- ""
        methods <- c()
        for(name in names(data)){
                print(name)
                d <- data[[name]]
                points <- attr(d, "err.n")
                isopleths <- attr(d, "err.iso")
                methods <- attr(d, "methods")
                res <- paste0(res,
"\\multirow{",length(points)*length(isopleths),"}{*}{",name,"}")

                for(p in points){
                        res <- paste0(res," & ")
                        res <- paste0(res, " \\multirow{",length(isopleths),"}{*}{",p,"}")
                        skip.amp <- T
                        for(iso in isopleths){
                                if(! skip.amp){
                                        res <- paste0(res, "&")
                                }
                                skip.amp <- F
                                res <- paste0(res, " & ", iso*100, "\\%")
                                means <- score.vector(colMeans(summarize.test(d, isopleth =
as.character(iso), points = p, kinds = kinds), na.rm=T))
                                for(m in 1:length(methods)){
                                        v <- means[m]
                                        if(kinds=="geo.err"){
                                                v <- paste0(round(v*100), "\\%")
                                        }
                                        res <- paste0(res, " & ", v)
                                }
                                res <- paste0(res, "\\\\\n")

                        }
                }
                res <- paste0(res, "\\cline{2-",4+length(methods)-1,"}\n")
        }

        column.format <- "l c c | "
        column.text <- " & $n$ & Isopleth "
        for(m in methods){
                column.format <- paste0(column.format, " c")
                column.text <- paste0(column.text, " & ",  method.latex(m))
        }
        column.text <- paste0(column.text, "\\\\\n")
        res <- paste0("\\begin{table}\n\\centering\n\\begin{tabular}{ ",
column.format," }\n ", column.text, "\\cline{2-",4+length(methods)-1,"}\n", res,
"\\end{tabular}\n\\caption{Overview of error results from simulation
study.}\n\\end{table}")
        res
}

latex.summarize <- function(out){
        library(xtable)
        names(out) <- method.latex(names(out))
        print(xtable(out), sanitize.colnames.function = function(x){x})
}

# add.best.methods()
# is.normal <- isopleth.summarize(normal.data)
# is.walk <- isopleth.summarize(walk.data)
# ps.normal <- point.summarize(normal.data)
# ps.polygon <- point.summarize(polygon.data)
# print(is.normal); print(is.walk); print(ps.normal); print(ps.polygon)
isopleth.summarize <- function(data, points = attr(data[[1]], "err.n"), kinds =
"geo.err"){
        isopleths <- attr(data[[1]], "err.iso")
        res <- c()

        for(iso in isopleths){
                print(iso)

                inner <- c()
                for(name in names(data)){
                        print(name)
                        d <- data[[name]]

                        for(p in points){
                                print(paste("Points:",p))
                                summary <- colMeans(summarize.test(d, as.character(iso), p,
kinds =  kinds), na.rm=T)
                                inner <- rbind(inner, summary)
                        }
                }
                res <- rbind(res, score.vector(colMeans(inner)))

        }

        res <- data.frame(res)
        names(res) <- attr(data[[1]], "methods")
        row.names(res) <- paste0(isopleths*100, "%-Isopleth")

        res
}

point.summarize <- function(data, isopleths = attr(data[[1]], "err.iso"), kinds =
"geo.err"){
        points <- attr(data[[1]], "err.n")
        res <- c()

        for(p in points){
                print(p)

                inner <- c()
                for(name in names(data)){
                        print(name)
                        d <- data[[name]]
                        for(iso in isopleths){
                                summary <- colMeans(summarize.test(d, as.character(iso), p,
kinds =  kinds), na.rm=T)
                                inner <- rbind(inner, summary)
                        }
                }
                res <- rbind(res, score.vector(colMeans(inner)))

        }

        res <- data.frame(res)
        names(res) <- attr(data[[1]], "methods")
        row.names(res) <- paste0("n=",points)

        res
}

score.vector <- function(vec){
        #return(order(order(as.vector(vec)))) # method rank
        return(vec) # raw error
        #return(paste(round(vec*100),"%")) # percent error
}

full.summarize.test <- function(z, points = attr(z, "n"), methods = attr(z, "methods"),
kinds = c("geo.err")){
        isopleths <- attr(z, "err.isopleths")
        res <- c()
        for(p in points){
                for(iso in isopleths){
                       summary <- colMeans(summarize.test(z, as.character(iso), p, methods,
kinds), na.rm=T)
                        res <- rbind(res, summary)
                }
        }
        row.names(res) <- paste( rep(points, each=length(isopleths)) , rep(isopleths,
length(points)))

        cm <- colMeans(res)
        res <- rbind(res, cm)
        row.names(res)[nrow(res)] <- "Mean"
        res <- rbind(res, order(order(as.vector(cm))))
        row.names(res)[nrow(res)] <- "Order"

        res
}

summarize.test <- function(z, isopleth = "0.9", points = attr(z, "err.n")[1], methods =
attr(z,"methods"), kinds = c("geo.err")){
        res <- c()
        for(i in 1:attr(z,"reps")){
                row <- c()
                for(m in methods){
                        for(k in kinds){
                                #print("--")
                                #print(m)
                                #print(k)
                                #print(z[[ which(attr(z,
"n")==points) ]][[i]][[isopleth]][[m]][[k]])
                                row <- c(row, z[[ which(attr(z,
"n")==points) ]][[i]][[isopleth]][[m]][[k]])
                        }
                }
                res <- rbind(res,row)
        }
        res <- as.data.frame(res, row.names=1:attr(z,"reps"))
        #print(names(res))
        #print(paste(rep(methods, each=length(kinds)), rep(kinds, length(methods))))
        names(res) <- paste(rep(methods, each=length(kinds)), rep(kinds, length(methods)))
        res
}
```

```
full.boxplot <- function(data, kinds="geo.err"){
        for(name in names(data)){
                print(name)
                d <- data[[name]]
                full.boxplot.test(d, kinds = kinds, main = name)
        }
}


if(FALSE){
        load(output.file("starburst.rdata")); load(output.file("bridge.rdata"));
load(output.file("gaussian.1.rdata")); load(output.file("gaussian.3.rdata"))
        b <- add.best.method(bridge); s <- add.best.method(starburst); g.1 <-
add.best.method(gaussian.1); g.3 <- add.best.method(gaussian.3)
        data <- list("Bivariate Normal" = g.1, "Three Bivariate Normals" = g.3, "Polygonal
Isopleths A" = b, "Polygonal Isopleths B" = s)
        full.boxplot(data)
        cat(full.latex(data))
}

full.boxplot.test <- function(z, methods = attr(z,"methods"), kinds = "geo.err", main =
NULL){
        library(scales)

        isopleths <- attr(z, "err.isopleths")
        points <- attr(z, "err.n")
        o.isopleths <- c()
        o.points <- c()
        o.methods <- c()
        o.y <- c()
        o.col <- c()

        for(i in isopleths){
                print(paste("Isopleth:", i))
                iso <- paste0(i*100, "%-Isopleth")
                for(p in points){
                        print(paste("Sample Size:", p))
                        data <- summarize.test(z, isopleth = as.character(i), points =
as.character(p), methods = methods, kinds = kinds)
                        for(r in 1:nrow(data)){
                                for(c in 1:length(methods)){
                                        o.isopleths <- c(o.isopleths, iso)
                                        o.points <- c(o.points, p)
                                        o.methods <- c(o.methods, methods[c])
                                        o.y <- c(o.y, data[r,c])
                                        if(methods[c]=="pure"){
                                                o.col <- c(o.col, "p")
                                        }else{
                                                o.col <- c(o.col, "n")
                                        }
                                }
                        }
                }
        }

        res <- data.frame(isopleths = o.isopleths, points = o.points, methods = o.methods,
y = o.y, col = o.col)

        res$points = factor(res$points, labels = paste0("n=", points))
        res$col = factor(res$col)

        p <- ggplot(res, aes(factor(methods), y)) + geom_boxplot(aes(fill=col),notch =
TRUE)

        labels <- c()
        for(m in methods){
                labels[m] <- method.symbol(m)
        }
        p <- p + scale_x_discrete(labels = labels) + xlab("Estimator") +
ylab(kind.name(kinds))

        p <- p + facet_grid(isopleths~points)

        p <- p + scale_y_continuous(labels = percent)

        if(! is.null(main)){
                p <- p + labs(title= main)
        }

        p <- p + theme(legend.position="none")
        p <- p + theme(axis.text.x = element_text(colour = "black"))


        print(p)
}

full.method.selection <- function(z, kinds = "geo.err", main = NULL){
        library(scales)

        isopleths <- attr(z, "err.isopleths")
        points <- attr(z, "err.n")
        reps <- attr(z, "reps")
        methods <- attr(z, "methods")
        o.isopleths <- c()
        o.points <- c()
        o.method <- c()
        o.frac <- c()

        for(i in isopleths){
                print(paste("Isopleth:", i))
                iso <- paste0(i*100, "%-Isopleth")

                for(p in points){
                        print(paste("Sample Size:", p))

                        best <- c()

                        for(r in 1:reps){
                                best <- c(best, z[[which(attr(z,
"n")==p)]][[r]][[as.character(i)]][["pure"]][["method"]])
                        }

                        for(m in methods){
                                if(m != "pure"){
                                        o.isopleths <- c(o.isopleths, iso)
                                        o.points <- c(o.points, p)
                                        o.method <- c(o.method, m)
                                        o.frac <- c(o.frac, sum(best==m)/reps)
                                }
                        }
                }
        }

        res <- data.frame(isopleths = o.isopleths, points = o.points, frac = o.frac,
method = o.method)

        res$points = factor(res$points, labels = paste0("n=", points))
        res$method = factor(res$method)

        p <- ggplot(res, aes(x=factor(1), y=frac, fill = method)) +
geom_bar(stat="identity", position="fill")+ coord_polar(theta="y")

        #labels <- c()
        #for(m in methods){
        #       labels[m] <- method.symbol(m)
        #}

        p <- p + theme(axis.ticks = element_blank(),
         axis.text.y = element_blank(),
         axis.text.x = element_blank()) +ylab("")+xlab("")


        p <- p + facet_grid(isopleths~points)

        #p <- p + scale_y_continuous(labels = percent)

        if(! is.null(main)){
                p <- p + labs(title= main)
        }

        p <- p + theme(legend.title=element_blank()) + theme(legend.position = "bottom")
        #p <- p + theme(axis.text.x = element_text(colour = "black"))


        print(p)

        res
}

full.isopleth.selection <- function(z, kinds = "geo.err", main = NULL){
        library(scales)

        isopleths <- attr(z, "err.isopleths")
        points <- attr(z, "err.n")
        reps <- attr(z, "reps")
        all.isopleths <- attr(z, "isopleths")
        o.isopleths <- c()
        o.points <- c()
        o.selected.iso <- c()
        o.frac <- c()

        for(i in isopleths){
                print(paste("Isopleth:", i))
                iso <- paste0(i*100, "%-Isopleth")

                for(p in points){
                        print(paste("Sample Size:", p))

                        best <- c()

                        for(r in 1:reps){
                                best <- c(best, z[[which(attr(z,
"n")==p)]][[r]][[as.character(i)]][["pure"]][["iso"]])
                        }

                        for(m in all.isopleths){
                                o.isopleths <- c(o.isopleths, iso)
                                o.points <- c(o.points, p)
                                o.selected.iso <- c(o.selected.iso, m*100)
                                o.frac <- c(o.frac, sum(best==m)/reps)
                        }
                }
        }

        res <- data.frame(isopleths = o.isopleths, points = o.points, frac = o.frac,
selected.iso = o.selected.iso)

        res$points = factor(res$points, labels = paste0("n=", points))
        res$selected.iso = factor(res$selected.iso)

        p <- ggplot(res, aes(x=factor(1), y=frac, fill = selected.iso)) +
geom_bar(stat="identity", position="fill")+ coord_polar(theta="y")

        #labels <- c()
        #for(m in methods){
        #       labels[m] <- method.symbol(m)
        #}

        p <- p + theme(axis.ticks = element_blank(),
        axis.text.y = element_blank(),
        axis.text.x = element_blank()) +ylab("")+xlab("")



        p <- p + facet_grid(isopleths~points)

        #p <- p + scale_y_continuous(labels = percent)

        if(! is.null(main)){
                p <- p + labs(title= main)
        }

        p <- p + theme(legend.title=element_blank()) + theme(legend.position = "bottom")
        #p <- p + theme(axis.text.x = element_text(colour = "black"))


        print(p)

        res
}



kind.name <- function(k){
        if(k=="geo.err"){
                return("Error")
        }else if(k=="area"){
                return("Scaled Area")
        }else if(k=="p.est"){
                return(expression(hat(p)))
        }else if(k=="p.true"){
                return("p")
        }
        return(k)
}

method.symbol <- function(s){
        return(sapply(s, function(x){

                if(x == "mcp"){
                        return(expression(psi["MCP"]))
                }else if(x == "kernel.href"){
                        return(expression(psi["KDE"]^{paste(italic("h"),"=Ref")}))
                }else if(x == "kernel.lscv"){
                        return(expression(psi["KDE"]^{paste(italic("h"),"=LSCV")}))
                }else if(x == "a.locoh.pure"){
                        return(expression(psi["a-LoCoH"]^{paste(italic("a"),"=ICE")}))
                }else if(x == "a.locoh.thumb"){
                        return(expression(psi["a-LoCoH"]^{paste(italic("a"),"=Thumb")}))
                }else if(x == "pure"){
                        return(expression(psi["ICE"]))
                }

        }))
}

method.latex <- function(s){
        return(sapply(s, function(x){

                if(x == "mcp"){
                        return("$\\psi_\\text{MCP}$")
                }else if(x == "kernel.href"){
                        return("$\\psi_{\\text{KDE}}^{h=\\text{Ref}}$")
                }else if(x == "kernel.lscv"){
                        return("$\\psi_{\\text{KDE}}^{h=\\text{LSCV}}$")
                }else if(x == "a.locoh.pure"){
                        return("$\\psi_\\text{a-LoCoH}^{a=\\text{ICE}}$")
                }else if(x == "a.locoh.thumb"){
                        return("$\\psi_\\text{a-LoCoH}^{a=\\text{Thumb}}$")
                }else if(x == "pure"){
                        return("$\\psi_\\text{ICE}$")
                }

        }))
}



# Size Plot:
if(F){
        starburst.large <- test(n=c(400), reps=40, isopleths=0.8, dists="starburst",
methods = c("mcp", "kernel.href", "kernel.lscv", "a.locoh.pure", "a.locoh.thumb"),
seed=1)

        load(output.file("starburst.rdata"))
        load(output.file("starburst.large.rdata"))
        starburst<-add.best.method(starburst);starburst.large<-
add.best.method(starburst.large)

        load("~/HR July 12/starburst.full.rdata");
        load("~/starburst.large.rdata");
        starburst.large<-add.best.method(starburst.large);
        points <- c()
        method <- c()
        val <- c()
        for(i in 1:3){
                p <- 0
                res <- 0
                if(i==1){
                        p <- 50
                        res <- summarize.test(starburst, "0.8", points=p)
                }else if(i==2){
                        p <- 200
                        res <- summarize.test(starburst, "0.8", points=p)
                }else if(i==3){
                        p <- 400
                        res <- summarize.test(starburst.large, "0.8", points=p)
                }

                for(r in 1:nrow(res)){
                        for(c in 1:ncol(res)){
                                points <- c(points, p)
                                method <- c(method, attr(starburst, "methods")[c])
                                val <- c(val, res[r,c])
                        }
                }
        }

        ggplot(data=data.frame(points=points, methods=as.factor(method), val=val),
aes(x=points, y=val, group=method, color=method)) + stat_summary(fun.data =
"mean_cl_normal", geom="errorbar", position = position_dodge(5),
lty=1)+stat_summary(fun.y = mean, geom="line", position = position_dodge(5)) +
scale_y_continuous(labels = percent_format())+ ylab("Error")+xlab("Sample Size") +
theme(legend.title=element_blank())+ scale_color_discrete(name  = "Method", breaks =
attr(starburst, "methods"), labels = method.symbol( attr(starburst, "methods"))) +
theme(legend.position = "top")

}
```

C++ a-LoCoH Implementation

The following is a C++ implementation of the *a*-LoCoH algorithm which offers order of magnitude performance increases over a native *R* implementation.

```
#include <Rcpp.h>
#include <algorithm>
#include <vector>
using namespace std;
using namespace Rcpp;

typedef std::pair<double, int> density;
bool comparator ( const density& l, const density& r){ return l.first > r.first; }
//reverse sort


struct Point {
  double x, y;

  bool operator <(const Point &p) const {
                return x < p.x || (x == p.x && y < p.y);
        }
};

double cross(const Point &O, const Point &A, const Point &B)
{
        return (A.x - O.x) * (double)(B.y - O.y) - (A.y - O.y) * (double)(B.x - O.x);
}

vector<Point> chull(vector<Point> P)
{
        int n = P.size(), k = 0;
        vector<Point> H(2*n);

        // Sort points lexicographically
        sort(P.begin(), P.end());

        // Build lower hull
        for (int i = 0; i < n; i++) {
                while (k >= 2 && cross(H[k-2], H[k-1], P[i]) <= 0) k--;
                H[k++] = P[i];
        }

        // Build upper hull
        for (int i = n-2, t = k+1; i >= 0; i--) {
                while (k >= t && cross(H[k-2], H[k-1], P[i]) <= 0) k--;
                H[k++] = P[i];
        }

        H.resize(k);
        return H;
}

// [[Rcpp::export]]
double calcHullArea(vector<double> x, vector<double> y){
 int points = x.size();

  double  area = 0. ;

  for (int i = 1; i < points; i++) {
      area += x[i]*y[i-1]-x[i-1]*y[i];
    }
    area += x[0]*y[points-1]-x[points-1]*y[0];

  area = area*0.5;
  /*if(area==0){
    Rprintf("-----");
    for (int i = 0; i < points; i++) {
      Rprintf("\n<%f, %f>", x[i], y[i]);
    }
  }*/
   // Rprintf("\n %f", area);
  //return 1;

  return area;  //XXX fixme, is abs needed?

}


int pointInPolygon(vector<double> vertx, vector<double> verty, double x, double y)
```

```
{
  int npol = vertx.size();
  int i, j, c = 0;
  for (i = 0, j = npol-1; i < npol; j = i++) {
   if( ((verty[i]>y) != (verty[j]>y)) && (x < (vertx[j]-vertx[i]) * (y-verty[i]) /
(verty[j]-verty[i]) + vertx[i])){
      c = !c;
   }
  }
  //Rprintf("\n %i", c);
  //return 1;
  return c;
}


template <typename T>
  string NumberToString ( T Number )
  {
     ostringstream ss;
     ss << Number;
     return ss.str();
  }

// [[Rcpp::export]]
SEXP testA(NumericMatrix xy, NumericVector as, double isopleth, NumericMatrix
nearestNeighborIds, NumericMatrix nearestNeighborDistance){

  bool verbose = false;

  int npoints = xy.nrow();

  List results;
  results["a"] = as;

  IntegerVector res(as.size());

  for(int aCounter = 0; aCounter < as.size(); aCounter++){


    double a = as[aCounter];
    List aresults;

    if(verbose){
      Rprintf("\na: %f", a);
    }

       int contained = 0; //just one isopleth

       for(int p = 0; p < npoints; p++){
       int hullCount = 0;
       vector< vector<double> > hullXs;
        vector< vector<double> > hullYs;
       double hullAreas[npoints-1];
       vector< vector<int> > hullIds;


       for (int counter = 0; counter < (npoints-1); counter++) {
          //Rprintf("\nCounter:%d", counter);
          int i = counter;
          if(counter >= p){
            i++;
          }
          NumericVector tempDistances = nearestNeighborDistance(i, _);
          NumericVector distances(npoints-2);
          int loc = 0;
          for(int j = 0; j < tempDistances.size(); j++){
            //if(p==10){
              // Rprintf("\n temp distance: %f", tempDistances[j]);
              //}
            if(nearestNeighborIds(i,j) != p){
              if(loc==0){
```

```
         distances[loc] = tempDistances[j];
       }else{
         distances[loc] = distances[loc-1] + tempDistances[j];
       }
       loc++;
     }
   }

  //Rprintf("\n  A");

  int maxi = -1;

   for(int k = distances.size()-1; k >= 0; k--){
    // if(p==10){
      // Rprintf("\n distance: %f", distances[k]);
     //}
     if(distances[k] <= a){
       maxi = k;
       break;
     }
   }

                maxi = max(maxi, 1); //FIXME SFR
   //if(p==10){
   //  Rprintf("\n p: %i, i: %i,    maxi: %i", p+1, i+1, maxi+1);
   //  stop("dsdsd");
   //}
   //Rprintf("\n  B");

        if (maxi >= 1) {
               hullCount = hullCount + 1;
               vector<int> hullPointIds;
     hullPointIds.resize(maxi+2);

     NumericVector origIds = nearestNeighborIds(i, _);
     int k=0, added=0;
     while(added <= maxi && k < origIds.size()){
       if(origIds[k] != p && added <= maxi){
         hullPointIds[added+1] = (int) origIds[k];
         added++;
       }
       k++;
     }
     hullPointIds[0] = i;

     vector<Point> pin;
     for(unsigned int k=0; k < hullPointIds.size(); k++){
       Point p = {xy(hullPointIds[k], 0), xy(hullPointIds[k], 1)};
       pin.push_back(p);
     }
               vector<Point> ps = chull(pin);


               vector<double> xs(ps.size()-1);
     vector<double>  ys(ps.size()-1);
     for(int k = ps.size() - 2; k >= 0; k--){
       xs[ps.size() - 2-k] = (double)ps[k].x;
       ys[ps.size() - 2-k] = (double)ps[k].y;
     }
     hullXs.push_back(xs);
     hullYs.push_back(ys);
               hullIds.push_back(hullPointIds);
               hullAreas[hullCount-1] = calcHullArea(xs, ys);
        }
   //return 111;
}


if (! hullCount > 0){
        stop("No hulls were created; try again with a larger value for a.");
 }
```

```
          //Rprintf("\n A");

         vector<int> counts;
         counts.resize(hullCount);
          for(unsigned int k = 0; k < counts.size(); k++){
            counts[k] = hullIds[k].size();
          }

          //Rprintf("\n AAAA");


          vector<density> densities;
          for(unsigned int k=0; k<counts.size(); k++){
            density d = make_pair(counts[k]/hullAreas[k], k);
            densities.push_back(d);
          }

       /*   if(p==10){
         printf("\n Hulls:");
          for(unsigned int k = 0; k<densities.size(); k++){
            printf("\n  --");
            printf("\n  Area: %f", hullAreas[k]);
            printf("\n  Count: %i", counts[k]);
            printf("\n  Density: %f", densities[k].first);
          }
           stop("end");
           }*/

          sort(densities.begin(), densities.end(),comparator);



          //Rprintf("\n B");
        aresults["count"] = isopleth*hullCount;
        for(int q = 0; q < isopleth*hullCount; q++){
                 int k = densities[q].second;
            List hull = List::create(Rcpp::Named("x") = hullXs[k], Rcpp::Named("y") =
hullYs[k]);
            aresults[NumberToString(q+1)] = hull;
        }
               for(int q = 0; q < hullCount; q++){
          //Rprintf("\n C");
                         int k = densities[q].second;
          //Rprintf("\n k: %i", k);
          //Rprintf("\n Density: %f",densities[q].first);


               if(pointInPolygon(hullXs[k], hullYs[k], xy(p,0), xy(p,1))){
                       contained = contained + 1;
           // Rprintf("\n IN HULL : k - %i, p - %i",k,p);

            //for(unsigned int j=0; j<hullXs[k].size(); j++){
            //  Rprintf("\nHull: <%f, %f>", hullXs[k][j],  hullYs[k][j]);
            //}

          //  Rprintf("\n <%f, %f>",xy(p,0), xy(p,1));;
                       break;
               }
               if(q+1 >= isopleth*hullCount){
                       break;
               }
               }

      }
      //Rprintf("\n end");
      res[aCounter] = contained;

      results[NumberToString(aCounter+1)] = aresults;
  }
  results["contained"] = res;
  return results;

}
```

Contingent Kernels

The following implements the contingent kernel method in R.

```
# source("~/Dropbox/Berkeley/Contingent Kernels/CK.R")


# create.kernel.demo(10, 1, 84)
# create.kernel.demo(10, 2, 84)

# plot.kernel(C.gaussian.uniform, c(.05,.15, .45), data.frame(x=c(0.5),r=c( 0)), 1.1)
if (F){
 x<-create.filter.points(600)

 nd <- contingent.density(naive.points(x), C.gaussian.uniform, .3)
 cd <- contingent.density(contingent.points(x), C.gaussian.uniform, .3)

 plot(p<-seq(-8,8,.05), nd(p), type="l"); lines(p, reference.1D(p), col=2)
 plot(p<-seq(-8,8,.05), cd(p), type="l"); lines(p, reference.1D(p), col=2)

 area.under.1D(nd)
 area.under.1D(cd)

 optimize.h.1D(reference.1D, naive.points(x), C.gaussian.uniform)
 optimize.h.1D(reference.1D, contingent.points(x), C.gaussian.uniform)

 plot.comparisons(reference.1D, contingent.density(naive.points(x),
C.gaussian.uniform, .625), contingent.density(contingent.points(x),
C.gaussian.uniform, .288))
}

if(F){
 library(ggplot2); theme_set(theme_bw());
 q<-opt.tester.1D(reps=200, points=c(5, 25, 50, 100, 150, 200))#20 reps -- 5, 25, 50, 100,
150, 200
 q$Kernel.Type <- factor(q$Kernel.Type, levels = rev(levels(q$Kernel.Type)))
 ggplot(q, aes(x=Points*3, y=MISE, group=Kernel.Type,
lty=Kernel.Type))+stat_summary(fun.data = "mean_cl_normal", geom="errorbar", position =
position_dodge(4), lty=1)+stat_summary(fun.y = mean, geom="line", position =
position_dodge(6))+labs(x="", y="")+opts(legend.position="top",
legend.direction="horizontal")+ scale_linetype_discrete(name = "")
 }

if(F){
  #analytical
 observations <- data.frame(r=c(1,2,.3,4,2.5), x=c(-1,1,1,2,3))
 x <- contingent.density(observations, C.gaussian.uniform, 0.1)
 plot(p<-seq(-5,10,.05), x(p), type="l")

  #numeric



 ################

 observations <- data.frame(r=c(2, 2), x=c(-1,1))
 observations <- data.frame(r=c(1,2,.3,4,2.5), x=c(-1,1,1,2,3))
 contingent.kernels <- c()
 for(i in 1:nrow(observations)){
       contingent.kernels <- c(contingent.kernels,
numeric.kernel(test.guassian,function(z){
               if(z <= observations[i,"r"] & z >= -observations[i,"r"]){
                       return(1/(2*observations[i,"r"]))
               }
               return(0)
       }, h=0.1, seq(-10,10,0.01)))
 }

 png("~/Desktop/Convergence/Plot %03d.png")
 o.ck <- contingent.kernels
 for(a in c(0,.01,.1,.25,.5,.75,1)){
       print(paste("a:",a))
       contingent.kernels <- o.ck
        x <- numeric.contingent.density(observations$x, contingent.kernels)
        plot(p<-seq(-5,10,.05), sapply(p, x), type="l", main=paste("a:",a),
xlab="x",ylab="")
       # rug(jitter(observations$x,factor=.1))
        for(i in 1:nrow(observations)){
        #      rug(jitter(c(observations[i,"x"]+observations[i,"r"], observations[i,"x"]-
observations[i,"r"]), factor=.1), col=i+1,lwd=2)
        }


        for(j in 1:10){
               for(i in 1:length(contingent.kernels)){
                       contingent.kernels[[i]] <- scale.ck(observations[i,"x"], o.ck[[i]],
x, a)
               }

                 x <- numeric.contingent.density(observations$x, contingent.kernels)
                 print(paste("j:",j))
                 print(integrate(x,-10,10,subdivisions=10000))
                plot(p<-seq(-5,10,.05), sapply(p, x), type="l",
main=paste("Iteration:",j), xlab="x",ylab="")
                 #rug(jitter(observations$x, factor=.1))
                for(i in 1:nrow(observations)){
                       #rug(jitter(c(observations[i,"x"]+observations[i,"r"],
observations[i,"x"]-observations[i,"r"]), factor=.1), col=i+1,lwd=2)
               }
        }
 }
 dev.off()

 ###########


 observations <- data.frame(r=c(2, 2), x=c(-1,1))
 contingent.kernels <- c()
 for(i in 1:nrow(observations)){
       contingent.kernels <- c(contingent.kernels,
numeric.kernel(test.guassian,function(z){
               if(z <= observations[i,"r"] & z >= -observations[i,"r"]){
                       return(1/(2*observations[i,"r"]))
               }
               return(0)
       }, h=0.1, seq(-10,10,0.01)))
 }
  o.ck <- contingent.kernels
 a<-.5
       print(paste("a:",a))
       contingent.kernels <- o.ck
        x <- numeric.contingent.density(observations$x, contingent.kernels)
        plot(p<-seq(-5,10,.05), sapply(p, x), type="l", main=paste("a:",a))
        rug(jitter(observations$x,factor=.1))
        for(i in 1:nrow(observations)){
               rug(jitter(c(observations[i,"x"]+observations[i,"r"], observations[i,"x"]-
observations[i,"r"]), factor=.1), col=i+1,lwd=2)
        }
```

```
                for(i in 1:length(contingent.kernels)){
                        contingent.kernels[[i]] <- scale.ck(observations[i,"x"], o.ck[[i]],
x, a)
                }

                plot(p<-seq(-5,5,.1), sapply(p,contingent.kernels[[1]]))
                plot(p<-seq(-5,5,.1), sapply(p,contingent.kernels[[2]]))



}

scale.ck <- function(x, source.contingent.kernel, density, a){
        bound <- 10
        test.points <- seq(x-bound, x+bound, .01)
        if(source.contingent.kernel(min(test.points))>.00000001 |
source.contingent.kernel(max(test.points))>.00000001){
                die("non-bounded contingent kernel")
        }

        cs <- sapply(test.points-x, source.contingent.kernel)
        ds <- sapply(test.points, density)

        ncs <- cs*ds
       ncs <- ncs/integrate(approxfun(test.points, ncs, yleft=0, yright=0), -bound, bound,
subdivisions=10000)$value

        ncs <- a*ncs + (1-a)*cs

        return(approxfun(test.points-x, ncs, yleft = 0, yright = 0))
}

numeric.contingent.density <- function(xs, cks){
        require(pbapply)

        mixture.fn <- function(x){
                s <- 0
                for(i in 1:length(xs)){
                        f<-cks[[i]]
                        s <- s + f(x-xs[i])
                }
                return(s/length(xs))
        }

        return(mixture.fn)
}

numeric.kernel <- function(kernel.function, contingency.distribution, h, range = seq(-
1,1,0.1)){
        kernel.function.h <- function(x){
                return(kernel.function(x/h))
        }

        ks <- sapply(range, kernel.function.h)
        cs <- sapply(range, contingency.distribution)

        #plot(range,ks)
        #plot(range,cs)

        y <- convolve(ks, cs, type="open")/h # should now be normalized

        range.step <- range[2]-range[1]
        output.length <- length(y)
        y <- y*range.step

        output.range <- seq(-(output.length-1)/2*range.step, (output.length-
1)/2*range.step, range.step)
        fn <- approxfun(x = output.range, y = y, yleft = 0, yright = 0)
        #print(y)
        #plot(output.range, y)
        #print(integrate(fn, lower = min(output.range), upper = max(output.range)))

        return(fn)
}

test.guassian <- function(x){
        return(dnorm(x))
}

test.uniform <- function(x, r=5){
        if(x <= r & x >= -r){
                return(1/(2*r))
        }
        return(0)
}

create.kernel.demo <- function(l, bw, seed){
        set.seed(seed)
        x<-rmult(l)
        plot(density(x, bw=bw), lwd=3, main="",xlab="",ylab="", xlim=c(-10,20),
ylim=c(0,.15), yaxt="n")
        h<-g.kernel(0)*.05
        rug(x, lty=2, ticksize=.65, lwd=1.8, col="gray50")
        r <- seq(-15, 25, .1)
        for(i in x){
                lines(r, g.kernel((r-i)/bw)/h*.005, col="gray50", lwd=1.8)
        }
        lines(density(x, bw=bw, from=-15, to=25), lwd=3)
}

rmult <- function(l){
        res<-c()
        for(i in 1:l){
                r <- runif(1)
                if(r>.5){
                        res<-c(res, rnorm(1, 0, 1))
                }else{
                        res<-c(res, rnorm(1, 10, 4))
                }
        }
        return(res)
}

g.kernel <- function(x){
        return(1/sqrt(2*pi)*exp(-x^2/2))
}


opt.tester.1D <- function(reps=10, points = c(5,10,25,50,100,250,500,1000)){
        set.seed(100)
        sim.points <- c()
        sim.err <- c()
        sim.types <- c()

        naive.errs <- c()
        contingent.errs <- c()

        for(p in points){
                print(paste("Points", p))
                contingent.err <- c()
                naive.err <- c()
                for(r in 1:reps){
                        print(paste("Rep", r))
                        sim.points <- c(sim.points, p)
                        sim.points <- c(sim.points, p)

                        test.mean.1 <<- runif(1,-5,5)
                        test.sd.1 <<- runif(1,.5,3)
                        test.mean.2 <<- runif(1,-5,5)
                        test.sd.2 <<- runif(1,.5,3)
                        test.mean.3 <<- runif(1,-5,5)
                        test.sd.3 <<- runif(1,.5,3)

                        points <- create.filter.points(p*3)

                        print("Contingent")
                        c.opt <- optimize.h.1D(reference.1D, contingent.points(points),
C.gaussian.uniform)
                        contingent.err <- c(contingent.err, c.opt$err)
                        sim.types <- c(sim.types,"Contingent Kernel")
                        sim.err <- c(sim.err, c.opt$err)

                        print("Naive")
                        n.opt <- optimize.h.1D(reference.1D, naive.points(points),
C.gaussian.uniform)
                        naive.err <- c(naive.err, n.opt$err)
                        sim.types <- c(sim.types, "Standard Kernel")
                        sim.err <- c(sim.err, n.opt$err)

                }
                naive.errs <- rbind(naive.errs, naive.err)
                contingent.errs <- rbind(contingent.errs, contingent.err)
        }

        return(data.frame(Points=sim.points, "MISE"=sim.err, "Kernel Type"=sim.types))
}

plot.comparisons <- function(reference, naive, contingent, limits = c(-10,10)){
        p <- seq(limits[1],limits[2], .05)

        rp <- reference(p)
        np <- naive(p)
        cp <- contingent(p)

        range <- range(rp,cp,np)

        plot(p, rp, ylim=range, type="l", xlab="", ylab="Density", lwd=1.8)
        lines(p, np, lty=2, lwd=2, col=2)
        lines(p, cp, lty=3, lwd=2, col=3)
        legend("topright", "(x,y)", c("Reference", "Standard", "Contingent"),
lwd=c(1.8,2,2), lty=1:3, col=1:3, title="", bty="n", inset=range*.0)

        plot(p, (np-rp)^2, lty=2, ylim=range((np-rp)^2,(cp-rp)^2), type="l", xlab="",
ylab="Squared Error", lwd=2, col=2)
        lines(p, (cp-rp)^2, lty=3, lwd=2, col=3)
        legend("topright", "(x,y)", c("Standard", "Contingent"), lwd=1.8, lty=2:3,
title="", bty="n", inset=range*.0, col=2:3)
}

create.filter.points <- function(n=900){
        p1 <- filter.points(rreference(n/3, 1), .5)
        p2 <- filter.points(rreference(n/3, 2), 1)
        p3 <- filter.points(rreference(n/3, 3), 2)

        return(list(x=list(p1$x[[1]],p2$x[[1]],p3$x[[1]]),
n=list(p1$n[[1]],p2$n[[1]],p3$n[[1]]), resolution=c(.5,1,2)))
}

optimize.h.1D <- function(reference, observations, kernel, lower=-100, upper=100,
interval=c(0,10), scale=1){

        print("Optimizing h")
        opt.fn <- function(h){
                dist <- contingent.density(observations, kernel, h)
                x <- ISE.1D(reference, dist, lower, upper, scale = scale)
                print(paste("h:", h, "ISE:",x))
                return(x)
        }
        opt <- optimize(opt.fn, interval=interval, tol=(interval[2] - interval[1])/10000)
        return(list(h=opt$minimum, err=opt$objective))
```

```
}

test.mean.1 <- -4
test.sd.1 <- 2
test.mean.2 <- 3
test.sd.2 <- 1
test.mean.3 <- 0
test.sd.3 <- .75

test.type <- "norm"#norm - normal, u - uniform, w - weibull
reference.1D <- function(x, facet = NULL){
        if(test.type=="w"){
                return((dweibull(x+3, scale=1, shape=2)+ dweibull(x, scale=3, shape=1)+
dweibull(x, scale=4, shape=1.5))/3) #gamma
        }
        if(test.type=="u"){
                return((dunif(x, min=-4, max=2)+dunif(x, min=1, max=3)+dunif(x, min=0,
max=.75))/3) #log normal
        }
        if(test.type=="norm"){
                return((dnorm(x, mean= test.mean.1, sd= test.sd.1)+dnorm(x, mean=
test.mean.2, sd= test.sd.2)+dnorm(x, mean= test.mean.3, sd= test.sd.3))/3) #normal
        }
}

rreference <- function(npoints, facet=NULL){
        return(sapply(1:npoints, function(x) r.reference.1D(facet)))
}

r.reference.1D <- function(facet = NULL){
        if(is.null(facet)){
                facet=ceiling(runif(1)*3)
        }

        if(test.type=="norm"){
        if(facet==3){
                return(rnorm(1, mean = test.mean.1, sd = test.sd.1))
        }
        if(facet==2){
                return(rnorm(1, mean = test.mean.2, sd = test.sd.2))
        }
        if(facet==1){
                return(rnorm(1, mean = test.mean.3, sd = test.sd.3))
        }
        }
        if(test.type=="u"){
        if(facet==3){
                return(runif(1, min=-4, max=2))
        }
        if(facet==2){
                return(runif(1, min=1,  max=3))
        }
        if(facet==1){
                return(runif(1, min=0, max=.75))
        }
        }
        if(test.type=="w"){
        if(facet==3){
                return(rweibull(x, scale=1, shape=2)-3)
        }
        if(facet==2){
                return(rweibull(x, scale=3, shape=1))
        }
        if(facet==1){
                return(rweibull(x, scale=4, shape=1.5))
        }
        }
}

ISE.1D <- function(dist1, dist2, lower = -100, upper = 100, scale = 1){
        require(cubature) # for adaptIntegrate, multidimimensional integration
        wrapper <- function(l){
                #loc <- data.frame(x=l[1], y=l[2])
                return((dist1(l)*scale-dist2(l))^2)
        }
        adaptIntegrate(wrapper, lower, upper)$integral
}

area.under.1D <- function(dist, lower = -100, upper = 100){
        require(cubature) # for adaptIntegrate, multidimimensional integration
        wrapper <- function(l){
                return(dist(l))
        }
        x<-adaptIntegrate(wrapper, lower, upper)
        return(x$integral)
}

filter.points <- function(points, resolution=c(.5, 1, 2, 4)){
        npoints <- length(points)
        nbreak <- npoints/length(resolution)
        break.points<-list()
        filtered.count<-list()
        filtered.locs<-list()
        for(i in 1:length(resolution)){
                break.points[[i]] <- points[((i-1)*nbreak+1):(i*nbreak)];
                step <- resolution[i];
                h <- hist(break.points[[i]], plot=FALSE, breaks=seq(step*-
1000,step*1000,step));
                filtered.count[[i]] <- h$counts[which(h$count != 0)];
                filtered.locs[[i]] <- h$mid[which(h$count != 0)];
        }

        return(list(x=filtered.locs, n=filtered.count, resolution=resolution))
}

naive.points <- function(filtered, exclude=NULL){
        res <- c();
        for(i in 1:length(filtered$x)){
                if(is.null(exclude) || (! filtered$resolution[i] %in% exclude)){
                for( k in 1:length(filtered$x[[i]])){
                        res <- c(res, rep(filtered$x[[i]][k], filtered$n[[i]][k]));
                }
                }
        }
        return(cbind(rep(0,length(res)),res));
}

contingent.points <- function(filtered){
        res <- c();
        r <- c();
        for(i in 1:length(filtered$x)){
                for( k in 1:length(filtered$x[[i]])){
                        res <- c(res, rep(filtered$x[[i]][k], filtered$n[[i]][k]));
                        r <- c(r, rep(filtered$resolution[i], filtered$n[[i]][k])/2);
                }
        }
        return(cbind(r, res));
}



dist.tester <- function(n=10000){
        orig.points <- rnorm(n, mean = 0, sd = 10)
        error.points <- orig.points + rnorm(n, mean = 0, sd = 10)

        return(list(orig=orig.points, error=error.points))
}

within.range <- function(p){
        sapply(p, function(x) (x<30.8 && x>30.2))
}
```

```
plot.estimates <- function(observations, resolution = 240){
        h_ref = 2.896723
        h = 0.8
        h=h_ref
        data = cbind(observations$radius.latitude, observations$longitude,
observations$latitude)

        require(compiler)
        enableJIT(3)
        f.contingent <- contingent.density(data, C.gaussian.uniform, h)
        f.naive <- contingent.density(cbind(rep(0,nrow(data)), data[,2:3]),
C.gaussian.uniform, h)
        g.contingent <- make.grid(f.contingent, xlim = c(-127, -65), ylim = c(20, 55),
resolution = resolution)
        g.naive <- make.grid(f.naive, xlim = c(-127, -65), ylim = c(20, 55), resolution =
resolution)
        enableJIT(0)
        image(g.contingent$x, g.contingent$y,(g.contingent$z)^.2, xlim = c(-126, -66),
ylim = c(25, 50), xlab="Longitude", ylab="Latitude", asp=1); require(maps); map("world",
add=TRUE);map("state", add=TRUE); contour(g.contingent, add=TRUE, col="brown", drawlabels
= FALSE);

        image(g.naive$x,g.naive$y,(g.naive$z)^.2, xlim = c(-126, -66), ylim = c(25, 50),
xlab="Longitude", ylab="Latitude", asp=1); require(maps);map("world",
add=TRUE);map("state", add=TRUE); contour(g.naive, add=TRUE, col="brown", drawlabels =
FALSE);
}



# observations <- read.delim("~/Dropbox/Berkeley/Contingent Kernels/Twitter Locations
10,000.txt");
# observations <- observations[(observations$longitude > -130) * (observations$longitude
< -60) * (observations$latitude > 19) * (observations$latitude < 57)==1,]
# observations[,"radius.latitude"] <- observations[,"radius"]/111200 #convert from meters
to degrees
# observations[,"radius.longitude"] <- observations[,"radius"]/111200 #*
cos(observations$latitude*pi/180) #convert from meters to degrees
# plot.discs(observations)

plot.discs <- function(observations){
        require(maps)
        require(plotrix)
        locs <-
cbind( jitter(observations[,"longitude"],amount=0.2),jitter(observations[,"latitude"],amo
unt=0.2),observations[,"radius.longitude"], observations[,"radius.latitude"])

        #asp 1.3 would look better but draw.circle doesn't seem to adjust
        plot.header<-function(){
                plot(NULL,  xlim = c(-124, -66.95), ylim = c(25, 50), asp=1,
xlab="Longitude", ylab="Latitude");
                map('world', add=TRUE);
                map('usa', add=TRUE, fill=TRUE, col="grey93");
                map('state', add=TRUE);
        }
        col = "#0000A022";
        plot.header()
        points(locs[,1:2], col=col, pch=19, cex=.3)
        plot.header()

        for(i in 1:nrow(locs)){
                disc <- ellipsePoints(locs[i,3], locs[i,4], loc=c(locs[i,1:2]),n=100)

                #return(disc)
                polygon(disc, border=col)
                # draw.circle(locs[i,2], locs[i,3], locs[i,1], border=col);
        }
}

ellipsePoints <- function(a,b, alpha = 0, loc = c(0,0), n = 201)
```

```
{
    ## Purpose: ellipse points,radially equispaced, given geometric par.s
    ## -------------------------------------------------------------------------
    ## Arguments: a, b : length of half axes in (x,y) direction
    ##            alpha: angle (in degrees) for rotation
    ##            loc  : center of ellipse
    ##            n    : number of points
    ## -------------------------------------------------------------------------
    ## Author: Martin Maechler, Date: 19 Mar 2002, 16:26

    B <- min(a,b)
    A <- max(a,b)
    ## B <= A
    d2 <- (A-B)*(A+B)                   #= A^2 - B^2
    phi <- 2*pi*seq(0,1, len = n)
    sp <- sin(phi)
    cp <- cos(phi)
    r <- a*b / sqrt(B^2 + d2 * sp^2)
    xy <- r * cbind(cp, sp)
    ## xy are the ellipse points for alpha = 0 and loc = (0,0)
    al <- alpha * pi/180
    ca <- cos(al)
    sa <- sin(al)
    xy %*% rbind(c(ca, sa), c(-sa, ca)) + cbind(rep(loc[1],n),
                                                rep(loc[2],n))
}


# observations <- data.frame(r=c(1,2,.3,4,2.5), x=c(-1,1,1,2,3))
# x <- contingent.density(observations, C.gaussian.uniform, 0.1)
# plot(p<-seq(-5,10,.05), x(p), type="l")

contingent.density <- function(data, K, h){
        require(pbapply)
        data <- as.data.frame(data)
        mixture.fn <- function(x){
                res <- apply(as.data.frame(x), 1, function(xc){
                                return(K(xc,h,data));
                        }
                );
                return(res);
        }
        return(mixture.fn)
}

# Gaussian Kernel with Uniform PDF
# First column is radius, the rest of the columns are the point locations
# Currently only has scaling for univariate case

erf <- function(x) {2 * pnorm(x * sqrt(2)) - 1};

C.gaussian.uniform <- function(test.point, h, observation.points){
        r <- observation.points[,1]

        dists <- c()
        if(length(test.point)==2){
                dists <- sqrt((observation.points[,2]-
test.point[1])^2+(observation.points[,3]-test.point[2])^2)
        }else if(length(test.point)==1){
                dists <- abs(observation.points[,2]-test.point[1])
        }else{
                stop("Kernel only defined for 1- and 2-dimensional space.")
        }

        v <- (erf((r-dists)/(h*sqrt(2))) + erf((r+dists)/(h*sqrt(2))))/(4*r)

        v[r==0] <- dnorm(dists[r==0], 0, h)

        return(sum(v)/length(r))
}
```

```
# Gaussian Kernel with Gaussian PDF of a given SD
# First column is the SD, the rest of the columns are the point locations
# Currently only has scaling for bivariate case

C.gaussian.gaussian <- function(test.point, h, observation.points){
        sigmas <- observation.points[,1]
        dists <- c()
        if(length(test.point)==2){
                dists <- sqrt((observation.points[,2]-
test.point[1])^2+(observation.points[,3]-test.point[2])^2)
        }else if(length(test.point)==1){
                dists <- abs(observation.points[,2]-test.point[1])
        }else{
                stop("Kernel only defined for 1- and 2-dimensional space.")
        }

        v <- (1/((2*pi)*(h^2+sigmas^2))*exp(-dists^2/(2*(h^2+sigmas^2))))
        return(sum(v)/length(sigmas))
}


# observations <- data.frame(r=c(0,.1,.2,.3,0), x=c(-1,1,-2,2,3), y=c(1,3,4,2,-3))
# x <- contingent.density(observations, C.gaussian.uniform, 0.5)
# g <- make.grid(x, xlim = c(-5, 5), ylim = c(-5,5), resolution = 40)
# persp(g)

make.grid <- function(f, xlim = c(-100,100), ylim = c(-100,100), resolution = 40){
        xs<-seq(xlim[1], xlim[2], length.out=resolution)
        ys<-seq(ylim[1], ylim[2], length.out=resolution)

        cx <- rep(xs, resolution)
        cy <- rep(ys, each=resolution)
        ps <- cbind(cx, cy)
        zs <- f(ps)
        zs <- matrix(zs, ncol=resolution)
        return(list(x=xs, y=ys, z=zs))
}

# plot.kernel(C.gaussian.uniform, c(.05,.15, .45), data.frame(x=c(0.5),r=c( 0)), 1.1)

plot.kernel <- function(K, h, observation, range = 1){
        savepar<-par(mar=c(4, 4, 1.5, 1.5) + 0.1)
        results <- list()
        labels <- c()
        points <- seq(-range, range, range*2/200)

        count <- 0;
        for(myh in h){
                count <- count+1
                labels <- c(labels,  myh)
                results[[count]] <- sapply(points, function(p) K(p, myh, observation))
        }

        max.y <-max(sapply(results, max))*1.1

        lwd=2
        plot(points, results[[1]], xlim=c(-range, range), ylim=c(0,max.y), type="l",
ylab="Density", xlab="x", lwd=lwd)
        for(i in 2:count){
                lines(points, results[[i]], lty=i, lwd=lwd)
        }

        legend("topright", "(x,y)", labels, lwd=lwd, lty=1:count,
title=expression(italic("h")), bty="n", inset=range*.04)
        par(savepar)
}

# plot.observations(observations, uniform.plot, c(-5,10))

plot.observations <- function(observations, f.plot, range, add=FALSE, ...){
        max.y <- max(apply(observations, 1, uniform.range))
        if(! add){
                plot(NA, ylim=c(0, max.y), xlim=range)
        }
        x<-apply(observations, 1, uniform.plot)
}

uniform.plot <- function(observation, ...){
        x = observation[1]
        r = observation[2]

        h <- 1/(2*r)
        lines(c(-99999999999999, x-r, x-r, x+r, x+r, 99999999999999), c(0, 0, h, h, 0,
0),  ...)
}

uniform.range <- function(observation){
        x= observation[1]
        r= observation[2]

        h <- 1/(2*r)

        return(h);
}

# Returns the Mean Integrated Squared Error between two distributions

MISE <- function(dist1, dist2, lower = rep(-100,2), upper = rep(100,2), scale=1){
        require(cubature) # for adaptIntegrate, multidimimensional integration
        wrapper <- function(l){
                loc <- data.frame(x=l[1], y=l[2])
                return((dist1(loc)*scale-dist2(loc))^2)
        }
        adaptIntegrate(wrapper, lower, upper)$integral
}

# Returns the area under a distribution

area.under <- function(dist, lower = rep(-100,2), upper = rep(100,2)){
        require(cubature) # for adaptIntegrate, multidimimensional integration
        wrapper <- function(l){
                loc <- data.frame(x=l[1], y=l[2])
                return(dist(loc))
        }
        x<-adaptIntegrate(wrapper, lower, upper)
        #print(x)
        return(x$integral)
}

within <- function(root, error, sample){
        if((sample < root*(1+error)) && (sample > root*(1-error))){
                return(TRUE);
        }else{
                return(FALSE);
        }
}


optimize.h <- function(reference, observations, kernel, lower, upper, interval=c(0,10),
scale=1){
        print("Optimizing h")
        opt.fn <- function(h){
                dist <- contingent.density(observations, kernel, h)
                x <- MISE(reference, dist, lower, upper, scale = scale)
                print(paste("h:", h, "MISE:",x))
                return(x)
        }
        opt <- optimize(opt.fn, interval=interval, tol=(interval[2] - interval[1])/10000)
        return(list(h=opt$minimum, err=opt$objective))
}

# results <- lapply(c(.1,1,10), function(sd) MC.opt.tester(sd))
```

```
MC.opt.tester <- function(errors.sd, dist.sd = 5, points = 500, n = 5, lower = rep(-
200,2), upper = rep(200,2)){
        library(adehabitat)

        naive.errors <- c()
        contingent.errors <- c()

        naive.hs <- c()
        contingent.hs <- c()

        ref.hs <- c()
        LSCV.hs <- c()

        error.contingent.LSCV <- c()
        error.contingent.ref <- c()
        error.naive.LSCV <- c()
        error.naive.ref <- c()

        reference <- function(x){dnorm(sqrt(x[1,"x"]^2+x[1,"y"]^2),
sd=dist.sd)/(sqrt(2*pi))}
        volume <- area.under(reference)
        print(paste("Volume:", volume))

        for(i in 1:n){
                print(paste("Iteration", i, "of", n))

                ps <- data.frame("x"=rep(0, points), "y"=rep(0, points))
                ps <- perturb.points(ps, rep(dist.sd, points))
                #print(ps)
                standard.deviations <- sapply(rnorm(points, errors.sd, errors.sd*10),
function(x) max(x,0))
                #print(standard.deviations)
                #print(standard.deviations)
                ps <- perturb.points(ps, standard.deviations)
                #print(ps)
                #plot(ps);
                #return(ps)

                hr<-kernelUD(ps)
                ref.hs <- c(ref.hs, hr[[1]]$h)
                print(paste("h_ref:", ref.hs[length(ref.hs)]))
                hr<-kernelUD(ps, h="LSCV")
                LSCV.hs <- c(LSCV.hs, hr[[1]]$h$h)
                print(paste("h_LSCV:",LSCV.hs[length(ref.hs)]))

                naive.kernel <- contingent.density(cbind(rep(0, points), ps),
C.gaussian.gaussian, ref.hs[length(ref.hs)])
                contingent.kernel <- contingent.density(cbind(standard.deviations, ps),
C.gaussian.gaussian, ref.hs[length(ref.hs)])

                error.naive.ref <- c(error.naive.ref, MISE(reference, naive.kernel, lower,
upper, scale = 1/volume))
                error.contingent.ref <- c(error.contingent.ref, MISE(reference,
contingent.kernel, lower, upper, scale = 1/volume))

                naive.kernel <- contingent.density(cbind(rep(0, points), ps),
C.gaussian.gaussian, LSCV.hs[length(LSCV.hs)])
                contingent.kernel <- contingent.density(cbind(standard.deviations, ps),
C.gaussian.gaussian, LSCV.hs[length(LSCV.hs)])

                error.naive.LSCV <- c(error.naive.LSCV, MISE(reference, naive.kernel,
lower, upper, scale = 1/volume))
                error.contingent.LSCV <- c(error.contingent.LSCV, MISE(reference,
contingent.kernel, lower, upper, scale = 1/volume))

                n.opt <- optimize.h(reference, cbind(rep(0, points), ps),
C.gaussian.gaussian, lower, upper, scale = 1/volume)
                c.opt <- optimize.h(reference, cbind(standard.deviations, ps),
C.gaussian.gaussian, lower, upper, scale = 1/volume)
```

```
                naive.errors <- c(naive.errors, n.opt$err)
                contingent.errors <- c(contingent.errors, c.opt$err)
                naive.hs <- c(naive.hs, n.opt$h)
                contingent.hs <- c(contingent.hs, c.opt$h)

                print(naive.errors[length(naive.errors)])
                print(contingent.errors[length(contingent.errors)])
                print(naive.hs[length(naive.hs)])
                print(contingent.hs[length(contingent.hs)])
        }
        print(paste("Standard Error Mean:", mean(naive.errors)))
        print(paste("Contingent Error Mean:", mean(contingent.errors)))
        return(data.frame(Standard.Kernel = naive.errors, Contingent.Kernel =
contingent.errors, Standard.h = naive.hs, Contingent.h = contingent.hs, Ref.h=ref.hs,
LSCV.h = LSCV.hs, error.naive.ref = error.naive.ref, error.contingent.ref =
error.contingent.ref, error.naive.LSCV = error.naive.LSCV, error.contingent.LSCV =
error.contingent.LSCV))
}


# Error Tester - Theoretical Gaussian
# results <- lapply(c(1,4,8), function(sd) MC.tester(sd))

MC.tester <- function(errors.sd, dist.sd = 5, points = 5, n = 10, lower = rep(-100,2),
upper = rep(100,2)){
        set.seed(99)
        naive.errors <- c()
        contingent.errors <- c()

        reference <- function(x){dnorm(sqrt(x[1,"x"]^2+x[1,"y"]^2),
sd=dist.sd)/(sqrt(2*pi))}
        volume <- area.under(reference)
        print(paste("Volume:", volume))

        for(i in 1:n){
                print(paste("Iteration", i, "of", n))

                ps <- data.frame("x" = rep(0, points), "y" = rep(0, points))
                ps <- perturb.points(ps, rep(dist.sd, points))

                standard.deviations <- sapply(rnorm(points, errors.sd, errors.sd/4),
function(x) max(x,0))
                ps <- perturb.points(ps, standard.deviations)

                h <- sqrt(bw.nrd(ps[,"x"])^2+bw.nrd(ps[,"y"])^2)

                naive.kernel <- contingent.density(cbind(rep(0, points), ps),
C.gaussian.gaussian, h)
                cont.kernel <- contingent.density(cbind(standard.deviations, ps),
C.gaussian.gaussian, h)

                ne <- MISE(reference, naive.kernel, lower, upper, scale = 1/volume)
                ce <- MISE(reference, cont.kernel, lower, upper, scale = 1/volume)


                print(ne)
                print(ce)

                if(within(1, 0.01, area.under(naive.kernel)) && within(1, 0.01,
area.under(cont.kernel))){
                        naive.errors <- c(naive.errors, ne)
                        contingent.errors <- c(contingent.errors, ce)

                        if(ne>1 || ce>1){
                                return(list(naive= naive.kernel, cont= cont.kernel,
reference=reference, ps = ps, sds= standard.deviations))
                        }
                }else{
                        print("Volume under curves is not 1")
                        #return(list(naive= naive.kernel, cont= cont.kernel,
reference=reference, ps = ps, sds= standard.deviations))
```

```
                }
        }
        print(paste("Standard Error Mean:", mean(naive.errors)))
        print(paste("contingent Error Mean:", mean(contingent.errors)))

        return(data.frame(Standard.Kernel = naive.errors, contingent.Kernel =
contingent.errors))
}


# Error Tester
# ps <- data.frame(x=c(1, 5, 3, 5, 6, 7, 8, 1, 0, 10), y=c(3, 5, 1, 6, 7, 2, 4, 6, 7, 9))
# results <- lapply(c(1,2,4,8), function(sd) MC.tester.empirical(ps, sd))
# MC.tester.empirical(ps, 5)

MC.tester.empirical <- function(points, errors.sd, h =
sqrt(bw.nrd(points[,"x"])^2+bw.nrd(points[,"y"])^2)/2, lower = rep(-100,2), upper =
rep(100,2), sims = 60){
        set.seed(101)

        h=sqrt(bw.nrd(ps[,"x"])^2+bw.nrd(ps[,"y"])^2)/2#0.1
        print(paste("h:", h))

        naive.errors <- c()
        contingent.errors <- c()

        npoints <- nrow(points)
        reference <- contingent.density(cbind(rep(0, npoints), points),
C.gaussian.gaussian, h)

        #persp(make.grid(reference, xlim=c(-20, 20), ylim=c(-20,20), resolution=50))

        for(i in 1:sims){
                print(paste("Iteration", i, "of", sims))

                standard.deviations <- sapply(rnorm(npoints, errors.sd, errors.sd),
function(x) max(x,0))
                #print(standard.deviations)

                ps <- perturb.points(points, standard.deviations)
                #plot(ps)

                h=sqrt(bw.nrd(ps[,"x"])^2+bw.nrd(ps[,"y"])^2)/2#0.1
                #print(paste("h:", h))

                naive.kernel <- contingent.density(cbind(rep(0, npoints), ps),
C.gaussian.gaussian, h)
                cont.kernel <- contingent.density(cbind(standard.deviations, ps),
C.gaussian.gaussian, h)

                #return(list(orig=points, ps=ps, sd=standard.deviations, cont=cont.kernel,
naive=naive.kernel, reference=reference))

                #print(cbind(rep(0, npoints), ps))
                #print(cbind(standard.deviations, ps))
                #persp(make.grid(naive.kernel, xlim=c(-20, 20), ylim=c(-20,20),
resolution=50))
                #persp(make.grid(cont.kernel, xlim=c(-20, 20), ylim=c(-20,20),
resolution=50))
                #print(area.under(naive.kernel))
                #print(area.under(cont.kernel))

                #stop();

                naive.errors <- c(naive.errors, MISE(reference, naive.kernel, lower,
upper))
                contingent.errors <- c(contingent.errors, MISE(reference, cont.kernel,
lower, upper))

                print(naive.errors[length(naive.errors)])
                print(contingent.errors[length(contingent.errors)])
        }
        print(paste("Standard Error Mean:", mean(naive.errors)))
        print(paste("Contingent Error Mean:", mean(contingent.errors)))
        return(data.frame(Standard.Kernel = naive.errors, Contingent.Kernel =
contingent.errors))
}

perturb.points <- function(points, standard.deviations){
        dists <- rnorm(nrow(points), 0, standard.deviations)
        angles <- runif(nrow(points))*2*pi

        new.points <- data.frame(x=points[,1]+cos(angles)*dists,
y=points[,2]+sin(angles)*dists)

        return(new.points)
}
```

Twitter Data

The following implements an algorithm to pull live tweet data from Twitter and prepare it for analysis in a database. The code is implemented in PHP.

```
#!/usr/bin/php -q
<?php
require_once "System/Daemon.php";
// Bare minimum setup

// Setup
$options = array(
    'appName' => 'twitter',
    'appDir' => dirname(__FILE__),
    'appDescription' => 'Twitter downloader to mysql',
    'authorName' => 'Scott',
    'authorEmail' => 'scottfr@gmail.com',
    'sysMaxExecutionTime' => '0',
    'sysMaxInputTime' => '0',
    'sysMemoryLimit' => '1024M',
    'appRunAsGID' => 1000,
    'appRunAsUID' => 1000,
);

System_Daemon::setOptions($options);

echo(System_Daemon::writeAutoRun());

// Spawn Deamon!
System_Daemon::start();


        include_once('config.php');
        $opts = array(
                'http'=>array(
                        'method'       =>      "POST",
                        'content'      =>      'track='.WORDS_TO_TRACK,
                )
        );
        //We're going to store the data in the database, so, let's open a connection:
        $db = mysql_connect('localhost', 'twitter_alerts', 'somepasword');
        mysql_select_db('twitter_alerts', $db);

        $context = stream_context_create($opts);
        while (1){
                $instream =
fopen('http://'.TWITTER_USERNAME.':'.TWITTER_PASSWORD.'@stream.twitter.com/1/statuses/fil
ter.json','r' ,false, $context);
                while(! feof($instream)) {
                        if(! ($line = stream_get_line($instream, 20000, "\n"))) {
                                continue;
                        }else{
                                $tweet = json_decode($line);
                                //Clean the inputs before storing
                                $id = $tweet->{'id_str'};
                                if(! isset($id)){
                                        $id=0;
                                }
                                $text = mysql_real_escape_string($tweet->{'text'});
                                $screen_name = mysql_real_escape_string($tweet->{'user'}-
>{'screen_name'});
                                $followers_count = mysql_real_escape_string($tweet-
>{'user'}->{'followers_count'});
                                $statuses_count = mysql_real_escape_string($tweet-
>{'user'}->{'statuses_count'});
                                $friends_count = mysql_real_escape_string($tweet->{'user'}-
>{'friends_count'});
                                $line=mysql_real_escape_string($line);
                                #print($id."\n");
                                $q="INSERT INTO tweets
(id ,text ,screen_name ,followers_count, created_at, statuses_count,friends_count,
full_text) VALUES ($id, '$text', '$screen_name', '$followers_count', NOW(),
'$statuses_count', '$friends_count', '$line')";
                                #print($q."\n\n");
                                //We store the new post in the database, to be able to read
it later
                                $ok = mysql_query($q);
                                if (!$ok) {echo "Mysql Error: ".mysql_error();}
                                flush();
                        }
                }
        }
System_Daemon::log(System_Daemon::LOG_INFO, "Stopping");
// Stop daemon!
System_Daemon::stop();
?>
<?php
require_once('double_metaphone_func.php');
require_once('genderfromname.php');

date_default_timezone_set('America/Los_Angeles');

$pool = mysql_connect('localhost', 'root', 'quicktime1', true);
mysql_select_db('twitter_alerts', $pool) || die(mysql_error());

$database = mysql_connect('localhost', 'root', 'quicktime1', true);
mysql_select_db('twitter', $database) || die(mysql_error());

$result = mysql_fetch_array(mysql_query("SELECT max(twitter_id) FROM Tweets",
$database));
if(isset($result[0])){
        $result = mysql_fetch_array(mysql_query("SELECT mid FROM Tweets WHERE id =
".$result[0], $pool));
}
if(! isset($result[0])){
        $result[0] = -1;
}

print("Starting at ".$result[0]."\n\n");

$result = mysql_unbuffered_query("SELECT id, full_text FROM Tweets WHERE mid >
".$result[0], $pool);
if($result==FALSE){
        die("Mysql(d) Error: ".mysql_error());
}

$words                                              = array();
$words["word_flu"]                          = array("flu", "influenza", "influensa",
"h1n1", "h2n3 ", "h3n2");
$words["word_major"]                 = array("autism", "hiv", "aids", "cancer");
```

```
$words["word_cold"]                   = array("cough", "sniffle", "sneeze", "headache",
"fever", "runny nose", "sore throat", "chills");
$words["word_constipation"]   = array("constipation", "constipated");
$words["word_diarrhea"]               = array("diarrhea", "diaherra", "diarhea");
$words["word_injury"]                 = array("sprain", "bleeding", "infection",
"infected", "rash");
$words["word_general"]                        = array("sick");
$words["word_allergies"]              = array("allergies");
$words["word_stomach"]                        = array("upset stomach", "stomachache");
$words["word_sleep"]                  = array("apnea", "insomnia");
$words["word_control"]                        = array("facing", "stated");
$key_words = array_keys($words);

$i = 0;
while($row = mysql_fetch_array($result)){
        if(($i % 1000) == 0){
                print($i."\n");
        }
        $i = $i+1;

        #print($row["id"]."\n");
        if($row["id"] != 0){
                $tweet = json_decode($row['full_text']);
                $ud = array();
                $ud["twitter_id"] = $tweet->{'user'}->{'id'};
                $uid = userID($ud["twitter_id"], $database);
                if($uid == -1){
                        $ud["screen_name"] = $tweet->{'user'}->{'screen_name'};
                        $ud["name"] = $tweet->{'user'}->{'name'};
                        $ud["male"] = getGender($ud["name"]);
                        if($ud["male"] == "m"){
                                $ud["male"] = true;
                        }else if($ud["male"] == "f"){
                                $ud["male"] = false;
                        }
                        $ud["description"] = $tweet->{'user'}->{'description'};
                        $ud["statuses_count"] = $tweet->{'user'}->{'statuses_count'};
                        $ud["friends_count"] = $tweet->{'user'}->{'friends_count'};
                        $ud["followers_count"] = $tweet->{'user'}->{'followers_count'};
                        $ud["lang"] = $tweet->{'user'}->{'lang'};
                        $ud["time_zone"] = $tweet->{'user'}->{'time_zone'};
                        $ud["location"] = $tweet->{'user'}->{'location'};
                        $ud["listed_count"] = $tweet->{'user'}->{'listed_count'};
                        $ud["verified"] = $tweet->{'user'}->{'verified'};
                        $ud["created_at"] =  mysqlDate($tweet->{'user'}->{'created_at'});
                        $ud["url"] = $tweet->{'user'}->{'url'};
                        $ud["geo_enabled"] = $tweet->{'user'}->{'geo_enabled'};
                        $ud["verified"] = $tweet->{'user'}->{'verified'};
                        $ud["profile_sidebar_fill_color"] = hexdec($tweet->{'user'}-
>{'profile_sidebar_fill_color'});
                        $ud["profile_text_color"] = hexdec($tweet->{'user'}-
>{'profile_text_color'});
                        $ud["profile_background_color"] = hexdec($tweet->{'user'}-
>{'profile_background_color'});
                        $ud["profile_background_image_url"] = $tweet->{'user'}-
>{'profile_background_image_url'};
                        $ud["profile_background_tile"] = $tweet->{'user'}-
>{'profile_background_tile'};
                        $ud["default_theme"] = ($ud["profile_background_tile"]==FALSE &&
strtolower($tweet->{'user'}->{'profile_sidebar_fill_color'})=="ddeef6" &&
strtolower($tweet->{'user'}->{'profile_background_color'})=="c0deed" &&
strtolower($tweet->{'user'}->{'profile_text_color'})=="333333" && strtolower($tweet-
>{'user'}->{'profile_link_color'})=="0084b4");

                        $dominantColor = RGB_TO_HSV($tweet->{'user'}-
>{'profile_background_color'});
                        if($dominantColor["h"] == -1){
                                $ud["hue_class"] = -1; #grey color, no hue
                        }else{
                                $ud["hue_class"] = round($dominantColor["h"]/60);
                                if($ud["hue_class"] == 6){
                                        $ud["hue_class"] = 0;
                                }
                        }
                        $ud["hue"] = $dominantColor["h"];
                        $ud["saturation"] = $dominantColor["s"];
                        $ud["brightness"] = $dominantColor["v"];

                        unset($ud["name"]); # do not insert name into database to protect
users privacy

                        mysql_query(mysql_insert_array("Users", $ud), $database)  ||
die("MySQL(a) Error".mysql_error());
                        $uid = mysql_insert_id($database);
                }

                $td = array();
                $td["user_id"] = $uid;
                $td["twitter_id"] = $tweet->{'id'};
                $td["text"] = $tweet->{'text'};
                $td["created_at"] = mysqlDate($tweet->{'created_at'});
                $td["in_reply_to_status_id"] = $tweet->{'in_reply_to_status_id'};
                $td["source"] = $tweet->{'source'};
                $td["retweeted"] = $tweet->{'retweeted'};
                $td["favorited"] = $tweet->{'favorited'};
                $td["geo"] = clean($tweet->{'geo'});
                $td["contributors"] = clean($tweet->{'contributors'});
                $td["hashtags"] = clean($tweet->{'entities'}->{'hashtags'});
                $td["user_mentions"] = clean($tweet->{'entities'}->{'user_mentions'});
                $td["urls"] = clean($tweet->{'entities'}->{'urls'});
                $td["in_reply_to_user_id"] = $tweet->{'in_reply_to_user_id'};

                if(clean($td["geo"])){
                        $latlong = $tweet->{'geo'}->{'coordinates'};
                        $td["latitude"] = $latlong[0];
                        $td["longitude"] = $latlong[1];
                }

                foreach($key_words as $word){
                        $td[$word] = contains_words($td["text"], $words[$word]);
                }
                if($td["word_control"]){
                        foreach($key_words as $word){
                                if($word != "word_control"){
                                        $td["word_control"] = $td["word_control"] && (!
$td[$word]);
                                }
                        }
                }

                mysql_query(mysql_insert_array("Tweets", $td), $database);#  ||
die("MySQL(b) Error".mysql_error());
        }
}

mysql_close($pool);
mysql_close($database);

function getGender($name){
        if(isset($name) && $name != "" && $name!=" "){
                $firstName = explode(" ", $name);
                $firstName = $firstName[0];
                $g = gender($firstName, 2);
                if(isset($g)){
                        return($g["gender"]);
                }
        }
}

function clean($s){
        if(empty($s)){
                return(NULL);
        }
        $s = json_encode($s);
        if($s == "null"){
                return(NULL);
        }else{
                return($s);
        }
}

function userID($id, $db){
        $result = mysql_query("SELECT id FROM Users WHERE twitter_id = ".$id, $db);
        if(! $result){
                die("MySQL(c) Error".mysql_error());
        }
        $num_rows = mysql_num_rows($result);
        if($num_rows > 0){
                $row = mysql_fetch_array($result);
                return($row["id"]);
        }else{
                return(-1);
        }
}

function mysqlDate($date){
        return(date( 'Y-m-d H:i:s', strtotime($date) ));
}

function mysql_insert_array($table, $data) {
        foreach ($data as $field=>$value) {
                $fields[] = '`' . $field . '`';

                if(is_null($value)){
                        $values[] = "NULL";
                }else{
                        $values[] = "'" . mysql_real_escape_string($value) . "'";
                }
        }
        $field_list = join(',', $fields);
        $value_list = join(', ', $values);

        $query = "INSERT INTO `" . $table . "` (" . $field_list . ") VALUES (" .
$value_list . ")";

        return $query;
}


function RGB_TO_HSV ($C)  // RGB Values:Number 0-255
{                                  // HSV Results:Number 0-1
   $r=hexdec(substr($C,0,2));
   $g=hexdec(substr($C,2,2));
   $b=hexdec(substr($C,4,2));


        //Get the min and max
        $min = min(array($r,$g,$b));
        $max = max(array($r,$g,$b));

        //ensure it is within bounds
        if ($max>255) return null;
        if ($min<0) return null;

        //Convert the brightness to a percentage
        $v = $max/255;

        //Get the delta of max and min
        $delta = $max-$min;

        //Is max zero?
        if (($max!=0)&&($delta!=0)) {
            //COmpute the saturation
            $s = $delta/$max;
        }
        else {
            //Get out of here, we have no color
            $s=0;
            $h=-1;
            return array("h"=>$h, "s"=>$s, "v"=>$v);
        }

        //Compute the hue
        if ($r==$max) {
            $h = ($g-$b)/$delta;
        } elseif ($g==$max) {
            $h = 2+($b-$r)/ $delta;
        } else {
            $h = 4+($r-$g)/$delta;
        }

        //To degrees
        $h*=60;
        if ($h<0) $h+=360;



        //Return the results
        return array("h"=>$h, "s"=>$s, "v"=>$v);
}

function contains_words($text, $words){
        foreach($words as $word){
                $pos = stripos($text, $word);
                if(!($pos === false)) {
                        return true;
                }
        }
        return false;
}
?>
```

References for Appendices and Conclusions

Andersena, D., & Richardsona, G. (1997). Scripts for group model building. *System Dynamics Review*, *13*(2), 107–129.

Forrester, J. W., & Senge, P. M. (1979). Tests for building confidence in system dynamics models. *System Dynamics Group, Sloan School of Management*.

McDonald, P., Mohebbi, M., & Slatkin, B. (2012). Comparing Google consumer surveys to existing probability and non-probability based Internet surveys.

McDonnell, G. (2013, December 5). Personal Communication. Oakland, Ca.

Meadows, D. H., & Robinson, J. M. (1985). *The Electronic Oracle*. Albany, NY: System Dynamics Society.

Morecroft, J., & Robinson, S. (2005). Explaining puzzling dynamics: comparing the use of system dynamics and discrete-event simulation. *… Of the System Dynamics Society*.

Overstreet, C. M., Nance, R. E., & Balci, O. (2002). Issues in enhancing model reuse. *… Grand Challenges for Modeling …*.

Paul, R. J., & Taylor, S. J. E. (2002). What use is model reuse: is there a crook at the end of the rainbow? (pp. 1–5). Presented at the Winter Simulation Conference.

Ries, E. (2011). *The Lean Startup*. New York: Crown Business.

Robinson, S., Nance, R. E., Paul, R. J., Pidd, M., & Taylor, S. J. E. (2004). Simulation model reuse: definitions, benefits and obstacles. *Simulation Modelling Practice and Theory*, *12*(7-8), 479–494. doi:10.1016/j.simpat.2003.11.006

Silver, N. (2010, November 10). Which Polls Fared Best (and Worst) in the 2012Presidential Race. *New York Times*, pp. 1–10.

Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society B*, 111–147.