ECONSTOR
Make Your Publications Visible.

A Service of

ZBW Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Ganz, Scott C.

Working Paper

Goldilocks vs. Robin Hood: Using Shape-constrained Regression to Evaluate U-Shaped (Or Inverse U-Shaped) Theories in Data

AEI Economics Working Paper, No. 2022-12

Provided in Cooperation with:
American Enterprise Institute (AEI), Washington, DC

Suggested Citation: Ganz, Scott C. (2022) : Goldilocks vs. Robin Hood: Using Shape-constrained Regression to Evaluate U-Shaped (Or Inverse U-Shaped) Theories in Data, AEI Economics Working Paper, No. 2022-12, American Enterprise Institute (AEI), Washington, DC

This Version is available at:
https://hdl.handle.net/10419/280659

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

Goldilocks vs. Robin Hood: Using Shape-constrained Regression to Evaluate U-Shaped (Or Inverse U-Shaped) Theories in Data

Scott C. Ganz
Georgetown McDonough School of Business
American Enterprise Institute

AEI Economics Working Paper 2022-12
Updated December 2022

© 2022 by Scott C. Ganz. All rights reserved.

The American Enterprise Institute (AEI) is a nonpartisan, nonprofit, 501(c)(3) educational organization and does not take institutional positions on any issues. The views expressed here are those of the author(s).

Goldilocks vs. Robin Hood: Using shape-constrained regression to evaluate u-shaped (or inverse u-shaped) theories in data.

Scott C. Ganz

Georgetown McDonough School of Business and AEI

scott.ganz@georgetown.edu

Theories that predict u-shaped and inverse u-shaped relationships are ubiquitous throughout the social sciences. As a result of this widespread interest in identifying non-monotonic relationships in data and the well-known problems with standard parametric approaches based on quadratic regression models, there has been considerable recent interest in finding new ways to evaluate such theories using alternative approaches. In this paper, I propose a new semi-parametric method for evaluating these theories, which I call the "Goldilocks" algorithm. The algorithm is so named because it involves estimating three models in order to evaluate a u-shaped or inverse u-shaped hypothesis. One model is too flexible ("too hot") because it permits multiple inflection points in the expected relationship between x and y. One is too inflexible ("too cold") because it does not permit any inflection points. The final model ("just right") permits exactly one inflection point. In a simulation study based on 405 monotonic-increasing or inverse u-shaped functional forms and over 200 thousand simulated datasets, I show that my proposed algorithm outperforms the current favored method for testing u-shaped and inverse u-shaped hypotheses, which uses the Robin Hood algorithm in conjunction with a two-lines test, in terms of controlling the false rejection rate and the power of the test. I also show that these advantages of the Goldilocks algorithm can be further leveraged when it is used in an ensemble method that utilizes the output from both algorithms.

Key words: Statistics: nonparametric; Simulation: statistical analysis; Organizational studies: strategy

1. Introduction

Theories that predict u-shaped and inverse u-shaped relationships are ubiquitous throughout the social sciences (see, e.g. Haans et al. 2016, Lind and Mehlum 2010, Simonsohn 2018). In management theory and strategy, for example, non-monotonic hypotheses about the expected relationship between an independent variable x and a dependent variable y are integral to theories of organizational founding and failure (Hannan and Freeman 1989), competitive positioning (Deephouse 1999), innovation rates in competitive industries (Aghion et al. 2005), social status and conformity (Phillips and Zuckerman 2001), and ambidextrous search (Laursen and Salter 2006). As a result of

this widespread interest in identifying u-shaped and inverse u-shaped relationships in data and the well-known problems with standard parametric approaches based on quadratic regression models, there has been considerable recent interest in finding new ways to evaluate such theories using alternative approaches.

In this paper, I propose a new semi-parametric method for evaluating these theories, which I call the "Goldilocks" algorithm. The algorithm is so named because it involves estimating three models in order to evaluate a u-shaped or inverse u-shaped hypothesis. One model is too flexible ("too hot") because it permits multiple inflection points in the expected relationship between x and y. One is too inflexible ("too cold") because it does not permit any inflection points. The final model ("just right") permits exactly one inflection point. The logic behind the Goldilocks algorithm is that, if the expected relationship between x and y is indeed well-characterized by a single inflection point, e.g., an inverse u-shape, then the "too cold" model should be a poor fit for the data and the "just right" and "too hot" models should fit the data just about equally well after correcting for overfitting.

After introducing the algorithm, I compare the performance of Goldilocks to the recently-proposed Robin Hood algorithm, which is a heuristic method for setting the break point in a two-lines test (Simonsohn 2018). Based on a simulation study that evaluates 315 monotonic-increasing functional forms and 3420 inverse u-shaped functional forms, Simonsohn (2018) concludes that his approach, which uses a cubic spline regression model to approximate the ideal breakpoint in an interrupted regression model, "is arguably the most straightforward test of the hypothesis that the average effect of x on y is of opposite sign for high versus low values on x [and] the Robin Hood procedure to set the break point for the two lines achieves notably higher power than any alternative with which I have compared it" (p. 553). As a result, the Robin Hood algorithm has quickly become a favored method for evaluating u-shaped and inverse u-shaped theories in the management and strategy literatures (see, e.g., Aven et al. 2021, Bechler et al. 2021, Denrell and Liu 2021, Shin and Grant 2019).

Based on a representative sample of the same functions used to demonstrate the superiority of the Robin Hood algorithm over existing methods, I show that the Goldilocks algorithm outperforms the Robin Hood algorithm both with respect to not falsely identifying monotonic-increasing data as inverse u-shaped (i.e., control of the false rejection rate at a 5 percent target) and correctly identifying inverse u-shaped data (i.e., the power of the test). Across 63 monotonic-increasing functional forms, the Goldilocks algorithm incorrectly identifies the data as inverse u-shaped 3.1 percent of the time, compared with 3.8 percent for Robin Hood. Additionally, the false rejection rate for the Goldilocks algorithm exceeds the 5 percent target for just 9 of the functional forms, compared with 18 for Robin Hood. Across 342 inverse u-shaped functional forms, the Goldilocks

algorithm correctly identifies the data as inverse u-shaped 53.4 percent of the time, compared with 49.3 percent for Robin Hood. Further, the power of the Goldilocks algorithm exceeds 90 percent for 110 functional forms compared with 87 for Robin Hood. The advantages of the Goldilocks algorithm in terms of control of false rejection rate and statistical power can be further leveraged when it is used in conjunction with the Robin Hood algorithm. Specifically, I propose an ensemble method in which a dataset is identified as inverse u-shaped if either algorithm identifies the data as inverse u-shaped at the 0.01 significance level, which reduces the average false rejection rate to 1.6 percent while achieving the equivalent power as when the Goldilocks algorithm is used on its own.

The paper proceeds as follows. I begin by introducing the Golidlocks algorithm, focusing first on the "too hot, too cold, just right" approach to identifying functional families using nested shape-constraints, then describing the implementation of the algorithm, and finally applying the algorithm to an example dataset. Then, I briefly summarize the Robin Hood algorithm for setting the break point in the two-lines test, focusing in particular on how the Robin Hood approach departs from the Golidlocks approach. I then apply both methods to a representative sample of the functional forms evaluated in Simonsohn (2018). I also introduce two ensemble methods—one in which the significance level on both tests is raised to 0.1 and the data is classified as inverse u-shaped if *both* classify the data as inverse u-shaped and one in which the significance level is decreased to 0.01 and the data is classified as inverse u-shaped if *either* classify the data as inverse u-shaped—and demonstrate that the latter improves the false rejection rate of the Goldilocks algorithm at no cost with respect to statistical power.

2. The Goldilocks Algorithm

In this section, I introduce the Goldilocks algorithm. I begin by introducing the high-level structure of my proposed approach. I continue by describing each of the steps of the algorithm in detail.[1] I then apply the Goldilocks algorithm to an example dataset, which is the same example presented in Simonsohn (2018, fig. 6).

2.1. "One Too Hot, One Too Cold, One Just Right."

The "too hot, too cold, just right" logic of the Goldilocks algorithm involves estimating three models—one under-constrained, one over-constrained, one appropriately-constrained—and then evaluating their relative fit in order to evaluate whether the data are consistent with the appropriately-constrained model. This logic points to the following three-stage structure for evaluating an inverse u-shaped hypothesis: The first stage involves selecting three models: one that

[1] Computer code for implementing the algorithm and replication materials for the presented simulations are also available from the author upon request. An `R`-package for implementing the algorithm is in the works, as well.

permits multiple inflection points in the conditional mean, one that permits no inflection points, and one that permits exactly one inflection point. In the implementation of the algorithm used here, I use unconstrained, monotonic-increasing, and inverse u-shaped linear spline regression models. The second stage evaluates each model in order to determine which fit the data best. I use the Model Confidence Set (MCS) procedure, which returns a set of models that are equally well-fitting for which a model is incorrectly excluded at a user-specified rate (Hansen et al. 2011). The third stage uses this set of best-fitting models to evaluate the claim that the data are well-characterized by the theorized non-monotonic relationship, e.g., u-shaped or inverse u-shaped. Note that, because the monotonic-increasing vs. inverse u-shaped test is the one emphasized in the simulations in Simonsohn (2018) it is also the one examined here, with the former hypothesis treated as the null and the latter hypothesis treated as the alternative. However, other hypothesis tests with a monotonic null and an alternative with a single inflection point are easily accommodated.

Next, I describe each of the three stages in detail, emphasizing both the choices made in my proposed implementation of the algorithm and, in some cases, opportunities for alternative implementations.

2.1.1. Step 1: Select three models The algorithm begins with the identification of three models—one under-constrained, one over-constrained, and one appropriately constrained. In the implementation of the algorithm presented here, I select three shape-constrained linear spline regression models. In linear spline regression, a function is fit to the data in which a set of values of x called *knots* are connected by linear *pieces*. The estimated conditional expectation of y given x is therefore linear in each *bin* but not differentiable at the knots where the slope of the function changes. For background on estimating basis splines, see Cattaneo et al. (2019) or Hastie et al. (2009, pp. 142–150).

A shape constraint describes a family of functions that satisfy a set of constraints that collectively describe a shape. For example, in order for a function $f(x)$ to be monotonically-increasing, $f(x_1) \leq f(x_2)$ if $x_1 < x_2$ for all x_1, x_2. A monotonic-increasing shape-constrained regression, by extension, returns a conditional expectation function $E(y|x)$ subject to the constraint that $E(y|x_1) \leq E(y|x_2)$ if $x_1 < x_2$ for all x_1, x_2. A monotonic-increasing shape constraint applied to a linear spline thus requires that the slope of each piece is weakly positive. An inverse u-shaped shape constraint applied to a linear spline regression model asks that one knot serve as an inflection point such that the slopes of the linear pieces are weakly positive for low values of x and weakly negative for high values of x. An unconstrained linear spline regression places no restrictions on the slopes of the pieces in the conditional expectation function. These three models are therefore nested, i.e., an inverse u-shaped model is a constrained version of an unconstrained model and a monotonic-increasing model is a constrained version of an inverse u-shaped model.

Next, I discuss why I use shape-constrained linear splines, given the multitude of potential choices for describing the expected relationship between x and y. Then I discuss why I choose these three shape constraints, noting the impact of excluding the unconstrained model or including additional shape-constrained models.

The choice to estimate linear spline regression models as opposed to other flexible summaries of the conditional expectation function is motivated by three interrelated concerns. The first is computational. I need to be able to compute a best-fitting shape-constrained model for the monotonic-increasing and inverse u-shaped models. For linear spline regression, adding a shape constraint involves solving a quadratic program, which can be done quickly and reliably using standard software (see, e.g., Boyd and Vandenberghe 2004). For other non-parametric models, e.g., cubic spline regression, algorithms for computing the optimal solution may not exist or require grid search over a large number potential inflection points in order to estimate. Further, shape-constrained linear splines can be straightforwardly estimated as part of a wide variety of multivariate models through the application of a convex optimization algorithm. Thus, the Goldilocks algorithm can be applied to multivariate linear models, generalized additive models, nonlinear least squares, discrete choice models, count models, and hazard models, among others (see Boyd and Vandenberghe 2004, Hothorn et al. 2018).

The second relates to controlling the false rejection rate. If, e.g., the overly-flexible model is *too* flexible, then the application of an inverse u-shaped shape-constraint may produce a better fit with the data than an unconstrained model when evaluated using a loss criterion that incorporates an overfitting penalty in cases for which the true expected relationship between x and y has multiple inflection points. This would lead the monotonic-increasing vs. inverse u-shaped test to be biased toward supporting the inverse u-shaped hypotheses. A feature of the linear spline model is that there exist algorithms for selecting knots that minimize the asymptotic integrated mean squared error (IMSE) (see Cattaneo et al. 2019) and, thus—at least for large samples—I can feel fairly confident that the unconstrained model is not badly over-fit.

The third relates to maximizing the power of the test. By estimating nested functions, I improve the MCS procedure's ability to distinguish between the fit of the three models relative to testing three functions from different non-parametric families (Hansen et al. 2011, Ganz 2020). So, while I could, e.g., estimate an isotonic regression for the monotonic-increasing model, an interrupted regression for the inverse u-shaped model, and a kernel regression for the unconstrained model—and perhaps these functions would individually fit the data better than a shape-constrained linear spline would—it would come at a large cost in terms of the ability to reject the claim that a subset of models are equally well-fitting at the target confidence level.

Of note, these second and third considerations imply a trade-off between the false rejection rate and the power of the test given my choice of models. In order for the models to be nested, the shape-constrained linear splines must have the same number and location of knots as the unconstrained linear spline. But, the number and location of those knots may be a worse choice for the shape-constrained models than they are for the unconstrained model. This can lead to elevated false rejection rates in cases where the monotonic-increasing model is rejected in favor of the inverse u-shaped and unconstrained models due to the choice of knots. Relative to other sources of error that emerge from the fact that the models are being estimated on limited samples, I suspect this bias is quite small. Nevertheless, it does exist and, as a result, makes it difficult to make statistical claims about the large-sample properties of the Goldilocks algorithm.

Next, I discuss how selecting *three* models—one "too hot", one "too cold", and one "just right"—helps balance these competing concerns about the false rejection rate and the power of the test, as well. Given the goal of differentiating between monotonic-increasing and inverse u-shaped data, it is perhaps counter-intuitive to also estimate a third unconstrained model. The purpose of the unconstrained model is to add a conservative bias to the test by ensuring that data that are not well-characterized as either monotonic-increasing or inverse u-shaped are not incorrectly classified as inverse u-shaped merely because the inverse u-shaped model is more flexible. Applying a hypothesis testing framework that does not also account for the possibility that the data are neither monotonic-increasing or inverse u-shaped will thus lead to an elevated false-rejection rate (see, e.g., the shape-restricted hypothesis-testing approach in Cattaneo et al. 2019).

A related question, then, is why *just* three models? I could estimate a larger set of nested models, including, e.g., a constant model or a "sideways s-shaped" model with two inflection points. Because, in expectation, the monotonic-increasing model and constant model will fit the data equally well if the data are constant (and the sideways s-shaped model and the unconstrained model will fit the data equally well if the data are sideways s-shaped), the inclusion of these additional shapes has an ambiguous impact on the false rejection rate and the power of the test, but comes at a high computation cost. That said, if a researcher has a theory about a specific additional shape the data are likely to take then there is likely little harm in adding that shape to the test.

2.1.2. Step 2: Evaluate model fit The second step of the algorithm involves evaluating the relative fit of each model with the data. I implement the MCS procedure (Hansen et al. 2011), which is a step-down multiple hypothesis testing procedure applied to the problem of model selection. The MCS procedure has two benefits which explain why I adopt it rather than a different model selection algorithm. First, it permits the user to specify a rate at which a true best-fitting model is excluded from the set returned by the procedure, which permits control of the Goldilocks algorithm's false

rejection rate at a user-specified level. Second, prior simulation studies have shown that when the expected Kullback-Leibler Information Criterion (KLIC) loss criterion is used, the MCS procedure performs well for shape-constrained bivariate regression models with ordinal data (Wand 2012, Ganz 2020) and for multivariate linear regression models with continuous data (Hansen et al. 2011). Thus, it is a good candidate for evaluating the relative fit of shape-constrained models estimated on continuous data as well.

The MCS procedure involves iterative comparisons of the relative fit of sets of candidate models with the data. At each stage, the algorithm evaluates whether all of the candidate models $\mathcal{M}$ fit the data equally well according to a given loss criterion. If the hypothesis that the models indeed fit the data equally well is rejected, then the worst-fitting model is removed from the set of candidates and the culled set of models $\mathcal{M}' \subset \mathcal{M}$ is subsequently evaluated. The Model Confidence Set $\hat{\mathcal{M}}^*$ is the set of models that is returned either when the algorithm fails to reject the claim that the models fit the data equally well or when just a single model remains. The design of the MCS procedure ensures that, as $n \to \infty$, $P(\mathcal{M}^* \subset \hat{\mathcal{M}}^*)$ converges to a user-specified rate that exceeds $1 - \alpha$, where $\mathcal{M}^*$ is the subset of true best-fitting models in an initial candidate set of models $\mathcal{M}^0$.

In my implementation of the MCS procedure, the initial set of candidate models $\mathcal{M}^0$ includes the monotonic-increasing linear spline model ($M_{\nearrow}$), the inverse u-shaped linear spline model ($M_{\frown}$), and the unconstrained linear spline model ($M_{\rightsquigarrow}$). I evaluate the fit of the models with the data according to the expected KLIC loss criterion. For regression models, ranking models according to expected KLIC is equivalent to ranking them according to the maximum of their quasi-log-likelihood. Define $Q(Z, \theta_i)$ as twice the negative quasi-log-likelihood for the data $Z = (y, x)$ evaluated at the population parameters for model M_i, where $i \in \{\nearrow, \frown, \rightsquigarrow\}$. Define the true set of best-fitting models according to the expected KLIC loss criterion as $\mathcal{M}^*_{\text{KLIC}} = \{j : E[Q(Z, \theta_j)] = \min_i E[Q(Z, \theta_i)]\}$ and $\hat{\mathcal{M}}^*_{\text{KLIC}}$ as the set of best-fitting models inferred from the sample.

Estimating $E[Q(Z, \theta_i)]$ from the sample analog $E[Q(Z, \hat{\theta}_i)]$ requires an effective degrees of freedom correction $k_i^* = E[Q(Z, \theta_i) - Q(Z, \hat{\theta}_i)]$. Unlike in a traditional linear regression model, the effective degrees of freedom k_i^* for a shape-constrained linear spline cannot be directly inferred from the number of estimated parameters because the location of the knots and the expected number of binding shape-constraints depend on the data itself. To see this, consider the case where the true data were produced by a monotonic-decreasing function. The difference in effective degrees of freedom for the best-fitting monotonic-increasing function and the best-fitting constant function should be approximately equal. In contrast, if the true data were produced by a monotonic-increasing function, it is likely that each piece of the monotonic-increasing function would have a different slope and, thus, the difference in the the effective degrees of freedom would approximately equal the number of pieces.

The need to estimate k_i^* given the data implies the desirability of a simulation-based estimator, which I compute in the following manner, as recommended in Shibata (1997). Define $Z_{b,i}^* = (y_b, x)$ for $b = 1, ..., B$ bootstrap samples constructed from a semiparametric bootstrap, where $y_b = E[y|x, \hat{\theta}_i] + e_{b,i}$ and $e_{b,i}$ is randomly sampled with replacement from $e_i = y - E[y|x, \hat{\theta}_i]$. Define $\hat{k}_i^* = \frac{1}{B}\sum_b [Q(Z_{b,i}^*, \hat{\theta}_i) - Q(Z_{b,i}^*, \hat{\theta}_{b,i}^*)]$ as our estimate of k_i^* given the data, where $\hat{\theta}_{b,i}^*$ is the estimate of $\hat{\theta}_i$ inferred from bootstrap sample $Z_{b,i}^*$.

For my implementation of the MCS procedure it is also useful to define $\hat{k}_{i,j}^* = \frac{1}{B}\sum_b [Q(Z_{b,j}^*, \hat{\theta}_i) - Q(Z_{b,j}^*, \hat{\theta}_{b,i}^*)]$ as the effective degrees of freedom for model i when estimated on data produced by model j. That is, if i is an inverse u-shaped model and j is a monotonic-increasing model, then $k_{\cap,\nearrow}^*$ is the effective degrees of freedom correction applied to $Q(Z, \theta_\cap)$ under the assumption that $E(y|x)$ is in fact characterized by the monotonic-increasing model. Further, define $\Delta\hat{k}_{i,j,m}^* = \hat{k}_{i,m}^* - \hat{k}_{j,m}^*$, i.e., the difference in the number of effective degrees of freedom for models i and j for data produced by model m.

Next, I describe the details of my implementation of the MCS procedure, which follows a recommendation in Wand (2012) for taking advantage of the nesting of the models in M^0 in order to improve the power of the test.[2] Note that, for each iteration of the MCS procedure, the hypothesis that all of the models fit the data equally well is equivalent to the claim that the data are consistent with the most restrictive model in $\mathcal{M}$. For example, in the first iteration of the procedure, the monotonic-increasing model and inverse u-shaped model must fit the data equally well in order for the hypothesis of equal fit to be rejected, which occurs only if the data are in fact monotonic-increasing.

Each iteration of the MCS procedure begins with a set of candidate models $\mathcal{M}$, where $\mathcal{M} = \mathcal{M}^0$ in the first iteration of the procedure. Define $M_0 \in \mathcal{M}$ as the most restrictive model in $\mathcal{M}$. The procedure tests the claim that all of the models in $\mathcal{M}$ fit the data equally well by computing a range statistic $\hat{\mathcal{T}}_\mathcal{M} = \max_{i,j\in\mathcal{M}} |[Q(Z, \hat{\theta}_i) - Q(Z, \hat{\theta}_j) + \Delta\hat{k}_{i,j,M_0}^*]|$. $\hat{\mathcal{T}}_\mathcal{M}$ is thus the largest pairwise difference in expected KLIC among candidate models in $\mathcal{M}$ under the null assumption that the data are produced by M_0.

I next compute the distribution of $\hat{\mathcal{T}}_\mathcal{M}$ and compare $\hat{\mathcal{T}}_\mathcal{M}$ to the $1-\alpha$ quantile of this distribution using the following algorithm:

(1) Define $Z_{b,M_0}^* = (y_{b,M_0}, x)$, where $y_{b,M_0} = E(y|x, \theta_{M_0}) + e_{b,M_0}$ for $b \in 1, ..., B$ and e_{b,M_0} is randomly sampled with replacement from $e_{M_0} = Y - E(y|x, \theta_{M_0})$.

(2) For each bootstrap sample b, compute the range statistic $\hat{\mathcal{T}}_{0,b,\mathcal{M}} = \max_{i,j\in\mathcal{M}} |[Q(Z_{b,M_0}^*, \hat{\theta}_i) - Q(Z_{b,M_0}^*, \hat{\theta}_j) + \Delta\hat{k}_{i,j,M_0}^*]|$.

[2] Many thanks to Jonathan Wand for sharing the code used in his implementation of the MCS procedure.

(3) Compare $\hat{\mathcal{T}}_{\mathcal{M}}$ to the $1-\alpha$ quantile of $\hat{\mathcal{T}}_{0,b,\mathcal{M}}$. If $\hat{\mathcal{T}}_{\mathcal{M}}$ exceeds this level, then reject the claim that all of the models in $\mathcal{M}$ are all equally well-fitting. Otherwise, fail to reject the claim.

If we fail to reject the claim that all of the models in $\mathcal{M}$ are equally well-fitting, then $\hat{\mathcal{M}}^*_{KLIC} = \mathcal{M}$. If we reject the claim and there are more than two models in $\mathcal{M}$, then remove the model $i \in \mathcal{M}$ with the highest value of $Q(Z, \hat{\theta}_i) + \hat{k}^*_{i,M_0}$ and re-evaluate the smaller candidate set $\mathcal{M}' \subset \mathcal{M}$. If we reject the claim and there are two models in $\mathcal{M}$ then we define the model with the lowest value of $Q(Z, \hat{\theta}_i) + \hat{k}^*_{i,M_0}$ as the lone member of $\hat{\mathcal{M}}^*_{KLIC}$.

As a result of the nesting of models in $\mathcal{M}^0$ and the expected KLIC loss criterion, it is usually the case that, if the claim that all models in $\mathcal{M}^0$ are equally well-fitting is rejected in each iteration, $M_{\nearrow}$ will be rejected in the first iteration, followed by $M_{\frown}$ in the second. Thus, if there is only one model in $\hat{\mathcal{M}}^*_{KLIC}$ then it is usually $M_{\backsim}$. But, on rare occasions, simulation-based error can lead a less-constrained model to be rejected in favor of a more-constrained one.

2.1.3. Step 3: Monotonic-increasing vs. inverse u-shaped We use $\hat{\mathcal{M}}^*_{KLIC}$ to evaluate the claim that the data are monotonic-increasing vs. inverse u-shaped. Recalling that, as $n \to \infty$, a model will correctly be included in $\hat{\mathcal{M}}^*_{KLIC}$ with probability approaching $1-\alpha$ and incorrectly included with probability approaching 0, we reject the claim that the data are produced by an inverse u-shaped function if $M_{\nearrow} \in \hat{\mathcal{M}}^*_{KLIC}$ or if $M_{\frown} \notin \hat{\mathcal{M}}^*_{KLIC}$ and support the claim if $M_{\nearrow} \notin \hat{\mathcal{M}}^*_{KLIC}$ and $M_{\frown} \in \hat{\mathcal{M}}^*_{KLIC}$.

Note that the construction of the test is consistent with classical hypothesis testing in the linear regression framework. All else equal, we are biased in favor of a null, more constrained model relative to an alternative, less constrained model. This explains why the algorithm does not reject the monotonic-increasing model in favor of the inverse u-shaped model when neither $M_{\nearrow}$ nor $M_{\frown}$ are in $\hat{\mathcal{M}}^*_{KLIC}$. This conservative bias is important for controlling the rate at which the algorithm incorrectly classifies data as inverse u-shaped at (approximately) the user-defined α in small samples. Further, it helps explain why it is inappropriate to use tests of the form monotonic-increasing vs. not monotonic-increasing to evaluate claims of inverse u-shaped relationships in data, because the alternative hypothesis would both be supported when the data are inverse u-shaped and when the data have a different non-monotonic functional form. Asymptotic control of the false rejection rate when the alternative is "not monotonic-increasing" does not control the false rejection rate when the alternative is "inverse u-shaped."

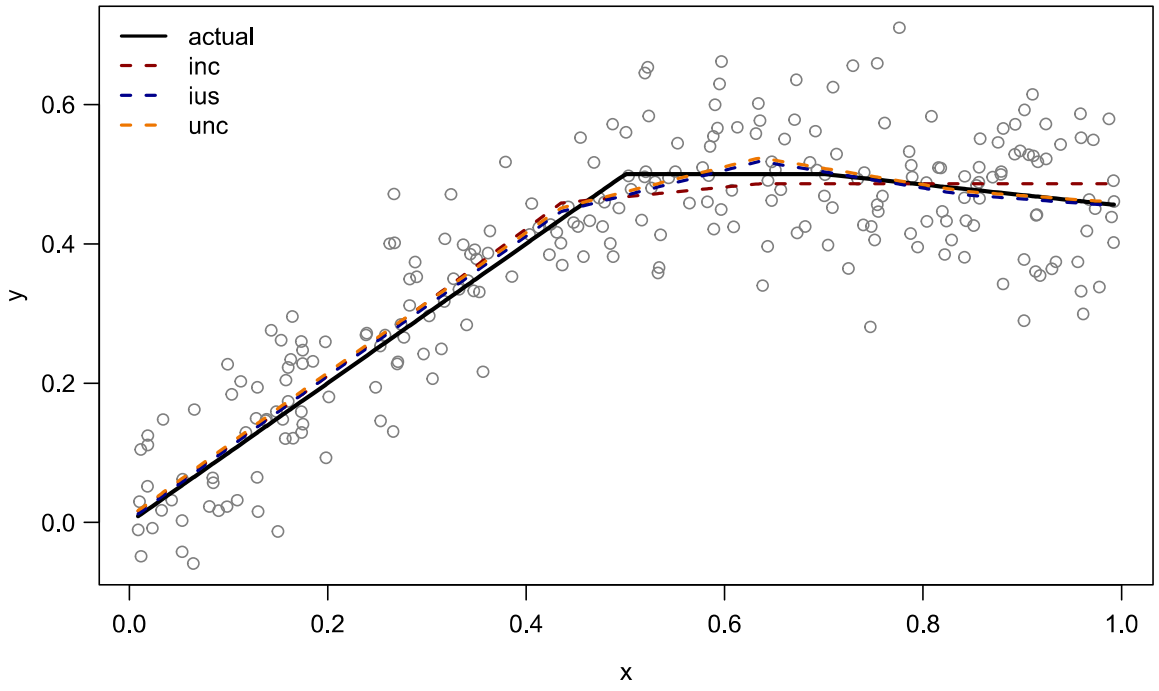
Finally, it is worth highlighting why we cannot expect the test to achieve the exact user-defined false rejection rate for all data, even in large samples. The MCS procedure selects the best-fitting models out of a set of candidates, all of which may be misspecified. And, in the proposed implementation of the MCS procedure, this is almost surely the case. (It is hard to imagine data in the

real world that are produced by a linear spline conditional expectation function.) Our test relies on the assumption that linear spline regression offers a reasonably good description of conditional expectation of y given x. But, as I demonstrate in the simulated data, we may experience elevated false rejection rates for data that are especially poorly modeled by the estimated piecewise-linear functional form.

2.2. An Application of the Goldilocks algorithm

I now apply the Goldilocks algorithm to an example in which data are produced by an inverse u-shaped function. In Figure 1, I display the sample along with the actual function (the black line), the best-fitting monotonic-increasing linear spline (the red dashed line), inverse u-shaped linear spline (the blue dashed line), and unconstrained linear spline (the orange dashed line). Note that the latter two models fit the exact same function to the data, because the best-fitting unconstrained linear spline is inverse u-shaped.

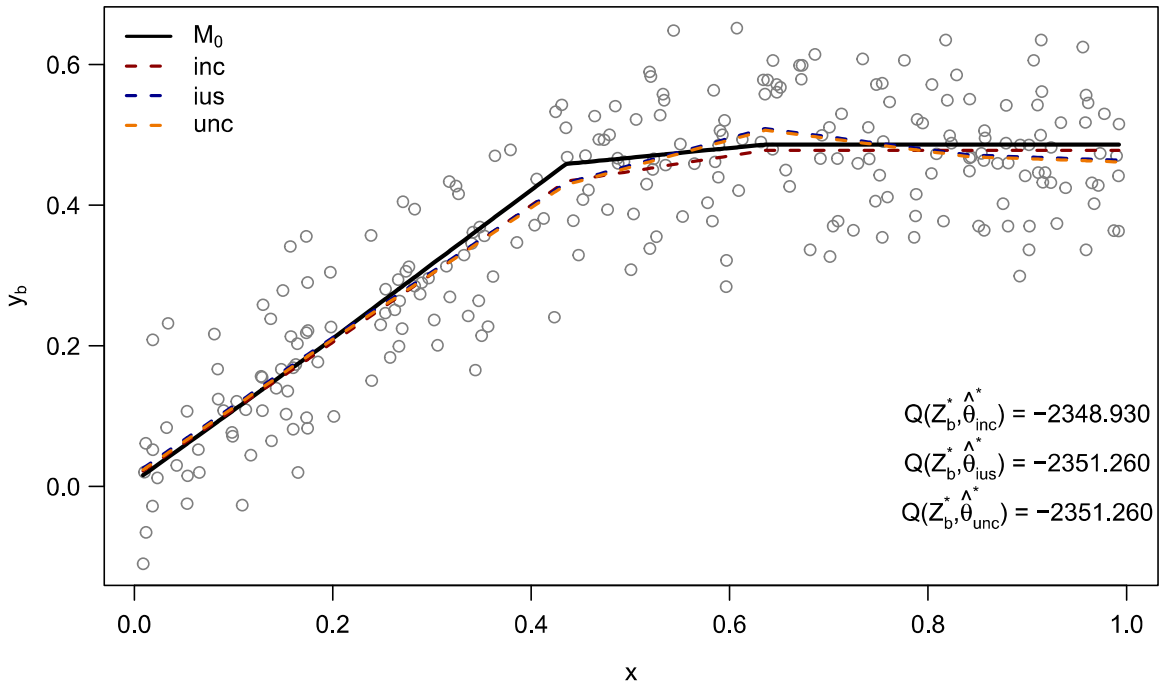
Figure 1 Estimated Shape Constrained Models



Next, I illustrate some of the mechanics of the MCS procedure. Figure 2 displays one bootstrap sample produced by the monotonic-increasing model, which is the most-constrained model in $\mathcal{M}^0$ (and thus denoted M_0 in the figure). I display the bootstrap sample along with the monotonic-increasing function fit to the original sample (the black line) and the best-fitting monotonic-increasing linear spline, inverse u-shaped linear spline, and unconstrained linear spline for the

bootstrap sample (the red dashed, blue dashed, and orange dashed lines, respectively). In the bottom-right, I present $Q(Z_b^*, \hat{\theta}_i)$ for $i \in \mathcal{M}^0$, which are the log-likelihoods of the three models in $\mathcal{M}^0$ fit to the bootstrap sample. The difference between these values of Q offer insight into the difference in effective degrees of freedom across the shape-constrained models when applied to data produced by this monotonic-increasing function. Note that the inverse u-shaped and unconstrained models fit two additional slope parameters than the monotonic-increasing model, leading $Q(Z_b^*, \hat{\theta}_{\nearrow})$ to be greater than $Q(Z_b^*, \hat{\theta}_{\cap})$ and $Q(Z_b^*, \hat{\theta}_{\rightsquigarrow})$.

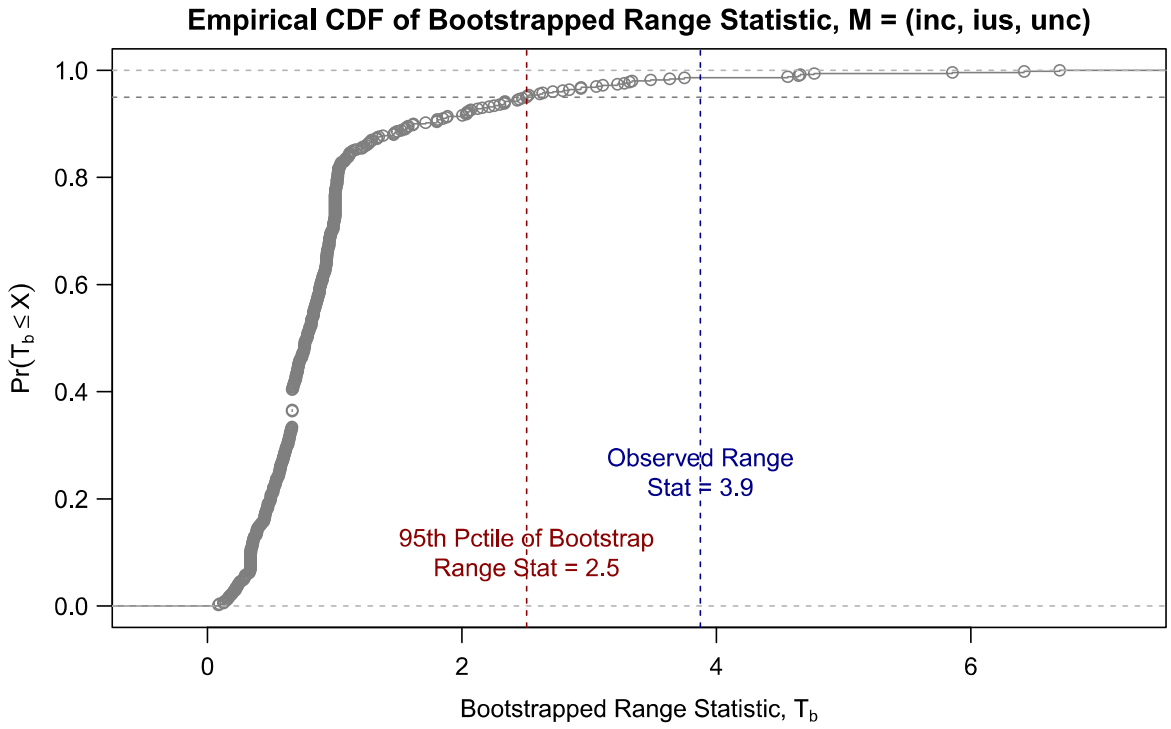
Figure 2 Iteration 1 of the MCS Procedure: Evaluating Relative Fit from Bootstrap Samples



In Figure 3, I display the CDF of the estimated range statistic under the null that all the models in $\mathcal{M}$ are equally well-fitting $\hat{\mathcal{T}}_{0,b,\mathcal{M}}$ across 500 bootstrap samples. The 95$^{\text{th}}$ percentile is 2.5, as indicated by the red dashed line. The observed range statistic $\hat{\mathcal{T}}_{\mathcal{M}}$ is 3.9, which exceeds this critical value. Thus, the null is rejected and the worst-fitting model $M_{\nearrow}$ is dropped from $\mathcal{M}$.

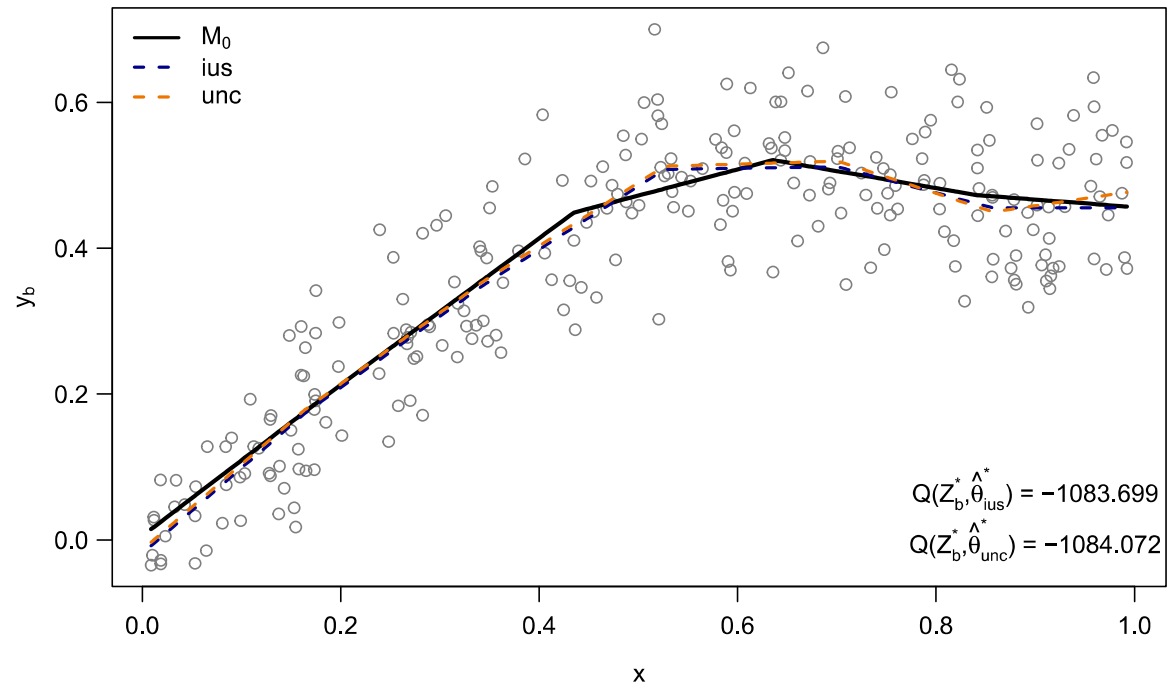
In the second iteration of the MCS procedure, $\mathcal{M}$ includes the inverse u-shaped and unconstrained models. In Figure 4, I display one bootstrap sample produced by the current most-constrained model M_0, which is the inverse u-shaped model. As before, I display the bootstrap sample along with the inverse u-shaped function fit to the original sample (the black line) and the best-fitting inverse u-shaped linear spline and unconstrained linear spline for the bootstrap sample (the blue dashed and orange dashed lines, respectively). In this case, the unconstrained model fits

Figure 3 **Iteration 1 of the MCS Procedure: Implementing the Test of Equal Fit**



one additional slope parameter than the inverse u-shaped model, shown by the increasing slope of the right-most piece of the unconstrained linear spline.

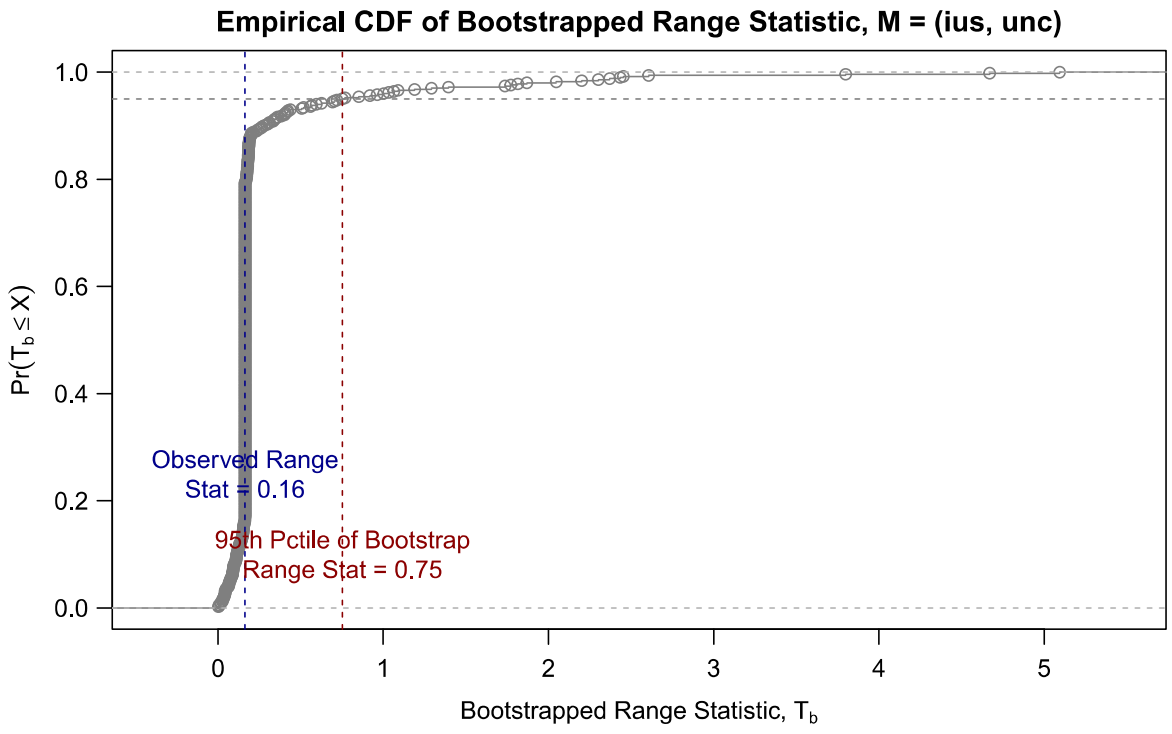
Figure 4 **Iteration 2 of the MCS Procedure: Evaluating Relative Fit from Bootstrap Samples**



In Figure 5, I display the CDF of the estimated range statistic under the null that the two models in $\mathcal{M}$ are equally well-fitting $\hat{\mathcal{T}}_{0,b,\mathcal{M}}$ across 500 bootstrap samples. The 95^{th} percentile is 0.75, as indicated by the red dashed line. The observed range statistic $\hat{\mathcal{T}}_{\mathcal{M}}$ is 0.16, which is less than this critical value. Thus, the null is not rejected and both the inverse u-shaped and unconstrained models ($M_{\frown}$ and $M_{\rightsquigarrow}$, respectively) are returned as $\hat{\mathcal{M}}^{*}_{KLIC}$.

Because $M_{\frown} \in \hat{\mathcal{M}}^{*}$ and $M_{\nearrow} \notin \hat{\mathcal{M}}^{*}$, the data are classified as inverse u-shaped.

Figure 5 Iteration 2 of the MCS Procedure: Implementing the Test of Equal Fit



3. The Robin Hood Algorithm

In this section, I briefly describe the Robin Hood algorithm in order to compare the manner in which the two algorithms evaluate the inverse u-shaped claim. I also apply the algorithm to the same example dataset analyzed previously.

The Robin Hood algorithm returns a break-point for use in an interrupted regression model. The slope parameters and significance levels of the model are then used to evaluate the inverse u-shaped hypothesis. The method for computing the break-point involves first estimating a flexible conditional expectation function (i.e., cubic spline regression) in order to get a good approximation of its peak, partitioning the data into two groups using this approximation, and then intelligently adjusting the partition so that the side with weaker statistical significance receives relatively more

data than the size with stronger statistical significance. The hypothesis test then uses the parameters estimated from the interrupted regression model with the adjusted partition to evaluate the inverse u-shaped claim. If the estimated slope parameter for the data to the left of the break-point is significantly positive at the 0.05-level and the estimated slope parameter to the right of the break-point is significantly negative at the 0.05-level, then the inverse u-shaped hypothesis is supported[3]

The logic that supports the two approaches to evaluating non-monotonic theories in data are at once complementary and contrasting. The two methods are complements in that they both estimate constrained models in order to evaluate the inverse u-shaped hypothesis. Further, both constrained models are estimated for the purpose of facilitating a valid hypothesis test, rather than to return the best-fitting conditional expectation function or for out-of-sample prediction. Neither the inverse u-shaped linear spline regression estimated by the Golidlocks algorithm nor the interrupted regression model estimated using the Robin Hood algorithm's break-point is necessarily a good summary of the data generating process. If the analyst, e.g., wants an estimate of the location of the inflection point of the inverse u-shaped function, there are many better methods that exist.

However, the manner in which the two algorithms use these constrained models to inform the hypothesis test is quite different. Whereas the hypothesis test in Goldilocks algorithm is derived from the relative fit of shape-constrained families of models with the data, the hypothesis test associated with the Robin Hood algorithm is derived from the slope and significance levels of parameters within a specific model. The different ways of approaching of the problem of testing inverse u-shaped hypotheses leads the two methods to arrive at widely divergent conclusions for certain monotonic-increasing and inverse u-shaped functional families.

Finally, the two methods disagree about whether a function with multiple inflection points whose average slope is strongly increasing to the left of global peak and then strongly decreasing to the right of that peak is indeed inverse u-shaped. For the Golidlocks algorithm, it is not. The multiple inflection points in the function may lead the unconstrained model to be a better fit for the data than the inverse u-shaped model. For the Robin Hood algorithm it is. The strongly increasing and decreasing average slopes will lead the interrupted regression model to identify the slopes as strongly statistically significant and of different sign. That said, if the researcher desires the latter

[3] In the original presentation of the algorithm, Simonsohn (2018) identifies the data as inverse u-shaped or u-shaped if the slope parameters are of different sign and both are significant at the 0.05-level. For the purposes of testing a monotonic-increasing vs. inverse u-shaped hypothesis, I add the additional restrictions that the slope for low values of x is positive and high values of x is negative. Given the setup of the original simulation study in which all of the functional forms are either monotonic-increasing or inverse u-shaped, this change has no effect on the presented results.

interpretation of inverse u-shaped and the functional form of the non-monotonicity that generates the multiple inflection points is known, then this can be easily accommodated through the inclusion of of appropriate control variables in the linear spline models estimated as part of the Goldilocks algorithm.

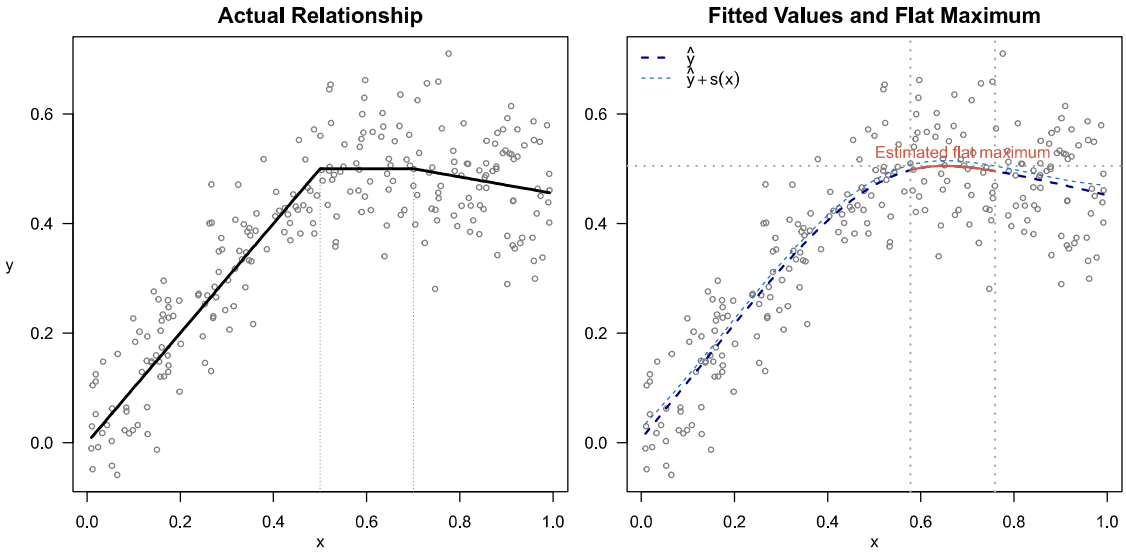
Next, I describe the steps of the Robin Hood algorithm in detail and illustrate its implementation using an example.

3.1. Robin Hood Step 1: Estimate a Flat Maximum Range

The Robin Hood algorithm begins by fitting a cubic spline regression to the data and then identifying the values of x for which $\hat{y} + s(x)$ exceeds the maximum value of $\hat{y}$, where $\hat{y}$ are fitted values of y and $s(x)$ is the standard error of $\hat{y}$ evaluated at x. These values of x correspond to the flat maximum region.

In Figure 6, I demonstrate how the flat maximum region is estimated. The left panel shows the actual functional relationship between x and y and the observed sample. The dashed dark blue line in the right panel shows $\hat{y}$ and the dashed light blue line shows $\hat{y} + s(x)$. The horizontal gray dotted line represents the maximum value of $\hat{y}$ and the vertical gray dotted lines represent the intersection of this horizontal line with $\hat{y} + s(x)$. The area between these vertical lines, represented in orange on the graph, is the estimated flat maximum range.

Figure 6 Identifying the Flat Maximum Range

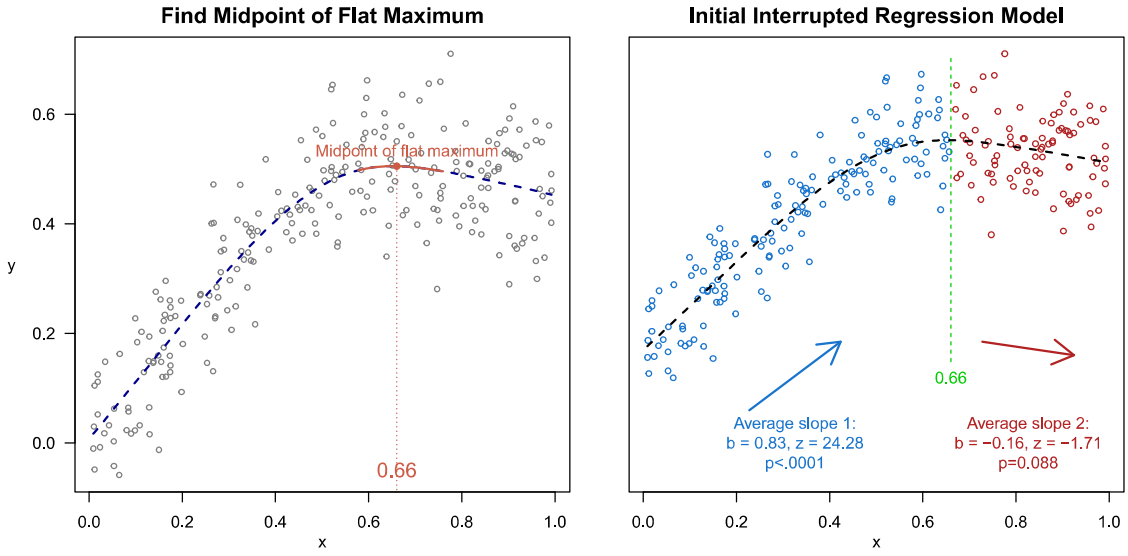


3.2. Robin Hood Step 2: Estimate an Initial Interrupted Regression Model

In the second stage, an interrupted regression is estimated where the break point is set at the midpoint of the estimated flat maximum range. The interrupted regression computes separate best-fit lines for the data to the left and right of the break point and, as a result, has a discontinuity at the break point.

In Figure 7, I show the midpoint of the estimated flat maximum range in the left panel and then present the results of the interrupted regression model in the right panel. The slope of the regression function to the left of the break point b_1 is 0.83 and to the right of the break point b_2 is -0.16. I also report the z-scores for the two slope parameters. Note that the absolute value of the z-score for b_1 is much larger than the absolute value of the z-score for b_2, indicating that the slope parameter differs from 0 at a higher significance level. In fact, b_2 is indistinguishable from 0 at the 0.05 significance level.

Figure 7 Computing Interrupted Regression Model Using Midpoint



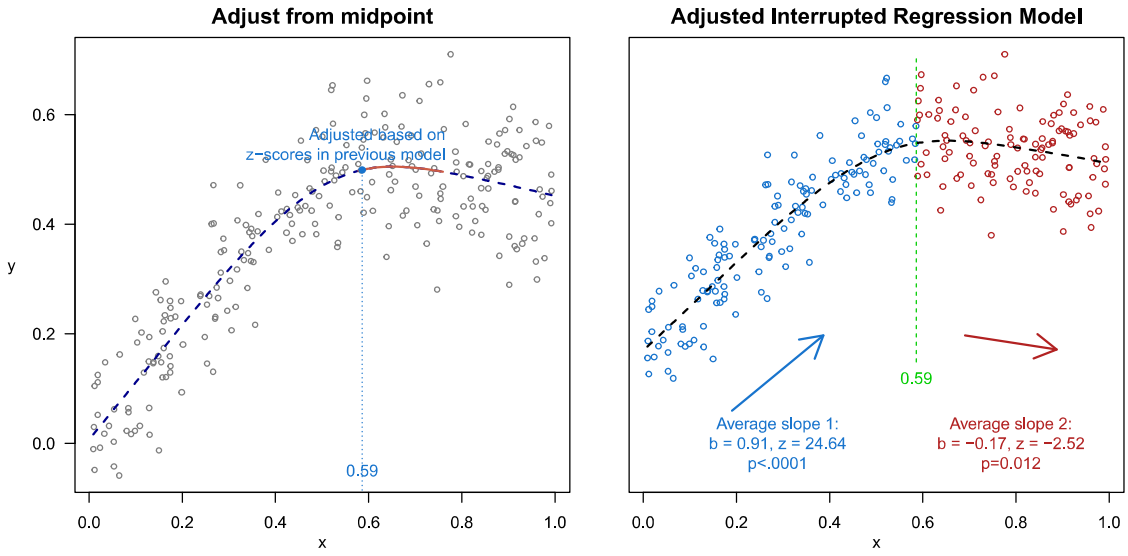
3.3. Robin Hood Step 3: Adjust the Break Point, Re-estimate the Model, and Interpret the Results

In the third stage, the coefficients estimated from the first interrupted regression model are used to update the break point. Define $Z = |z_2|/(|z_1| + |z_2|)$, where z_1 is the z-score associated with b_1 and z_2 is the z-score associated with z_2. Then place the adjusted break point at the Z^{th} percentile of the estimated flat maximum range and re-estimate the interrupted regression model. As a result, more of the sample is allocated to the segment of the interrupted regression that initially has a lower z-score.

In the left panel Figure 8, I show the adjusted midpoint of the interrupted regression model. In this example, $Z = 1.71/(24.28 + 1.71) = 6.58$, so the adjusted break point is much closer to the left boundary of the estimated flat maximum range than the right boundary. In the re-estimated model, both slope parameters b_1 and b_2 differ from 0 at the 0.05-significance level.

The Robin Hood algorithm evaluates the shape of the data based on the sign and significance level of b_1 and b_2. If both are significantly different from zero at the 0.05-level, the sign of b_1 is positive, and the sign of b_2 is negative, then the data are classified inverse u-shaped. Otherwise, they are not. Because the slope of the interrupted regression to the left of the break-point is significant and positive and to the left of the break-point is significant and negative, the test identifies the data as inverse u-shaped.

Figure 8 Computing Interrupted Regression Model Using Adjusted Break Point



3.4. Application to Data with Multiple Inflection Points

I next present a slight modification of the previous example in order to demonstrate how the two algorithms differ by design when the functional form has multiple inflection points. The example here replicates the previous example, except the values of y for all observations for which $x \leq 0.25$ or $0.5 < x \leq 0.75$ are increased by 0.2 and all observations for which $0.25 < x \leq 0.5$ or $x \geq 0.75$ are decreased by 0.2.

Figure 9 presents the true functional form (in the sold black line) and the best-fitting monotonic-increasing, inverse u-shaped, and unconstrained linear spline regression models (in the dashed red, blue, and orange lines, respectively). Given the additional flexibility afforded by the unconstrained model, it does a much better job of accommodating the multiple inflection points than the two

more-constrained models and, as a result, is the lone model returned in $\hat{M}^{*}_{KLIC}$, implying that the data are not inverse u-shaped.

Figure 9 Goldilocks Applied to Modified Example

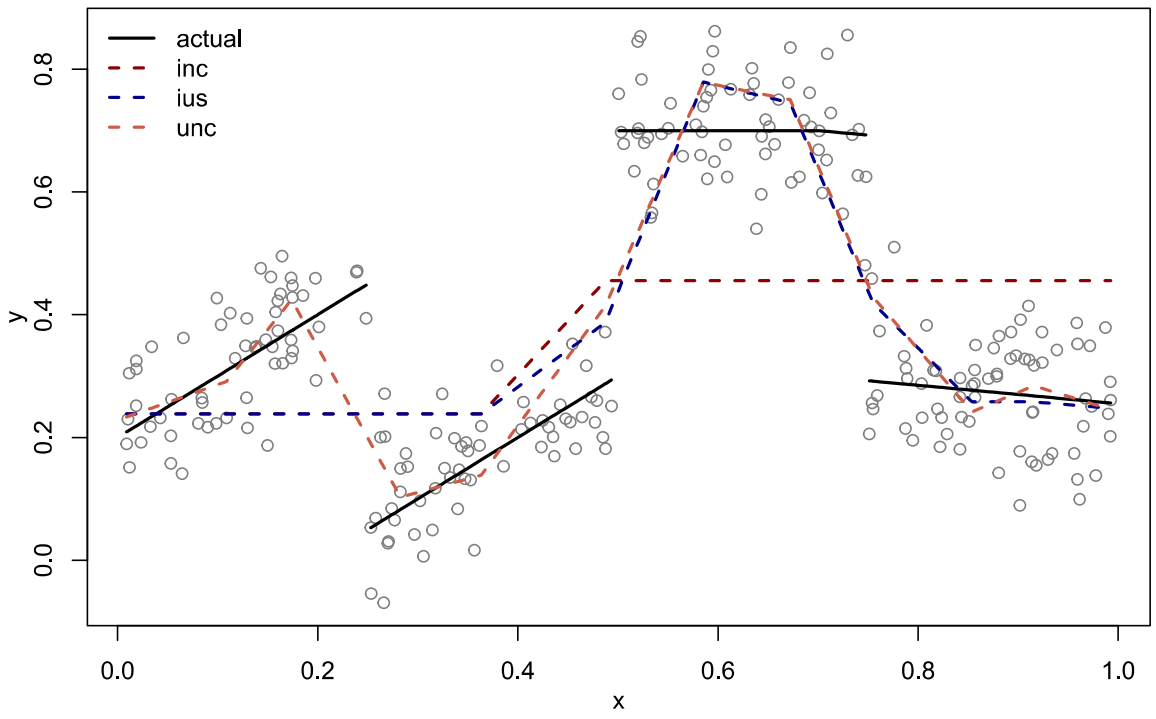
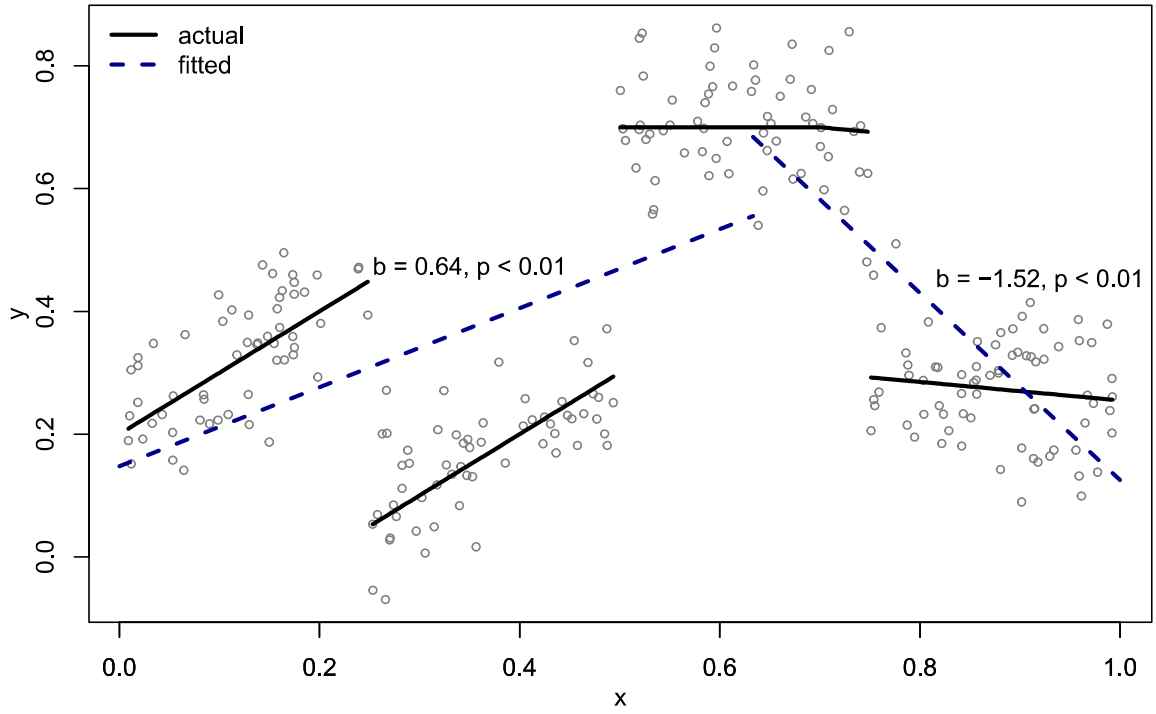


Figure 10 evaluates the same example using the Robin Hood algorithm. The blue dashed line indicates the estimated interrupted regression after using the Robin Hood algorithm to identify the break-point. As the slope of the interrupted regression to the left of the break-point is significant and positive and to the right is significant and negative, the test identifies the data as inverse u-shaped.

Whether the researcher prefers the interpretation of the Goldilocks algorithm or the Robin Hood algorithm in this case is context-dependent, I suspect. That said, in cases where the functional form has multiple inflection points and the cause of the multiple inflection points cannot be controlled for *and* the researcher desires to characterize a function like the one in the example as inverse u-shaped, I would recommend using the Robin Hood algorithm and the two-lines test. In contrast, if the fact that the data are neither monotonic-increasing nor inverse u-shaped leads the researcher to prefer to retain the more restrictive null hypothesis, then it would be preferable to use the Goldilocks algorithm.

Figure 10 Robin Hood Applied to Modified Example



4. Goldilocks vs. Robin Hood

Next, I present results from a horse-race in which the performance of the Goldilocks algorithm and Robin Hood algorithm are compared. The simulation evaluates the performance of both algorithms in terms of falsely-identifying monotonic-increasing data as inverse u-shaped and correctly identifying inverse u-shaped data across a variety of functional forms. The horse-race is constructed to mirror the design of the simulations presented in Simonsohn (2018) so that the same evidence used to show the superiority of the Robin Hood algorithm to competing methods can also be used to show the superiority of the Goldilocks algorithm to competing methods.

Simonsohn (2018) evaluates four families of functions, two of which are monotonic increasing and two of which are inverse u-shaped. The first monotonic-increasing family includes 180 functions that are either piecewise-linear or piecewise-log-linear, where the function is increasing for low values of x and constant for high values of x. The second monotonic-increasing family includes 135 logistic functions of the form $y = e^{-bx}$. The first inverse u-shaped family includes 2520 piecewise-linear or piecewise-log-linear functions, where the function is increasing for low values of x, decreasing for high values of x and, in some cases, constant for intermediate values of x. The second inverse u-shaped family includes 900 functions of the form $y = x - ax^k$, where a and k are selected such that the inflection point of the function lies in the support of the data.

I first replicate the simulations in Simonsohn (2018) using the code made public by the author and then order the functions in each family according to the rate at which they classify the simulated

data as inverse u-shaped. For the monotonic-increasing families, I select every 5^{th} function from this ordered list (i.e., I select the function that identifies the smallest share of simulated datasets as inverse u-shaped, the function that identifies the 6^{th}-smallest share of simulated datasets as inverse u-shaped, and so on). For the inverse u-shaped family, I select every 10^{th} function (i.e., the 1^{st}, 10^{th}, and so on). In all, this sampling approach returns 36 functions from the first family, 27 from the second, 252 from the third, and 90 from the fourth. Then, I generate 500 simulated datasets from each of the sampled functions and apply the Goldilocks and Robin Hood algorithms to each.

4.1. Two Linear/Log-Linear Segments

I begin by replicating the results for the monotonic-increasing functions, starting with functions with two linear or log-linear segments for which the function is increasing for low values and flat for high values. The full list of 36 functions along with the estimated false rejection rates from the original simulations in Simonsohn (2018) is presented in Table 1 in section 6.

The functions vary according to:

- The distribution of x: Normal, Uniform, Beta with left skew, Beta with right skew, or Optimized for quadratic regression (see McClelland 1997, for discussion of the optimized distribution).
- Linear / Log-linear: For low values of x, $E(y|x) = x$ or $E(y|x) = log(x)$.
- Sample Size: 100, 200, or 500.
- Standard Deviation of the Noise: The noise is distributed $N(0, \sigma)$, where σ is 1×SD of de-noised y, 2×SD of de-noised y, or 3×SD of de-noised y.
- Break point: Break between positive slope and constant slope at 30^{th} or 50^{th} percentile.

Figure 11 is a scatter plot that compares the false rejection rate for the Robin Hood algorithm and the Goldilocks algorithm. The blue dashed line represents the boundary at which the false rejection rate for the two algorithms are equal and the gray dashed lines represent 5 percent false rejection rates. Across the 36 functions, the average false rejection rate for the Goldilocks algorithm is 4.0 percent, compared to 5.0 percent for the Robin Hood algorithm. 23 of the functions fall below the blue line, indicating the Robin Hood algorithm has a higher false rejection rate, compared with 13 that fall above the line. In all, the correlation of false rejection rates across the functions is 0.14.

Next, I present the 10 functions for which there is the largest difference in the false rejection rate in Figure 12. The top 5 panels in the figure are functions for which the Goldilocks algorithm outperforms the Robin Hood algorithm. The bottom 5 are functions for which Robin Hood outperforms Goldilocks. The Goldilocks algorithm consistently outperforms the Robin Hood algorithm when the standard deviation of the noise is elevated. The Robin Hood algorithm outperforms the Goldilocks algorithm when the standard deviation of the noise is low.

Figure 11 False Rejection Rates for Two Linear/Log-linear Segments

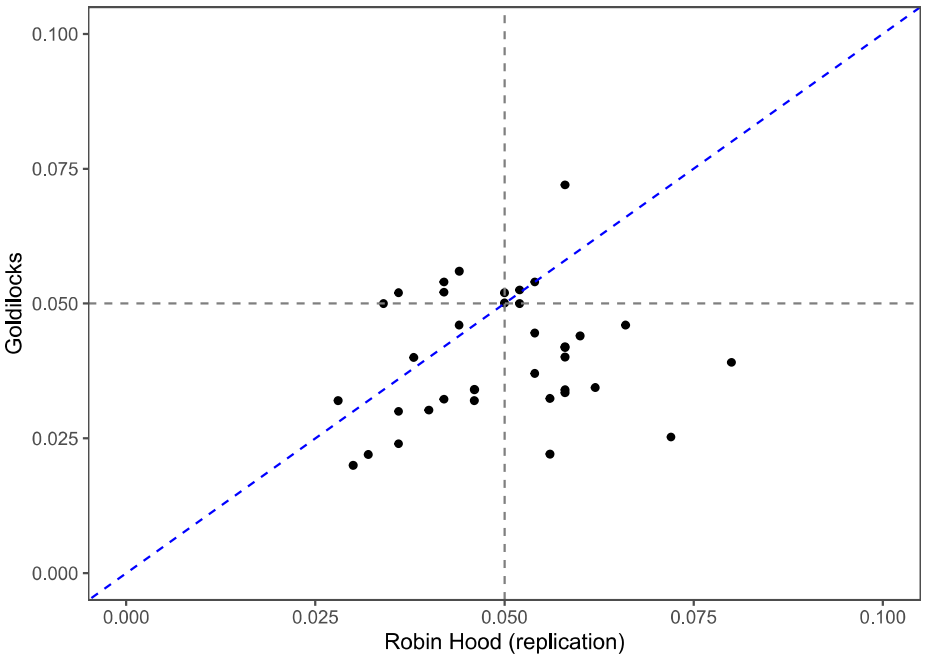
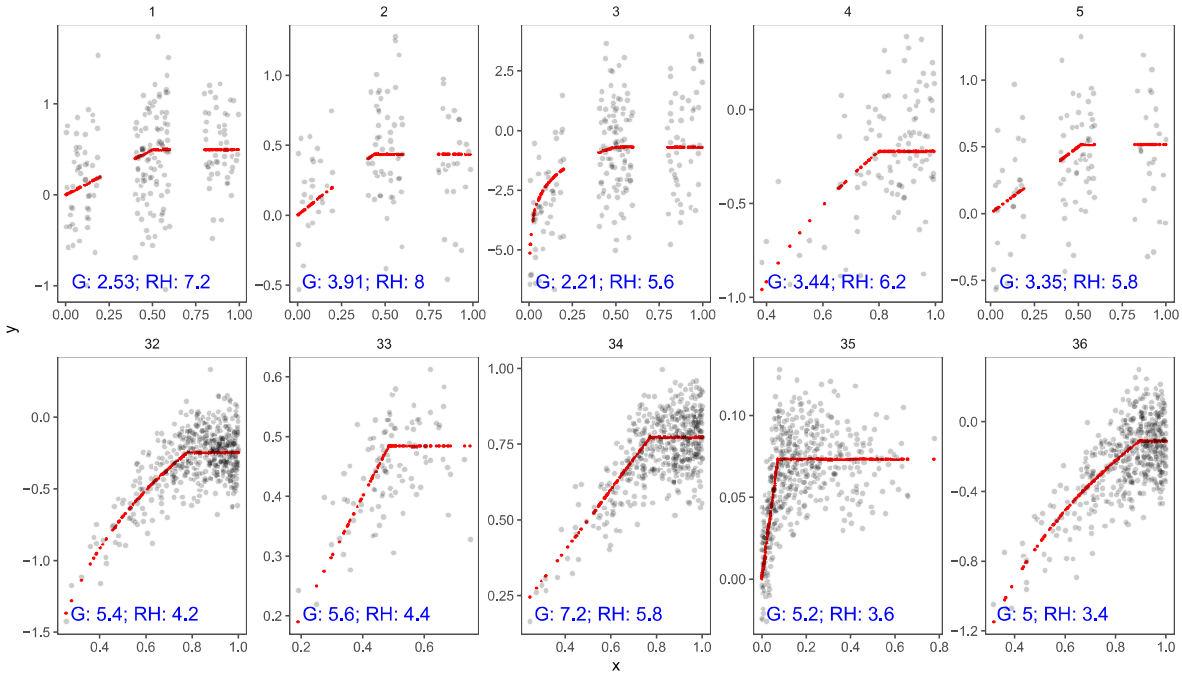


Figure 12 Cases with Largest Difference in False Rejection Rate, Two Linear/Log-linear Segments



4.2. Logistic Functions

I next replicate the results for logistic functions, which have the form $y = e^{-bx}$. The full list of 27 functions is presented in Table 2 in section 6.

The functions vary according to:

- The distribution of x: Normal, Uniform, Beta with left skew, Beta with right skew, or Optimized for quadratic regression.
- Sample Size: 100, 200, or 500.
- Standard Deviation of Noise: The noise is distributed $N(0, \sigma)$, where σ is 1×SD of de-noised y, 2×SD of de-noised y, or 3×SD of de-noised y.
- Slope parameter: b is 0.5, 1, or 1.5.[4]

Figure 13 replicates the analysis in Figure 11 for the logistic functions. Across the 27 functions, the average false rejection rate for the Goldilocks algorithm is 1.8 percent, compared with 2.2 percent for the Robin Hood algorithm. Further, the Goldilocks algorithm outperforms the Robin Hood algorithm for 18 of the 27 functions. The correlation between the false rejection rate for the two algorithms is 0.44 for the logistic functions, considerably higher than in the prior case.

Figure 13 False Rejection Rates for Logistic Functions

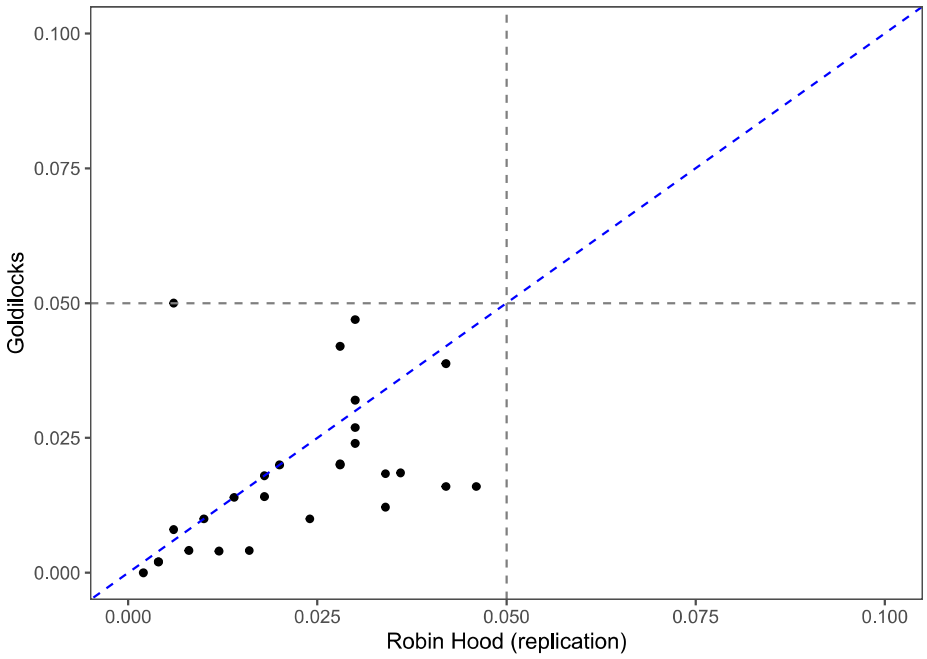
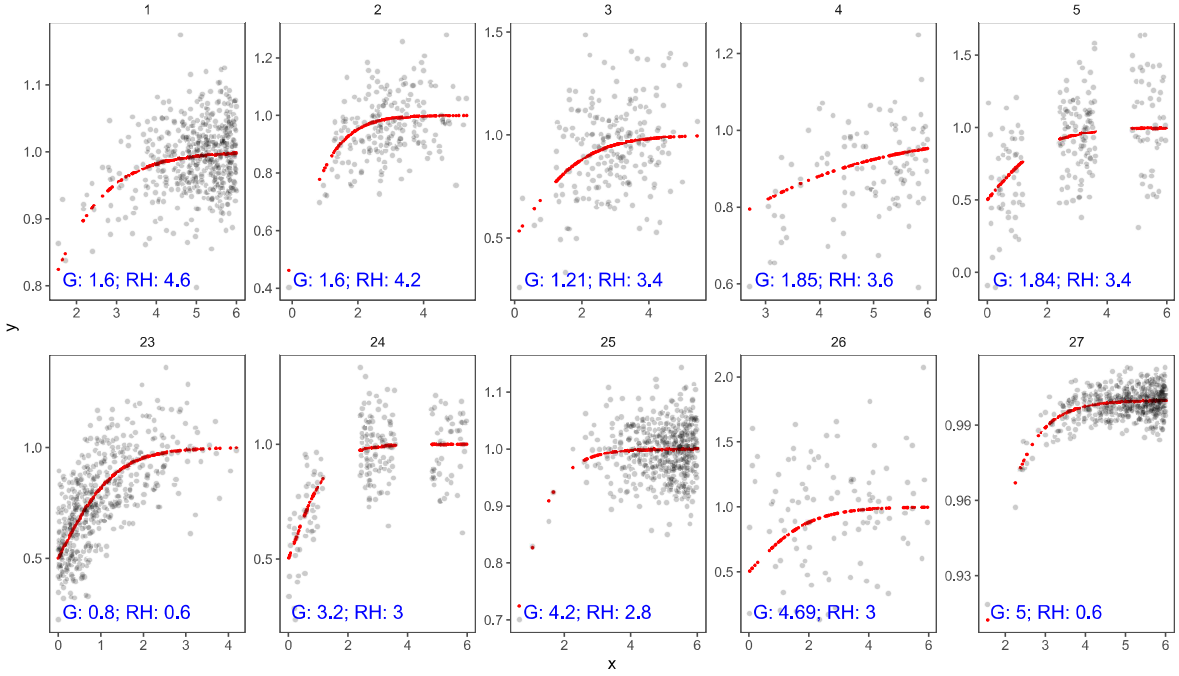


Figure 14 highlights the 10 functions for which the false rejection rate most differs. A pattern emerges when comparing the right-most plot on the bottom row to the right-most plot on the bottom row of Figure 12, two functions for which Robin Hood outperforms Golidlocks. In both, more data fall in the flat region of the function. For the Robin Hood algorithm, this suppresses the significance of the slope of the left segment in the initial interrupted regression with the break point set at the midpoint of the estimated flat maximum range. This leads to a relatively smaller

[4] The caption in Figure 7a in Simonsohn (2018) incorrectly lists the values of b as 0.5, 1.5, and 2.5.

adjustment of the break point than would be the case if the data were symmetrically distributed, ensuring that the function is not falsely classified as inverse u-shaped. However, as will become clear in the subsequent analyses, this also suppresses the power of the Robin Hood algorithm in some cases where the data are indeed inverse u-shaped.

Figure 14 Cases with Largest Difference in False Rejection Rate, Logistic Functions



4.3. Three Linear/Log-Linear Segments

Next, I present simulation results for data produced by the inverse u-shaped functional families. The first set of power analyses is for functions with three linear/log-linear segments. For these functions, the slope of $E(y|x)$ is increasing for low values of x and decreasing for high values of x and, in some cases, constant for intermediate values of x. The full list of 252 functions is presented in Table 3 through Table 6 in section 6.

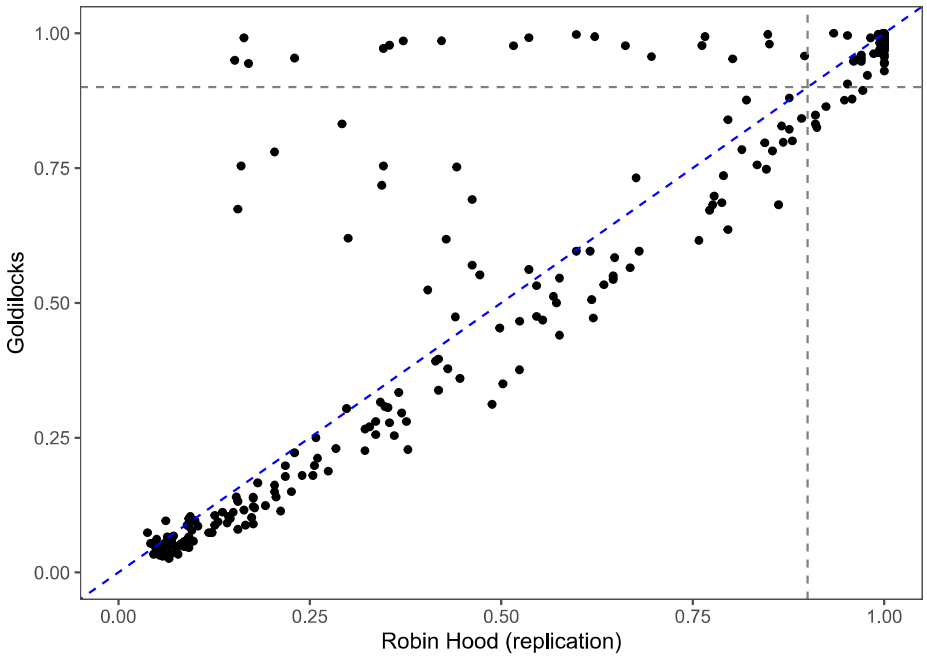
The functions vary according to:

- The distribution of x: Normal, Uniform, Beta with left skew, Beta with right skew, or Optimized for quadratic regression.
- Linear / Log-linear: For low values of x, $E(y|x) = x$ or $E(y|x) = log(x)$.
- Sample Size: 100, 200, or 500.
- Standard Deviation of the Noise: The noise is distributed $N(0, \sigma)$, where σ is 1×SD of de-noised y, 2×SD of de-noised y, or 3×SD of de-noised y.

- Break point 1: Break between positive slope and constant/negative slope at 30^{th} or 50^{th} percentile of x.
- Break point 2: Break between positive/constant slope and negative at 30^{th}, 50^{th}, 70^{th}, or 90^{th} percentile of x (where Break Point 1 is always less than or equal to Break Point 2)
- Magnitude of Change in Slope: For high values of x, the slope of $E(y|x)$ is -0.25, -0.5, -1, or -2 times the slope for low values of x.

I plot the power of each algorithm to detect an inverse u-shaped function in Figure 15. The blue dashed line represents the boundary at which the false rejection rate for the two algorithms are equal and the gray dashed lines represent 90 percent true classification rates. For many functions, the Goldilocks algorithm is slightly under-powered relative to the Robin Hood algorithm. However for a subset of the functions, the Goldilocks algorithm performs considerably better. As a result, the average power for the Goldilocks algorithm is 53.9 percent, compared with 52.0 percent for the Robin Hood algorithm. Although the Robin Hood algorithm outperforms the Goldilocks algorithm for 198 out of the 252 functions, the difference in power exceeds 10 percent for just 14 of the functions and the maximum difference is 18 percent. Goldilocks outperforms Robin Hood by more than 10 percent for 31 functions and by more than 25 percent for 22. The overall correlation in power to detect an inverse u-shaped function for the two algorithms is 0.89.

Figure 15 Power for Three Linear/Log-Linear Segments

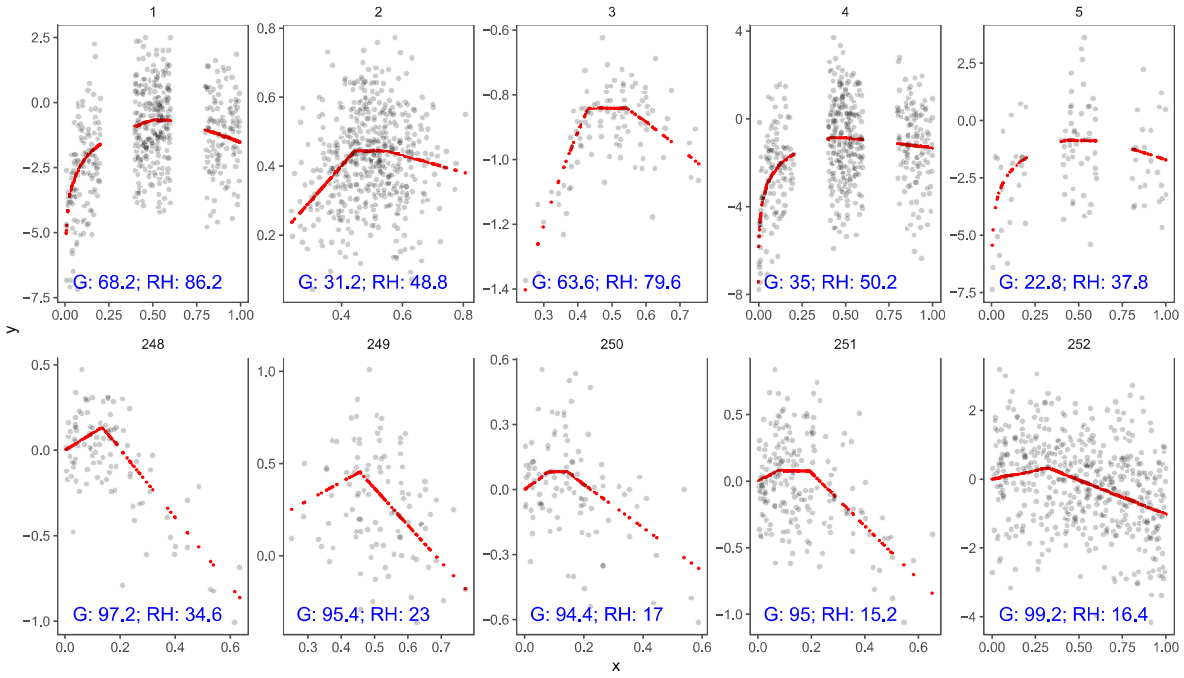


In Figure 16, I present the functions for which the two algorithms have the largest divergence in power. The functions for which Robin Hood outperforms Goldilocks in terms of power are similar

to those for which Goldilocks outperforms Robin Hood in terms of the false rejection rate, which is a classic trade-off in the design of classification algorithms. These functions have elongated flat or near-flat segments and high variance. As a result, they are settings where specification error is likely to occur in for the Golidlocks algorithm, largely due to the number and location of the knots in the linear spline functions.

Of particular interest, however, are the functions in the bottom panel, where Goldilocks so strongly outperforms Robin Hood. Here, we see the cost of the conservatism described earlier in the false rejection rate analysis. In settings where the majority of the observations in the sample is drawn from a weakly increasing region of the function and there are fewer observations from a strong decreasing region, the Robin Hood algorithm has difficulty identifying the function as inverse u-shaped.

Figure 16 Cases with Largest Difference in Power, Three Linear/Log-Linear Segments



4.4. $y = x - ax^k$ Functions

I last present the power analysis for the inverse u-shape functions of the form $y = x - ax^k$. The full list of 90 functions is presented in Table 7 and Table 8 in the appendix.

The functions vary according to:

- The distribution of x: Normal, Uniform, Beta with left skew, Beta with right skew, or Optimized for quadratic regression.
- Sample Size: 100, 200, or 500.

- Standard Deviation of Noise: The noise is distributed $N(0,\sigma)$, where σ is 1×SD of de-noised y, 2×SD of de-noised y, or 3×SD of de-noised y.
- Inflection point: The inflection point between negative slope and positive slope of $E(y|x)$ falls at the 50th, 60th, 70th, 80th, or 90th percentile of x.
- Exponent k: The exponent k takes values 2, 3, 4, or 5.

Figure 17 replicates the power analysis in Figure 15 for this family of functions. In this analysis, the Goldilocks algorithm more strongly outperforms the Robin Hood algorithm. The average power of the Goldilocks algorithm across the 90 functions is 52 percent, compared with 41 percent for the Robin Hood algorithm. As with the previous case, this divergence is largely driven by a subset of functions for which Goldilocks substantially outperforms Robin Hood. However, for the remainder, the two algorithms perform nearly equally well. The overall correlation for the power of the two tests across the functions is 0.74.

Figure 17 Power for $y = x - ax^k$ Functions

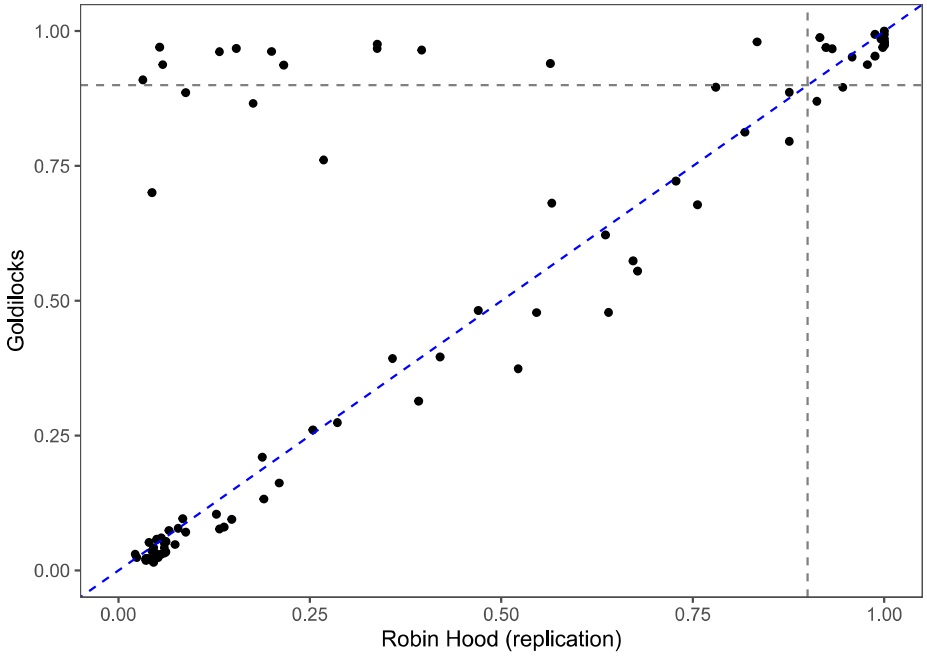
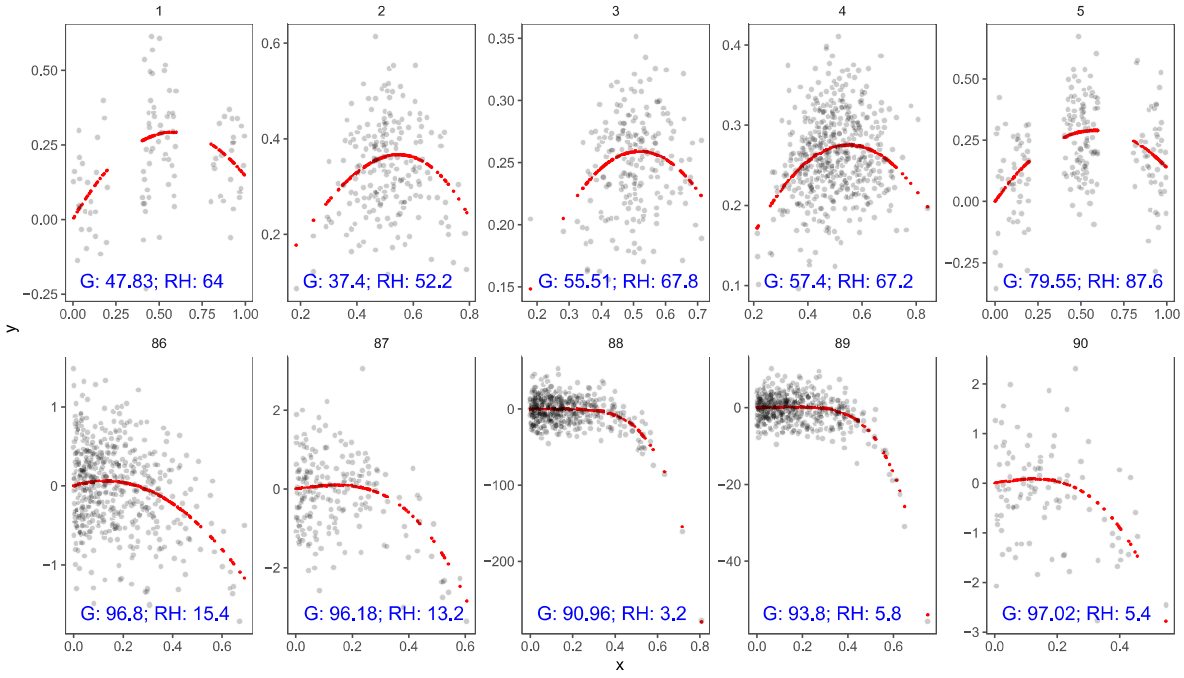


Figure 18 displays the functions for which the power of the two algorithms diverges most. As with Figure 16, the top row of the panel shows that the Robin Hood algorithm outperforms the Goldilocks algorithm for some functions with high variance errors. The bottom row similarly reinforces the conclusions drawn from the bottom row in Figure 16. The sample has a leftward skew for all five of the functions for which the Goldilocks algorithm most outperforms the Robin Hood algorithm and, for all five, the slope for low values of x is only weakly positive relative to the magnitude of the negative slope for high values of x.

Figure 18 Cases with Largest Difference in Power, $y = x - ax^k$ Functions



4.5. Goldilocks + Robin Hood: Ensemble Methods

The divergent approaches for evaluating the inverse u-shaped hypothesis create opportunities to use the Golidlocks and Robin Hood algorithms together in order to improve the false-rejection rate and power to identify inverse u-shaped data. In particular, the fairly low correlation in false-rejection rates across the 63 functional forms points to the potential to leverage the information of both tests in order to decrease the frequency with which monotonic-increasing data is incorrectly identified as inverse u-shaped.

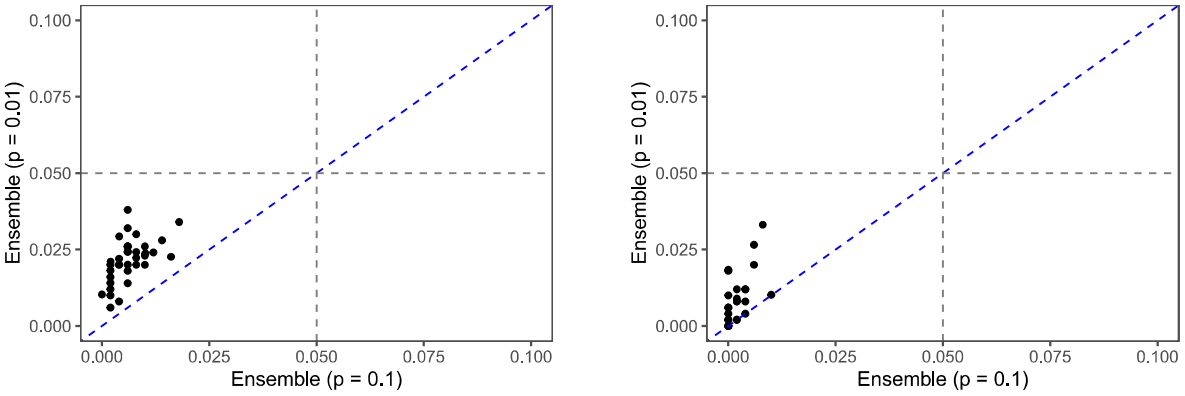
I explore this possibility with two ensemble methods. The first requires that both methods support the inverse u-shaped hypothesis at the 0.1-level. In this case, the increased false-rejection rate and increased power resulting from the higher significance level for any individual test is offset by the requirement that both methods agree in order to reject the null hypothesis. The second requires that either method support the inverse u-shaped hypothesis at the 0.01-level. In this case, the decreased false-rejection rate and decreased power resulting from the lower significance level for any individual test is offset by the requirement that just one method must reject the null in order to for inverse u-shaped hypothesis to be supported.

I begin by examining the false-rejection rates for the two ensemble methods across the 63 monotonic-increasing functional forms. Both control the false-rejection rate better than either Goldilocks or Robin Hood on their own. The first and second ensemble methods falsely identify the data as inverse u-shaped 0.07 percent and 1.6 of the time, respectively. The maximum false-rejection

rates for both ensemble methods—1.8 percent and 3.8 percent, respectively—are well below the 5 percent target. Figure 19 compares the two ensemble methods in terms of their false-rejection rates for all of the monotonic-increasing functional forms. Of note, although the first ensemble method (the horizontal axis) consistently outperforms the second (the vertical axis), the second continues to perform quite well across all the functional forms relative to the target 5 percent false-rejection rate.

Figure 19 False-rejection Rates for Monotonic-increasing Functions

(a) Two Linear/Log-Linear Segments (b) Logistic Functions



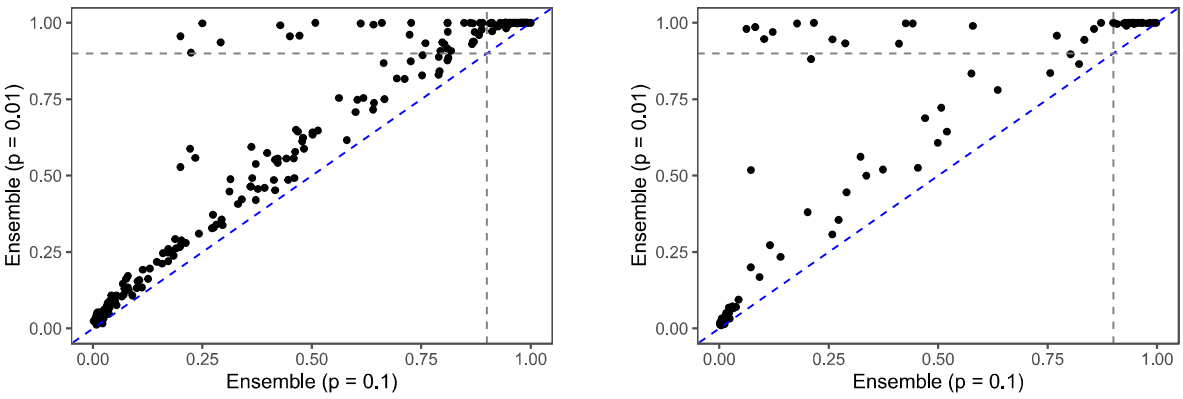
I next examine the power of the two ensemble methods to detect an inverse u-shaped function across the 342 functional forms. As demonstrated in Figure 20, the ability of the first ensemble method to control the false-positive rate comes at a high cost in terms of statistical power, which is indicated by the fact that all of the points in the figure lie above the blue dashed line. Note further that the power of the second ensemble method across these functional forms (53.3 percent) is nearly equivalent to the power of the Goldilocks algorithm on its own (53.4 percent), while also achieving a nearly 50 percent reduction in the false-rejection rate (3.1 percent vs. 1.6 percent). In all, the power of the second ensemble method exceeds 90 percent for 125 of the functional forms, compared with 92 for the second ensemble method and 110 for the Goldilocks algorithm on its own.

5. Discussion and Conclusion

This paper presents the Goldilocks algorithm as an alternative approach for evaluating u-shaped and inverse u-shaped hypotheses, which are exceedingly common throughout the social sciences and, specifically, in management theory and strategy. Rather than fitting a function to the sample and then conducting inference based on the estimated parameters, the Goldilocks algorithm identifies three potential shape-constrained functions *ex ante* and then evaluates the fit of the data to

Figure 20 Power for Inverse u-shaped Functions

(a) Three Linear/Log-Linear Segments (b) $y = x - ax^k$ Functions



those functions. In the implementation of the algorithm presented here, I evaluate the relative fit of three linear spline regressions—one constrained to be monotonic-increasing, one constrained to be inverse u-shaped, and one unconstrained—using the MCS procedure. One function is thus too flexible or "hot", one is too constrained or "cold", and one is "just right".

I then conduct a horse race to compare the performance of the Goldilocks algorithm to the Robin Hood algorithm, the latter of which has been demonstrated to have superior performance than existing methods for classifying inverse u-shaped functions in continuous data. Using a representative sample of the functional forms evaluated in Simonsohn (2018), I demonstrate that the Goldilocks algorithm achieves better control of the false rejection rate and better power across 405 functions in four functional families and, in all, over 200 thousand simulated samples. I then show that a combination of the two algorithms in an ensemble method—a "Robingold" method, perhaps—controls the false-rejection rate substantially better than either of two methods does individually and also produces very nearly equivalent power as the Goldilocks method. Based on the horse race, then, my recommendation for researchers evaluating inverse u-shaped theories in data is to apply both algorithms. If either support the inverse u-shaped claim at the 0.01-level, then this should be interpreted as support for the theory at the 0.05-level.

That being said, I am hopeful that this is just one more step forward in the continuing development of new research methods for evaluating shaped theories in data. I suspect that the implementation of the Goldilocks algorithm recommended here may not be the best one in terms of minimizing false rejection nor for maximizing power and look forward to future researchers discovering combinations of models and model selection algorithms that outperform mine. I also suspect that there are alternative implementations that are not quite so computationally costly that achieve roughly the same control of the false rejection rate and power. Further, while this paper demonstrates improved performance in a bi-variate setting using linear regression-based methods, it is

also possible that the relative performance of the algorithms could differ for multivariate or nonlinear models, which points to the desirability of expanding the scope of the simulation study beyond those functions evaluated in Simonsohn (2018) to include, e.g., control variables that are correlated with the independent variable of interest. To the extent that this paper sparks additional healthy competition among researchers proposing useful, whimsically-named algorithms, applied social science research will collectively be better off.

References

Aghion P, Bloom N, Blundell R, Griffith R, Howitt P (2005) Competition and Innovation: an Inverted-U Relationship. *The Quarterly Journal of Economics* 120(2):701–728, ISSN 0033-5533, 1531-4650, URL http://dx.doi.org/10.1093/qje/120.2.701.

Aven B, Morse L, Iorio A (2021) The valley of trust: The effect of relational strength on monitoring quality. *Organizational Behavior and Human Decision Processes* 166:179–193, ISSN 07495978, URL http://dx.doi.org/10.1016/j.obhdp.2019.07.004.

Bechler CJ, Tormala ZL, Rucker DD (2021) The Attitude–Behavior Relationship Revisited. *Psychological Science* 32(8):1285–1297, ISSN 0956-7976, 1467-9280, URL http://dx.doi.org/10.1177/0956797621995206.

Boyd SP, Vandenberghe L (2004) *Convex optimization* (Cambridge, UK ; New York: Cambridge University Press), ISBN 978-0-521-83378-3.

Cattaneo MD, Crump RK, Farrell MH, Feng Y (2019) On Binscatter. *arXiv:1902.09608 [econ, stat]* URL http://arxiv.org/abs/1902.09608, arXiv: 1902.09608.

Deephouse D (1999) To be different, or to be the same? It's a question (and theory) of strategic balance. *Strat. Mgmt. J.* 20(2):147–166.

Denrell J, Liu C (2021) When Reinforcing Processes Generate an Outcome-Quality Dip. *Organization Science* 32(4):1079–1099, ISSN 1047-7039, 1526-5455, URL http://dx.doi.org/10.1287/orsc.2020.1414.

Ganz SC (2020) Hypothesis testing sustained declines in COVID-19 intensity. Technical report, American Enterprise Institute, URL http://dx.doi.org/10.2307/resrep24601.

Haans RFJ, Pieters C, He ZL (2016) Thinking about U: Theorizing and testing U- and inverted U-shaped relationships in strategy research: Theorizing and Testing U-Shaped Relationships. *Strategic Management Journal* 37(7):1177–1195, ISSN 01432095, URL http://dx.doi.org/10.1002/smj.2399.

Hannan MT, Freeman J (1989) *Organizational Ecology* (Cambridge: Harvard Univ. Press).

Hansen PR, Lunde A, Nason JM (2011) The Model Confidence Set. *Econometrica* 79(2):453–497, ISSN 00129682, 14680262, URL www.jstor.org/stable/41057463.

Hastie T, Tibshirani R, Friedman JH (2009) *The elements of statistical learning: data mining, inference, and prediction.* Springer series in statistics (New York, NY: Springer), 2nd ed edition, ISBN 978-0-387-84857-0 978-0-387-84858-7.

Hothorn T, Möst L, Bühlmann P (2018) Most Likely Transformations. *Scandinavian Journal of Statistics* 45(1):110–134, ISSN 0303-6898, 1467-9469, URL http://dx.doi.org/10.1111/sjos.12291.

Laursen K, Salter A (2006) Open for innovation: the role of openness in explaining innovation performance among U.K. manufacturing firms. *Strategic Management Journal* 27(2):131–150, ISSN 0143-2095, 1097-0266, URL http://dx.doi.org/10.1002/smj.507.

Lind JT, Mehlum H (2010) With or Without U? The Appropriate Test for a U-Shaped Relationship*: Practitioners' Corner. *Oxford Bulletin of Economics and Statistics* 72(1):109–118, ISSN 03059049, 14680084, URL http://dx.doi.org/10.1111/j.1468-0084.2009.00569.x.

McClelland GH (1997) Optimal Design in Psychological Research. *Psychological methods* 2(1):3–19, ISSN 1082-989X, place: [Washington, D.C.] : Publisher: American Psychological Association,.

Phillips DJ, Zuckerman EW (2001) Middle-Status Conformity: Theoretical Restatement and Empirical Demonstration in Two Markets. *American Journal of Sociology* 107(2):379–429, ISSN 0002-9602, 1537-5390, URL http://dx.doi.org/10.1086/324072.

Shibata R (1997) Bootstrap Estimate of Kullback-Liebler Information for Model Selection. *Statistica Sinica* 7(2):375–394, ISSN 10170405, 19968507, URL www.jstor.org/stable/24306084.

Shin J, Grant AM (2019) Bored by Interest: How Intrinsic Motivation in One Task Can Reduce Performance on Other Tasks. *Academy of Management Journal* 62(2):415–436, ISSN 0001-4273, 1948-0989, URL http://dx.doi.org/10.5465/amj.2017.0735.

Simonsohn U (2018) Two Lines: A Valid Alternative to the Invalid Testing of U-Shaped Relationships With Quadratic Regressions. *Advances in Methods and Practices in Psychological Science* 1(4):538–555, ISSN 2515-2459, 2515-2467, URL http://dx.doi.org/10.1177/2515245918805755.

Wand J (2012) Testing Competing Theories with Shape Contrained Inference, URL https://www.jonathan-wand.org/pdf/wand-shapes-comparisons.pdf, mimeo.

6. Appendix

The appendix includes a selection of tables and figures references that are referenced in the body of the paper.

Table 1 List of Sampled Functions in Two Linear / Log-Linear Segments Family

Break point	SD multiplier	N	Distribution of x	Form of y	RH Classification Rate (original)
0.5	2	200	normal	log	0.0160
0.3	1	500	normal	log	0.0300
0.5	1	200	beta.rs	log	0.0300
0.5	1	500	beta.rs	log	0.0320
0.3	2	100	uniform	linear	0.0360
0.5	1	200	optimal	linear	0.0360
0.3	1	500	beta.rs	log	0.0380
0.3	3	100	beta.ls	linear	0.0400
0.5	1	200	beta.ls	log	0.0400
0.3	1	500	beta.ls	linear	0.0420
0.5	3	100	optimal	log	0.0420
0.5	2	100	normal	linear	0.0420
0.3	3	200	beta.ls	log	0.0440
0.3	1	500	beta.rs	linear	0.0440
0.5	3	100	beta.ls	linear	0.0440
0.5	1	100	normal	linear	0.0440
0.3	1	100	uniform	log	0.0460
0.3	2	500	beta.ls	linear	0.0480
0.3	1	200	beta.rs	linear	0.0480
0.5	1	200	normal	log	0.0480
0.3	3	500	optimal	log	0.0500
0.3	2	200	beta.rs	linear	0.0500
0.5	2	100	beta.ls	log	0.0500
0.3	2	500	beta.rs	linear	0.0520
0.5	3	100	uniform	linear	0.0520
0.5	2	100	optimal	linear	0.0520
0.3	2	100	beta.rs	log	0.0540
0.5	2	500	optimal	linear	0.0540
0.3	3	100	beta.rs	linear	0.0554
0.5	3	500	beta.rs	linear	0.0560
0.5	2	200	optimal	log	0.0560
0.3	2	200	beta.ls	linear	0.0580
0.5	3	200	optimal	linear	0.0580
0.5	1	100	beta.rs	log	0.0600
0.5	3	500	normal	log	0.0620
0.3	2	100	optimal	linear	0.0640

Table 2 List of Sampled Functions in Logistic Family

SD multiplier	N	Distribution of x	b	RH Classification Rate (original)
3	500	beta.ls	0.5	0.000
1	100	normal	0.5	0.000
3	200	normal	0.5	0.002
3	200	beta.ls	1.0	0.004
1	500	beta.ls	1.5	0.006
2	200	beta.ls	1.0	0.008
3	500	normal	1.0	0.012
1	200	normal	1.0	0.012
3	500	uniform	0.5	0.014
3	500	beta.rs	1.0	0.016
2	500	normal	1.0	0.018
1	500	beta.rs	1.5	0.018
2	500	beta.rs	0.5	0.022
1	200	optimal	0.5	0.022
2	200	optimal	1.0	0.024
3	100	uniform	1.5	0.026
3	200	normal	1.0	0.028
3	100	beta.rs	0.5	0.030
2	100	optimal	1.0	0.032
1	200	uniform	1.5	0.034
1	100	normal	1.5	0.036
3	500	normal	1.5	0.040
2	200	normal	1.5	0.042
1	200	optimal	1.5	0.044
3	100	optimal	0.5	0.048
3	100	uniform	1.0	0.050
3	500	beta.rs	1.5	0.054

Table 3 List of Sampled Functions in Three Linear / Log-Linear Segments Family

Break point 1	Break point 2	SD multiplier	N	Magitude of Change in Slope	Distribution of x	Form of y	RH Classification Rate (original)
0.3	0.9	3	100	0.25	uniform	log	0.0340
0.3	0.9	2	500	0.25	beta.rs	linear	0.0500
0.5	0.9	3	100	0.25	beta.rs	log	0.0520
0.5	0.7	3	100	0.50	beta.ls	log	0.0540
0.3	0.9	3	200	0.25	optimal	log	0.0560
0.3	0.9	3	100	0.50	optimal	log	0.0580
0.3	0.9	1	200	0.25	optimal	log	0.0600
0.3	0.9	3	100	2.00	optimal	log	0.0612
0.3	0.9	1	200	0.25	beta.rs	linear	0.0620
0.3	0.9	3	200	1.00	beta.rs	log	0.0632
0.5	0.9	2	100	1.00	beta.rs	log	0.0652
0.5	0.7	3	100	0.25	beta.ls	log	0.0660
0.5	0.9	1	100	1.00	uniform	log	0.0660
0.5	0.9	3	500	0.50	beta.rs	log	0.0676
0.5	0.9	3	100	0.25	optimal	linear	0.0680
0.5	0.9	3	100	1.00	beta.rs	linear	0.0696
0.5	0.9	3	100	1.00	beta.rs	log	0.0700
0.5	0.9	3	500	2.00	beta.rs	log	0.0716
0.3	0.9	2	100	0.50	optimal	log	0.0720
0.3	0.9	3	100	1.00	beta.rs	linear	0.0732
0.5	0.7	3	100	0.25	optimal	log	0.0740
0.5	0.9	2	200	0.50	beta.rs	linear	0.0748
0.5	0.9	3	500	1.00	uniform	log	0.0760
0.3	0.9	1	200	1.00	uniform	log	0.0772
0.3	0.9	2	200	0.50	beta.rs	log	0.0780
0.5	0.9	2	200	0.50	optimal	linear	0.0780
0.3	0.9	2	200	0.25	beta.rs	log	0.0800
0.5	0.9	2	200	0.25	optimal	linear	0.0800
0.5	0.9	3	200	0.25	beta.rs	log	0.0820
0.3	0.9	3	500	0.50	optimal	linear	0.0840
0.3	0.3	3	100	0.50	beta.ls	log	0.0860
0.5	0.9	1	200	0.25	uniform	linear	0.0860
0.5	0.9	3	500	0.50	optimal	linear	0.0880
0.3	0.9	1	500	0.50	uniform	log	0.0900
0.5	0.9	1	200	0.25	beta.rs	linear	0.0916
0.5	0.9	1	500	1.00	optimal	log	0.0920
0.5	0.9	3	200	1.00	beta.ls	log	0.0940
0.3	0.9	2	200	0.25	beta.ls	log	0.0980
0.3	0.9	3	200	1.00	optimal	linear	0.1000
0.3	0.9	2	500	0.50	uniform	log	0.1020
0.5	0.9	2	100	0.50	optimal	linear	0.1020
0.5	0.9	1	100	2.00	uniform	log	0.1040
0.5	0.9	3	500	0.50	beta.ls	log	0.1060
0.5	0.9	3	100	0.50	normal	log	0.1080
0.3	0.7	2	100	0.25	uniform	log	0.1120
0.5	0.9	1	200	0.50	optimal	linear	0.1120
0.5	0.7	3	100	0.25	normal	log	0.1140
0.3	0.7	3	200	0.50	uniform	log	0.1160
0.5	0.7	3	200	0.25	optimal	log	0.1180
0.5	0.7	3	100	0.25	optimal	linear	0.1200
0.3	0.7	2	100	0.50	uniform	log	0.1240
0.3	0.5	2	100	0.25	beta.ls	log	0.1280
0.5	0.9	3	200	2.00	optimal	linear	0.1280
0.3	0.3	2	100	0.25	beta.ls	log	0.1320
0.3	0.9	3	100	2.00	beta.ls	linear	0.1340
0.3	0.7	2	200	0.25	beta.ls	log	0.1360
0.5	0.9	2	200	2.00	uniform	linear	0.1372
0.3	0.3	3	100	1.00	uniform	linear	0.1420
0.5	0.5	2	500	0.25	beta.rs	linear	0.1440
0.3	0.7	2	200	0.50	beta.rs	linear	0.1460
0.5	0.7	2	200	0.50	beta.rs	log	0.1480
0.3	0.7	2	200	2.00	beta.ls	linear	0.1540

Table 4 List of Sampled Functions in Three Linear / Log-Linear Segments Family (cont'd)

Break point 1	Break point 2	SD multiplier	N	Magitude of Change in Slope	Distribution of x	Form of y	RH Classification Rate (original)
0.5	0.7	2	100	1.00	beta.rs	linear	0.1620
0.5	0.7	2	500	0.25	uniform	log	0.1660
0.5	0.7	3	100	2.00	beta.ls	linear	0.1680
0.5	0.9	1	500	0.50	beta.ls	log	0.1700
0.5	0.9	2	500	1.00	beta.ls	log	0.1740
0.5	0.7	2	500	0.50	beta.rs	linear	0.1780
0.5	0.5	3	500	0.50	beta.ls	log	0.1820
0.3	0.7	3	200	1.00	beta.ls	log	0.1860
0.5	0.9	3	100	1.00	normal	linear	0.1880
0.5	0.5	2	100	0.50	beta.rs	log	0.1920
0.3	0.5	2	200	0.25	beta.ls	log	0.1960
0.5	0.5	3	200	0.25	uniform	linear	0.2000
0.3	0.3	2	100	2.00	normal	linear	0.2040
0.5	0.9	2	500	2.00	uniform	linear	0.2040
0.5	0.9	1	100	1.00	uniform	linear	0.2080
0.5	0.9	3	500	0.50	normal	linear	0.2140
0.3	0.7	3	200	1.00	uniform	log	0.2180
0.5	0.5	2	100	0.25	optimal	linear	0.2200
0.5	0.9	2	100	1.00	normal	log	0.2240
0.3	0.7	2	100	1.00	beta.rs	log	0.2320
0.5	0.5	3	200	0.50	optimal	log	0.2340
0.3	0.5	1	100	0.25	beta.ls	log	0.2380
0.3	0.3	3	100	0.50	beta.rs	linear	0.2440
0.5	0.7	3	500	0.50	beta.rs	log	0.2460
0.3	0.7	2	500	0.50	beta.rs	linear	0.2500
0.3	0.5	3	200	0.50	beta.rs	linear	0.2540
0.3	0.7	1	200	0.25	uniform	log	0.2600
0.3	0.3	3	100	2.00	uniform	log	0.2660
0.5	0.9	1	200	2.00	uniform	linear	0.2700
0.5	0.7	3	500	2.00	uniform	log	0.2748
0.3	0.7	3	200	0.25	normal	log	0.2780
0.5	0.5	3	100	2.00	optimal	linear	0.2820
0.3	0.9	3	100	1.00	beta.ls	linear	0.2880
0.3	0.5	3	500	0.50	beta.ls	log	0.2940
0.3	0.7	3	100	2.00	normal	linear	0.2980
0.3	0.5	3	500	0.25	optimal	log	0.3040
0.3	0.7	3	100	2.00	beta.rs	linear	0.3120
0.3	0.7	2	200	0.25	beta.rs	log	0.3168
0.5	0.5	1	500	0.25	beta.rs	linear	0.3200
0.5	0.5	1	100	2.00	beta.ls	linear	0.3240
0.3	0.7	2	100	1.00	optimal	log	0.3340
0.5	0.9	3	200	1.00	normal	linear	0.3380
0.3	0.7	2	500	0.50	beta.ls	log	0.3460
0.5	0.9	1	100	0.50	normal	linear	0.3500
0.3	0.7	3	500	1.00	beta.rs	log	0.3560
0.5	0.5	2	500	0.25	optimal	log	0.3600
0.3	0.5	3	200	0.50	beta.ls	linear	0.3640
0.5	0.7	1	500	0.25	beta.ls	log	0.3660
0.3	0.7	3	500	0.25	beta.ls	linear	0.3704
0.3	0.5	2	200	0.50	optimal	log	0.3780
0.3	0.5	2	500	1.00	beta.ls	linear	0.3840
0.5	0.5	2	100	1.00	beta.rs	linear	0.3900
0.3	0.5	2	100	2.00	normal	linear	0.3960
0.5	0.7	3	100	2.00	optimal	log	0.4020
0.5	0.5	1	200	0.50	beta.ls	log	0.4100
0.5	0.9	2	200	0.25	beta.ls	linear	0.4160
0.3	0.7	1	200	0.50	uniform	log	0.4200
0.3	0.3	3	100	1.00	optimal	linear	0.4260
0.3	0.3	3	100	0.50	normal	log	0.4340
0.3	0.5	2	500	0.25	optimal	log	0.4380
0.3	0.5	3	100	1.00	normal	log	0.4460
0.5	0.7	3	100	2.00	normal	linear	0.4540
0.5	0.5	3	100	2.00	beta.rs	linear	0.4620

Table 5 List of Sampled Functions in Three Linear / Log-Linear Segments Family (cont'd)

Break point 1	Break point 2	SD multiplier	N	Magitude of Change in Slope	Distribution of x	Form of y	RH Classification Rate (original)
0.3	0.7	3	500	0.25	normal	linear	0.4720
0.5	0.7	1	100	0.25	optimal	linear	0.4800
0.3	0.3	3	100	2.00	optimal	log	0.4900
0.3	0.3	2	100	0.50	beta.rs	linear	0.4940
0.5	0.9	1	500	1.00	uniform	linear	0.4980
0.3	0.3	2	100	0.25	normal	linear	0.5100
0.3	0.3	3	200	1.00	optimal	log	0.5140
0.5	0.5	1	100	1.00	beta.ls	log	0.5200
0.3	0.3	2	200	1.00	uniform	linear	0.5300
0.3	0.3	2	200	2.00	optimal	linear	0.5380
0.3	0.5	2	200	1.00	beta.ls	log	0.5440
0.5	0.9	2	200	0.25	normal	linear	0.5508
0.3	0.3	3	500	2.00	uniform	linear	0.5564
0.5	0.7	3	200	0.25	beta.ls	linear	0.5660
0.5	0.9	2	500	1.00	normal	log	0.5720
0.5	0.7	1	200	0.25	beta.rs	linear	0.5804
0.5	0.7	3	500	1.00	optimal	log	0.5840
0.5	0.7	1	200	1.00	uniform	log	0.5920
0.3	0.5	3	200	1.00	beta.rs	linear	0.6000
0.3	0.3	1	100	0.50	beta.ls	linear	0.6140
0.3	0.3	3	200	2.00	beta.rs	linear	0.6200
0.3	0.9	1	100	1.00	normal	log	0.6248
0.3	0.5	2	200	2.00	normal	linear	0.6320
0.5	0.7	1	500	0.50	beta.rs	log	0.6400
0.3	0.9	2	500	0.50	normal	linear	0.6440
0.3	0.5	1	100	0.50	beta.rs	linear	0.6500
0.3	0.5	2	200	0.25	normal	linear	0.6620
0.5	0.7	2	100	1.00	uniform	linear	0.6800
0.5	0.5	2	500	0.50	beta.rs	log	0.6860
0.5	0.7	1	200	0.25	optimal	linear	0.7040
0.3	0.3	1	200	0.50	beta.ls	log	0.7160
0.3	0.5	2	100	2.00	normal	log	0.7240
0.3	0.7	1	100	0.25	normal	log	0.7300
0.3	0.3	2	200	2.00	normal	log	0.7380
0.3	0.3	1	100	1.00	uniform	linear	0.7460
0.5	0.5	1	500	0.25	uniform	log	0.7520
0.5	0.5	1	200	0.50	beta.rs	log	0.7620
0.3	0.9	2	200	0.25	beta.ls	linear	0.7752
0.3	0.3	3	500	0.25	beta.rs	log	0.7840
0.5	0.7	2	100	2.00	optimal	linear	0.7920
0.3	0.7	2	500	0.25	optimal	linear	0.7980
0.3	0.7	3	200	2.00	normal	log	0.8080
0.3	0.5	3	500	1.00	optimal	log	0.8160
0.3	0.5	3	500	2.00	optimal	linear	0.8200
0.5	0.5	2	200	2.00	normal	linear	0.8240
0.5	0.5	3	200	2.00	optimal	log	0.8300
0.3	0.3	2	100	1.00	uniform	log	0.8340
0.5	0.5	1	100	1.00	beta.rs	linear	0.8380
0.3	0.3	3	500	0.25	normal	linear	0.8440
0.5	0.7	2	500	1.00	optimal	log	0.8520
0.3	0.5	2	200	1.00	uniform	log	0.8620
0.5	0.9	2	500	0.50	normal	linear	0.8680
0.5	0.7	3	200	1.00	optimal	linear	0.8720
0.5	0.9	2	500	1.00	normal	linear	0.8764
0.3	0.7	3	500	0.50	uniform	linear	0.8840
0.5	0.5	3	500	0.50	uniform	linear	0.8880
0.3	0.5	2	500	0.25	optimal	linear	0.8920
0.5	0.9	3	500	0.50	beta.ls	linear	0.8960
0.5	0.7	3	500	0.25	beta.ls	linear	0.9000
0.3	0.7	2	200	0.50	optimal	linear	0.9060
0.5	0.9	3	500	2.00	beta.ls	linear	0.9100
0.5	0.7	3	500	1.00	normal	log	0.9180

Table 6 List of Sampled Functions in Three Linear / Log-Linear Segments Family (cont'd)

Break point 1	Break point 2	SD multiplier	N	Magitude of Change in Slope	Distribution of x	Form of y	RH Classification Rate (original)
0.3	0.5	3	500	1.00	beta.rs	linear	0.9240
0.5	0.9	1	200	0.50	normal	linear	0.9292
0.3	0.3	1	200	1.00	uniform	linear	0.9360
0.3	0.7	1	200	0.25	normal	log	0.9400
0.3	0.5	1	200	1.00	beta.ls	log	0.9480
0.5	0.5	1	100	2.00	uniform	linear	0.9520
0.5	0.7	1	100	0.25	beta.ls	linear	0.9580
0.5	0.5	1	500	0.50	uniform	log	0.9640
0.3	0.5	1	200	0.50	beta.ls	linear	0.9680
0.3	0.7	1	500	0.50	optimal	log	0.9700
0.3	0.7	1	100	2.00	normal	linear	0.9720
0.3	0.3	2	200	0.25	uniform	linear	0.9760
0.5	0.7	1	500	0.25	optimal	linear	0.9780
0.3	0.7	2	500	0.50	normal	log	0.9820
0.3	0.3	1	500	2.00	normal	linear	0.9840
0.5	0.5	1	500	0.50	beta.rs	log	0.9840
0.5	0.5	1	100	0.50	normal	linear	0.9860
0.3	0.5	1	200	0.25	normal	linear	0.9880
0.3	0.5	2	500	1.00	beta.rs	log	0.9900
0.3	0.3	2	500	1.00	normal	linear	0.9920
0.5	0.7	3	500	1.00	normal	linear	0.9928
0.3	0.5	1	200	0.25	normal	log	0.9940
0.3	0.3	2	500	0.25	normal	log	0.9960
0.3	0.7	2	500	2.00	normal	linear	0.9960
0.3	0.3	3	500	0.50	optimal	linear	0.9980
0.3	0.3	1	100	1.00	uniform	log	0.9980
0.3	0.7	2	500	0.50	optimal	linear	0.9980
0.5	0.5	3	500	2.00	optimal	log	0.9980
0.5	0.7	2	500	0.25	beta.ls	linear	0.9980
0.3	0.3	3	500	1.00	beta.rs	log	0.9988
0.3	0.3	2	500	0.50	uniform	linear	1.0000
0.3	0.3	2	500	2.00	optimal	log	1.0000
0.3	0.3	1	500	0.50	uniform	linear	1.0000
0.3	0.3	1	500	1.00	beta.ls	log	1.0000
0.3	0.3	1	500	2.00	optimal	linear	1.0000
0.3	0.3	1	200	0.50	optimal	linear	1.0000
0.3	0.3	1	200	2.00	beta.ls	log	1.0000
0.3	0.3	1	100	2.00	optimal	log	1.0000
0.3	0.5	2	500	1.00	optimal	linear	1.0000
0.3	0.5	1	500	0.25	uniform	linear	1.0000
0.3	0.5	1	500	1.00	normal	linear	1.0000
0.3	0.5	1	500	2.00	normal	log	1.0000
0.3	0.5	1	200	0.50	uniform	linear	1.0000
0.3	0.5	1	200	2.00	uniform	log	1.0000
0.3	0.5	1	100	2.00	beta.rs	linear	1.0000
0.3	0.7	2	500	1.00	uniform	linear	1.0000
0.3	0.7	1	500	0.50	normal	log	1.0000
0.3	0.7	1	500	2.00	normal	log	1.0000
0.3	0.7	1	200	1.00	normal	linear	1.0000
0.3	0.7	1	100	1.00	normal	linear	1.0000
0.3	0.9	1	500	1.00	normal	linear	1.0000
0.5	0.5	2	500	0.25	beta.ls	linear	1.0000
0.5	0.5	2	500	2.00	beta.rs	linear	1.0000
0.5	0.5	1	500	0.50	uniform	linear	1.0000
0.5	0.5	1	500	1.00	optimal	linear	1.0000
0.5	0.5	1	500	2.00	optimal	log	1.0000
0.5	0.5	1	200	2.00	normal	linear	1.0000
0.5	0.5	1	100	1.00	uniform	linear	1.0000
0.5	0.7	2	500	1.00	normal	linear	1.0000
0.5	0.7	1	500	0.50	normal	log	1.0000
0.5	0.7	1	500	2.00	normal	log	1.0000
0.5	0.7	1	200	1.00	normal	linear	1.0000
0.5	0.7	1	100	1.00	normal	linear	1.0000
0.5	0.9	1	500	0.50	beta.ls	linear	1.0000

Table 7 List of Sampled Functions in $y = x - ax^k$ Family

Inflection Point	Exponent k	SD multiplier	N	Distribution of x	RH Classification Rate (original)
0.8	2	1	500	beta.rs	0.018
0.5	5	1	500	beta.ls	0.024
0.9	4	1	200	beta.rs	0.026
0.8	2	2	500	optimal	0.030
0.9	5	2	200	beta.rs	0.030
0.9	2	1	100	uniform	0.032
0.8	4	2	100	beta.rs	0.034
0.9	5	1	100	beta.rs	0.034
0.9	3	2	200	beta.rs	0.036
0.9	3	3	200	optimal	0.038
0.8	3	2	200	optimal	0.040
0.6	5	1	500	beta.ls	0.042
0.5	5	3	100	beta.ls	0.044
0.7	2	3	500	beta.rs	0.046
0.8	2	3	500	optimal	0.048
0.7	4	3	500	beta.rs	0.050
0.9	5	3	100	beta.rs	0.050
0.9	5	2	200	uniform	0.052
0.9	5	2	100	optimal	0.054
0.8	4	3	200	beta.rs	0.056
0.8	4	2	100	optimal	0.058
0.8	2	2	200	uniform	0.060
0.7	2	2	500	beta.rs	0.062
0.7	4	2	500	beta.rs	0.064
0.7	5	2	500	beta.rs	0.066
0.8	2	1	100	uniform	0.068
0.9	2	2	100	normal	0.070
0.5	3	2	100	beta.ls	0.076
0.8	3	2	100	uniform	0.080
0.8	2	1	200	uniform	0.086
0.7	5	3	500	beta.ls	0.090
0.7	4	1	500	beta.rs	0.096
0.8	5	3	200	uniform	0.102
0.7	2	3	200	uniform	0.110
0.8	5	2	100	uniform	0.116
0.5	2	2	200	beta.rs	0.124
0.6	3	2	200	beta.ls	0.128
0.5	3	3	200	beta.rs	0.142
0.5	2	3	500	beta.ls	0.152
0.8	2	2	100	normal	0.164
0.8	5	3	200	beta.ls	0.180
0.5	5	3	200	uniform	0.192
0.9	4	1	100	normal	0.208
0.5	2	1	100	beta.rs	0.232
0.6	2	3	200	beta.ls	0.250

Table 8 List of Sampled Functions in $y = x - ax^k$ Family (cont'd)

Inflection Point	Exponent k	SD multiplier	N	Distribution of x	RH Classification Rate (original)
0.6	4	1	500	beta.rs	0.264
0.9	5	1	100	normal	0.282
0.5	5	3	500	uniform	0.298
0.6	4	3	100	optimal	0.322
0.7	5	3	100	normal	0.352
0.5	4	3	200	optimal	0.362
0.6	3	3	100	uniform	0.380
0.7	3	2	200	beta.ls	0.402
0.9	3	2	100	beta.ls	0.426
0.9	3	1	500	normal	0.456
0.7	3	3	200	normal	0.488
0.8	4	3	500	normal	0.522
0.9	5	3	200	beta.ls	0.548
0.8	4	2	200	beta.ls	0.586
0.7	2	2	100	optimal	0.606
0.9	3	2	200	beta.ls	0.642
0.7	2	3	500	normal	0.662
0.6	2	3	200	normal	0.688
0.9	3	1	100	beta.ls	0.718
0.6	2	3	500	uniform	0.742
0.5	3	2	100	normal	0.766
0.5	2	2	100	normal	0.798
0.5	4	2	500	uniform	0.828
0.9	5	2	200	beta.ls	0.858
0.7	2	2	200	optimal	0.884
0.8	2	1	500	normal	0.904
0.9	3	1	200	beta.ls	0.916
0.5	4	3	500	normal	0.926
0.7	2	1	100	beta.ls	0.940
0.8	3	1	100	beta.ls	0.954
0.6	3	2	200	normal	0.966
0.7	5	3	500	normal	0.974
0.6	2	2	200	optimal	0.984
0.5	2	2	200	uniform	0.988
0.5	3	1	100	optimal	0.994
0.6	3	2	500	uniform	0.996
0.5	4	1	500	uniform	0.998
0.8	3	1	200	beta.ls	0.998
0.5	2	1	200	uniform	1.000
0.5	3	1	200	optimal	1.000
0.6	2	1	200	optimal	1.000
0.6	4	2	500	normal	1.000
0.6	5	1	200	normal	1.000
0.7	3	1	500	optimal	1.000
0.7	5	1	500	normal	1.000