The Stata Journal (2025)
25, Number 1, pp. 3–50 DOI: 10.1177/1536867X251322960

Binscatter regressions

Matias D. Cattaneo
Princeton University
Princeton, NJ
cattaneo@princeton.edu

Richard K. Crump
Federal Reserve Bank of New York
New York, NY
richard.crump@ny.frb.org

Max H. Farrell
University of California Santa Barbara
Santa Barbara, CA
mhfarrell@gmail.com

Yingjie Feng
Tsinghua University
Beijing, China
fengyj@sem.tsinghua.edu.cn

Abstract. In this article, we introduce the package `binsreg`, which implements the binscatter methods developed by Cattaneo et al. (2024a, arXiv:2407.15276 [stat.EM]; 2024b, *American Economic Review* 114: 1488–1514). The package comprises seven commands: `binsreg`, `binslogit`, `binsprobit`, `binsqreg`, `binstest`, `binspwc`, and `binsregselect`. The first four commands implement binscatter plotting, point estimation, and uncertainty quantification (confidence intervals and confidence bands) for least-squares linear binscatter regression (`binsreg`) and for nonlinear binscatter regression (`binslogit` for logit regression, `binsprobit` for probit regression, and `binsqreg` for quantile regression). The next two commands focus on pointwise and uniform inference: `binstest` implements hypothesis testing procedures for parametric specifications and for nonparametric shape restrictions of the unknown regression function, while `binspwc` implements multigroup pairwise statistical comparisons. The last command, `binsregselect`, implements data-driven number-of-bins selectors. The commands offer binned scatterplots and allow for covariate adjustment, weighting, clustering, and multisample analysis, which is useful when studying treatment-effect heterogeneity in randomized and observational studies, among many other features.

Keywords: st0765, binsreg, binslogit, binsprobit, binsqreg, binstest, binspwc, binsregselect, binscatter, binned scatterplot, nonparametrics, semiparametrics, partitioning estimators, B-splines, tuning parameter selection, confidence bands, shape and specification testing

1 Introduction

Data visualization is a crucial step in any statistical analysis. The classic scatterplot is a fundamental visualization tool used for studying how an outcome y relates to a continuous covariate of interest x. However, in "big data" settings such as the administrative datasets now common in social science and medical research, the classic scatterplot yields a dense, uninformative cloud of points. Furthermore, there is no way to create the plot rigorously controlling for other covariates, which would be standard in any subsequent statistical analysis. A binned scatterplot, or binscatter, is a visualization method that addresses these limitations. Binscatter techniques offer flexible yet parsimonious ways of visualizing and summarizing regression (and other) functions. Binscatters have become popular in applied microeconomics for visualization, specification testing, treatment-effect heterogeneity, and other uses. See Starr and Goldfarb (2020) and references therein for current usage.

However, little was known about the statistical properties of binscatter until recently. Cattaneo, Crump, Farrell, and Feng (2024a,b, collectively CCFF hereafter) provided the first foundational and comprehensive analysis of binscatter methods, including an array of theoretical and practical results that aid in both understanding current practices (that is, validity or lack thereof) and offering theory-based guidance for future use. Maintaining rigor and statistical uncertainty is crucial for trustworthy data visualization in scientific settings (Healy 2019; Schwabish 2021).

This article introduces the package `binsreg`, which comprises seven commands implementing the main methodological results in CCFF. These commands are organized as follows.

- *Estimation, uncertainty quantification, and plotting.* The command `binsreg` implements canonical and extended least-squares binscatter methods, while the commands `binslogit`, `binsprobit`, and `binsqreg` implement generalized nonlinear binscatter methods (that is, logistic regression, probit regression, and quantile regression, respectively). All commands allow for higher-order polynomial fits within bins, smoothness restrictions across bins, and covariate adjustment for estimation, uncertainty quantification, and plotting, also covering higher-order derivatives and related partial effects of interest in linear and nonlinear settings, including multisample comparisons.

- *Hypothesis testing and statistical inference.* The command `binstest` implements hypothesis testing procedures for parametric specifications and for nonparametric shape restrictions of the unknown regression function, while `binspwc` implements multigroup pairwise comparisons. These two commands offer the same flexibility and features as the estimation commands and therefore allow for linear and nonlinear binscatter methods with within-bin higher-order polynomial fits, across-bins smoothness restrictions, and semilinear covariate adjustments, among several other features and options.

- *Optimal number of bins selection.* The six commands above take as input the binning scheme to construct the binscatter approximation, which requires selecting the position of the bins and the total number of bins on the support of the independent variable of interest. Whenever this information is not provided, the command `binsregselect` implements data-driven selectors for the number of bins for implementation using either quantile-spaced or evenly spaced binning or partitioning (quantile-spaced binning is chosen by default, following popular empirical practice).

The seven commands in the package `binsreg` offer several other important functionalities for empirical work. First, the commands incorporate mass-point and degrees-of-freedom checks and adjustments by default, which improves the stability of the implementation. Second, the commands allow for multiway fixed-effects and clustering estimation and inference whenever available in the underlying statistical software platform. Third, in Stata, the commands offer the option of estimation and inference with multiway fixed effects and multiway clustering via the community-contributed package `reghdfe` (Correia and Constantine 2014) and also allow for using the community-contributed package `gtools` (Caceres 2017), instead of our internal implementations, to potentially increase the speed of internal computations with ultralarge datasets. Depending on the data size and structure, the commands in the package `binsreg` may improve implementation execution speed when 1) mass-point and degrees-of-freedom checks are turned off or 2) the community-contributed packages `reghdfe` and `gtools` instead of our internal (open-source) implementations are used (in Stata). See section 2.9 for more discussion.

Two other community-contributed commands also implement binscatter methods: `binscatter` (Stepner 2013) and `binscatter2` (Droste 2019). Both of those packages incorporate other covariates or fixed effects incorrectly, yielding invalid results. Even without additional controls, those two packages only give the binned scatterplot for piecewise constant estimation using least-squares regression, lacking all the theoretically founded features of `binsreg` such as nonlinear models, valid uncertainty visualization, formal specification and shape testing, group comparisons, and optimal binning selection. See CCFF for further discussion.

The rest of the article is organized as follows. Section 2 gives an overview of the main methods available in the package `binsreg` and discusses some implementation details. Section 3 gives a numerical illustration. Section 4 concludes. The software help files contain a detailed description of all available options.

2 Methods overview and implementation details

This section summarizes the main methods implemented in the package `binsreg`. For further methodological and theoretical details, see CCFF.

Given a random sample $(y_i, x_i, \mathbf{w}_i')$, $i = 1, 2, \ldots, n$, where y_i is a scalar response variable, x_i is the scalar independent variable of interest, and $\mathbf{w}_i$ is a d-dimensional vector of additional covariates, a binscatter seeks to flexibly approximate the function

$$\vartheta_{\mathbf{w}}^{(v)}(x) = \frac{\partial^v}{\partial x^v} \eta \left\{ \mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0 \right\} \tag{1}$$

where $\mathbf{w}$ is some user-chosen evaluation point and the underlying parameters $\mu_0(\cdot)$ and $\boldsymbol{\gamma}_0$ are defined by

$$(\mu_0(\cdot), \boldsymbol{\gamma}_0) = \underset{\mu \in \mathcal{M}, \boldsymbol{\gamma} \in \mathbb{R}^d}{\operatorname{argmin}} \; \mathbb{E} \left(\rho \left[y_i; \eta \left\{ \mu(x_i) + \mathbf{w}_i'\boldsymbol{\gamma} \right\} \right] \right) \tag{2}$$

with $\rho(\cdot;\cdot)$ and $\eta(\cdot)$ being user-chosen loss and (inverse) link functions, respectively, and $\mathcal{M}$ being an appropriate space of functions satisfying certain conditions. Several settings of applied interest are covered by this formulation. [For any function $f(x)$, we define $f^{(v)}(x) = d^v f(x)/dx^v$, with the usual notation $f(x) = f^{(0)}(x)$.]

- *Semilinear regression*: $\rho(y;\eta) = (y-\eta)^2$ and $\eta(u) = u$. The parameter of interest becomes

$$\vartheta^{(v)}_{\mathbf{w}}(x) = \frac{\partial^v}{\partial x^v}\mathbb{E}(y_i|x_i = x, \mathbf{w}_i = \mathbf{w}) = \begin{cases} \mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0 & \text{if } v = 0 \\ \mu_0^{(v)}(x) & \text{if } v \geq 1 \end{cases}$$

  For example, $\vartheta_{\mathbf{w}}(x)$ [respectively, $\vartheta^{(1)}_{\mathbf{w}}(x)$] corresponds to the average (partial) effect of x on y for level $\mathbf{w}_i = \mathbf{w}$. In this setting, $\vartheta^{(v)}_{\mathbf{0}}(x) = \mu_0^{(v)}(x)$, which may be of interest in some applications.

- *Logistic/probit regression*: $\rho(y;\eta) = -y\log\eta - (1-y)\log(1-\eta)$ and $\eta(u)$ denotes the (inverse) link function of logistic or probit regression. The parameter of interest becomes

$$\vartheta^{(v)}_{\mathbf{w}}(x) = \frac{\partial^v}{\partial x^v}\mathbb{E}(y_i|x_i = x, \mathbf{w}_i = \mathbf{w}) = \frac{\partial^v}{\partial x^v}\eta\left\{\mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0\right\}$$

  which coincides with the usual average (partial) effect in binary response models.

- *Quantile regression*: $\rho(y;\eta) = \ell_\tau(y-\eta)$ and $\eta(u) = u$, where $\ell_\tau(u)$ denotes the check function associated with the τth quantile. The parameter of interest becomes

$$\vartheta^{(v)}_{\mathbf{w}}(x) = \frac{\partial^v}{\partial x^v}Q_\tau(y_i|x_i = x, \mathbf{w}_i = \mathbf{w})$$

  where $Q_\tau(y_i|x_i = x, \mathbf{w}_i = \mathbf{w}) = \mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0$ denotes the conditional τth quantile regression function of y_i given $x_i = x, \mathbf{w}_i = \mathbf{w}$.

The different parameters above, as well as many others, are determined by the choice of loss function $\rho(\cdot;\cdot)$ and (inverse) link function $\eta(\cdot)$. In the above formulation, we assume that the models are correctly specified relative to the true data-generating process [that is, relative to the assumptions on the probability distribution of the data $(y_i, x_i, \mathbf{w}_i')$, $i = 1, 2, \ldots, n$], which is also assumed to be an independent and identically distributed sample. However, in many settings, the choices of $\rho(\cdot;\cdot)$ and $\eta(\cdot)$ are only working models, which may not lead to the underlying target parameter but rather only to an approximation thereof in a principled way. That is, under incorrect specification, the parameter $\vartheta^{(v)}_{\mathbf{w}}(x)$ can be interpreted only as the solution to the minimization in (2). Under dependent data, binscatter can still be applied but statistical guarantees for parameter estimation and for uncertainty quantification are not available.

2.1 Binscatter construction

To approximate $\mu_0(x)$ and its derivatives in (2), binscatter first partitions the support of x_i into J quantile-spaced bins, leading to the partitioning scheme

$$\widehat{\Delta} = \left\{\widehat{\mathcal{B}}_1, \ldots, \widehat{\mathcal{B}}_J\right\}, \qquad \widehat{\mathcal{B}}_j = \begin{cases} \left[x_{(1)}, x_{(\lfloor n/J \rfloor)}\right) & \text{if } j = 1, \\ \left[x_{(\lfloor n(j-1)/J \rfloor)}, x_{(\lfloor nj/J \rfloor)}\right) & \text{if } j = 2, \ldots, J-1 \\ \left[x_{(\lfloor n(J-1)/J \rfloor)}, x_{(n)}\right] & \text{if } j = J \end{cases}$$

where $x_{(i)}$ denotes the ith order statistic of the sample $\{x_1, x_2, \ldots, x_n\}$, $\lfloor \cdot \rfloor$ is the floor operator, and $J < n$. Each estimated bin $\widehat{\mathcal{B}}_j$ contains roughly the same number of observations $N_j = \sum_{i=1}^n \mathbb{1}_{\widehat{\mathcal{B}}_j}(x_i)$, where $\mathbb{1}_{\mathcal{A}}(x) = \mathbb{1}(x \in \mathcal{A})$ with $\mathbb{1}(\cdot)$ denoting the indicator function. This binning approach is the most popular in empirical work, but, for completeness, all commands in the package `binsreg` also allow for evenly spaced binning and user-specified binning. See below for more implementation details.

Given the quantile-spaced partitioning or binning scheme, for a choice of number of bins J and a choice of loss function $\rho(\cdot;\cdot)$ and (inverse) link function $\eta(\cdot)$, the generalized nonlinear binscatter estimator of the vth derivative $\vartheta_{\mathbf{w}}^{(v)}(x)$ of $\eta\{\mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0\}$ in (1), using a pth-order polynomial approximation within each bin, imposing $(s-1)$ times differentiability across bins, and adjusting for additional covariates $\mathbf{w}_i$, is

$$\widehat{\vartheta}_{\mathbf{w}}^{(v)}(x) = \frac{\partial^v}{\partial x^v}\eta\left\{\widehat{\mu}(x) + \mathbf{w}'\widehat{\boldsymbol{\gamma}}\right\} \tag{3}$$

where

$$\widehat{\mu}^{(v)}(x) = \widehat{\mathbf{b}}_{p,s}^{(v)}(x)'\widehat{\boldsymbol{\beta}}, \qquad \begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\boldsymbol{\gamma}} \end{bmatrix} = \underset{\boldsymbol{\beta},\boldsymbol{\gamma}}{\operatorname{argmin}} \sum_{i=1}^n \rho\left[y_i;\ \eta\left\{\widehat{\mathbf{b}}_{p,s}(x_i)'\boldsymbol{\beta} + \mathbf{w}_i'\boldsymbol{\gamma}\right\}\right] \tag{4}$$

with $s \le p$, $v \le p$, and $\widehat{\mathbf{b}}_{p,s}(x) = \widehat{\mathbf{T}}_s \widehat{\mathbf{b}}_{p,0}(x)$ with

$$\widehat{\mathbf{b}}_{p,0}(x) = \begin{bmatrix} \mathbb{1}_{\widehat{\mathcal{B}}_1}(x) & \mathbb{1}_{\widehat{\mathcal{B}}_2}(x) & \cdots & \mathbb{1}_{\widehat{\mathcal{B}}_J}(x) \end{bmatrix}' \otimes \begin{bmatrix} 1 & x & \cdots & x^p \end{bmatrix}'$$

being the pth-order polynomial basis of approximation within each bin, hence of dimension $(p+1)J$, and $\widehat{\mathbf{T}}_s$ being a $\{(p+1)J - (J-1)s\} \times (p+1)J$ matrix of linear restrictions ensuring that the $(s-1)$th derivative of $\widehat{\mu}(x)$ is continuous.

When $s = 0$, $\widehat{\mathbf{T}}_0 = \mathbf{I}_{(p+1)J}$, the identity matrix of dimension $(p+1)J$, and therefore no restrictions are imposed: $\widehat{\mathbf{b}}_{p,0}(x)$ is the basis used for (disjoint) piecewise pth-order polynomial fits. Consequently, the binscatter $\widehat{\mu}(x)$ is discontinuous at the bins' edges whenever $s = 0$. On the other hand, $p \ge s$ implies that a pth-order polynomial fit is constructed within each bin $\widehat{\mathcal{B}}_j$, in which case setting $s = 1$ forces these fits to be connected at the boundaries of adjacent bins (leading to a continuous but nondifferentiable function), $s = 2$ forces these fits to be connected and continuously differentiable at the boundaries of adjacent bins, and so on. Enforcing smoothness on binscatter boils down to incorporating restrictions on the basis of approximation. The resulting constrained basis, $\widehat{\mathbf{b}}_{p,s}(x)$, corresponds to a choice of spline basis for approximation of $\mu_0(\cdot)$ in (2), with estimated quantile-spaced knots according to the partition $\widehat{\Delta}$. The package `binsreg` uses $\widehat{\mathbf{T}}_s$ leading to B-splines, which tend to have very good finite sample properties.

The binscatter estimator $\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x)$ in (3) is a plugin estimator for (2). Returning to the settings of applied interest mentioned previously, we have the following:

- *Semilinear regression*: $\rho(y;\eta) = (y-\eta)^2$ and $\eta(u) = u$. This case corresponds to linear least-squares binscatter, where the estimator becomes

$$\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x) = \frac{\partial^v}{\partial x^v}\widehat{\mathbb{E}}\left(y_i|x_i = x, \mathbf{w}_i = \mathbf{w}\right) = \begin{cases} \widehat{\mu}(x) + \mathbf{w}'\widehat{\boldsymbol{\gamma}} & \text{if } v = 0 \\ \widehat{\mu}^{(v)}(x) & \text{if } v \geq 1 \end{cases}$$

  This estimator is obtained by running the linear least-squares regression of y_i on $(\widehat{\mathbf{b}}_{p,s}(x_i)', \mathbf{w}_i')$ and then constructing predicted values at $(x, \mathbf{w}')$ for $v = 0$ or predicted values $\widehat{\mu}^{(v)}(x) = \widehat{\mathbf{b}}^{(v)}_{p,s}(x)'\widehat{\boldsymbol{\beta}}$ for $v \geq 1$. The command `binsreg` implements this case.

- *Logistic or probit regression*: $\rho(y;\eta) = -y\log\eta - (1-y)\log(1-\eta)$ and $\eta(u)$ denotes the (inverse) link function of logistic or probit regression. This case corresponds to nonlinear logistic or probit binscatter, where the estimator becomes

$$\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x) = \frac{\partial^v}{\partial x^v}\widehat{\mathbb{E}}\left(y_i|x_i = x, \mathbf{w}_i = \mathbf{w}\right) = \frac{\partial^v}{\partial x^v}\eta\{\widehat{\mu}(x) + \mathbf{w}'\widehat{\boldsymbol{\gamma}}\}$$

  This estimator is obtained by running the logit or probit nonlinear regression of y_i on $(\widehat{\mathbf{b}}_{p,s}(x_i)', \mathbf{w}_i')$ and constructing predicted values at $(x, \mathbf{w}')$ for $v = 0$ or derivatives thereof for $v \geq 1$. The commands `binslogit` and `binsprobit` implement these cases. These two commands allow only $v = 0$ or 1.

- *Quantile regression*: $\rho(y;\eta) = \ell_\tau(y-\eta)$ and $\eta(u) = u$, where $\ell_\tau(u)$ denotes the check function associated with the τth quantile. This case corresponds to nonlinear, nondifferentiable quantile regression binscatter, where the estimator becomes

$$\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x) = \frac{\partial^v}{\partial x^v}\widehat{Q}_\tau(y_i|x_i = x, \mathbf{w}_i = \mathbf{w})$$

  where $\widehat{Q}_\tau(y_i|x_i = x, \mathbf{w}_i = \mathbf{w}) = \widehat{\mu}(x) + \mathbf{w}'\widehat{\gamma}$ denotes the estimate of the conditional τth quantile function of y_i given $(x_i, \mathbf{w}_i')$. This estimator is obtained by running the quantile regression of y_i on $(\widehat{\mathbf{b}}_{p,s}(x_i)', \mathbf{w}_i')$ and constructing predicted values at $(x, \mathbf{w}')$ for $v = 0$ or derivatives thereof for $v \geq 1$. The command `binsqreg` implements this case.

In practice, the binscatter estimator $\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x)$ needs to be evaluated at some point $\mathbf{w}$. Typical choices are $\mathbf{w} = \mathbf{0}$, $\mathbf{w} = \overline{\mathbf{w}} = (1/n)\sum_{i=1}^n \mathbf{w}_i$, or $\mathbf{w} = \text{median}(\mathbf{w}_i)$, with $\mathbf{0}$ denoting a vector of zeros and median($\mathbf{w}_i$) denoting the empirical median of each component in $\mathbf{w}_i$. For discrete variables in $\mathbf{w}$, it is natural to set those components to some base category (for example, zero for binary variables), while for continuous variables, it may be better to set those components at some other value (for example, mean or some quantile). The point of evaluation affects the visual and statistical properties of the binscatter, as discussed below.

2.1.1 Canonical binscatter

Canonical binscatter, as implemented in the packages `binscatter` and `binscatter2`, corresponds to linear least-squares regression [$\rho(y;\eta) = (y-\eta)^2$ and $\eta(u) = u$] with $p = s = 0$ and without covariate adjustment [that is, not including $\mathbf{w}_i$ in (4)]. Specifically, in canonical binscatter, the basis $\widehat{\mathbf{b}}_{0,0}(x)$ is a J-dimensional vector of orthogonal dummy variables; that is, the jth component of $\widehat{\mathbf{b}}_{0,0}(x)$ records whether the evaluation point x belongs to the jth bin in the partition $\widehat{\Delta}$. Therefore, canonical binscatter can be expressed as the collection of J sample averages of the response variable y_i, one for each bin: $\overline{y}_j = (1/N_j)\sum_{i=1}^{n} \mathbb{1}_{\widehat{\mathcal{B}}_j}(x_i)y_i$ for $j = 1, 2, \ldots, J$. Empirical work using canonical binscatter typically plots these binned sample averages along with some other estimate or estimates of the regression function $\mu_0(x)$.

2.1.2 Covariate-adjusted binscatter

Previous work using binscatter methods, including `binscatter` and `binscatter2`, not only considered exclusively least-squares regressions with $p = s = 0$ but also performed covariate adjustment by residualization. To be precise, first the residuals from the linear regressions of y_i on $(1, \mathbf{w}_i')$ and of x_i on $(1, \mathbf{w}_i')$ were computed, and then a canonical binscatter was estimated using those residuals. We call this approach residualized canonical binscatter.

CCFF showed that residualized canonical binscatter is very hard to rationalize or justify and will lead to an inconsistent estimator of $\vartheta^{(v)}_{\mathbf{w}}(x)$ unless very special assumptions hold, even when the statistical model is correctly specified. In contrast, our proposed approach for covariate adjustment (4) is justified via (2) and is therefore principled and interpretable. Even when (2) is misspecified, the approach to covariate adjustment used by the package `binsreg` enjoys a natural probability limit interpretation, while the residualization approach does not. See CCFF for more discussion, numerical examples, and technical details.

2.1.3 Main implementation details

The four estimation commands `binsreg`, `binslogit`, `binsprobit`, and `binsqreg` implement, respectively, least-squares, logit, probit, and quantile regression binscatter estimators for a given choice of partitioning or binning $\widehat{\Delta}$. The option `deriv()` is used to set the value of v, and the option `at()` is used to set the value of $\mathbf{w}$ in the estimator

$\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x)$. The options `dots(p s)` and `line(p s)` generate "dots" and a "line" tracing out two distinct implementations of $\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x)$ with the corresponding choices of p and s selected in each case but using the same values of v and $\mathbf{w}$. If `dots(T)` (or `line(T)`) is specified, `dots(0 0)` (or `line(0 0)`) is used unless the degree p or smoothness s selection is requested via the option `pselect()` or `sselect()` (see details in the next subsection).

The defaults are to estimate the level of the function ($v = 0$, `deriv(0)`) and evaluate the covariates at the mean ($\mathbf{w}_i = \overline{\mathbf{w}}$, `at(mean)`). Evaluating the covariates at different points can affect a level shift of the plotted point estimates and change the statistical uncertainty, such as confidence bands (see the supplemental appendixes of CCFF). The option `at()` allows for the mean, the median, and a vector of zeros, the last being useful for dummy variables or fixed effects. For high-dimensional fixed effects, the community-contributed `reghdfe` command can be called with the `absorb` option. In this case, the fixed effects are normalized within `reghdfe`, and the point of evaluation cannot be set.

For example, when using `binsreg`, the default implementation yields the estimate $\widehat{\vartheta}_{\overline{\mathbf{w}}}(x) = \widehat{\mathbb{E}}(y_i|x_i = x, \mathbf{w}_i = \overline{\mathbf{w}}) = \widehat{\mu}(x) + \overline{\mathbf{w}}'\widehat{\boldsymbol{\gamma}}$. Thus, `dots(0 0)` leads to "dots" representing sample averages within each bin for the "long" regression with $\mathbf{w}_i = \overline{\mathbf{w}}$. In particular, if $\mathbf{w}_i$ are not included, then the default coincides with canonical binscatter (that is, the same results would be obtained using the packages `binscatter` and `binscatter2` for the same J). The line option is muted by default and needs to be set explicitly to appear in the resulting plot: for example, the option `line(3 3)` adds a line tracing out $\widehat{\mu}^{(v)}(x)$, implemented with $p = 3$ and $s = 3$, a cubic B-spline approximation of $\mu_0^{(v)}(x)$.

The common partitioning or binning used by the four estimation commands across all implementations is set to be quantile spaced for some choice of J. The option `nbins()` sets J manually (for example, `nbins(20)` corresponds to $J = 20$ quantile-spaced bins), but if this option is not supplied, then the companion command `binsregselect` is used to choose J in a fully data-driven way, as described below. As an alternative, an evenly spaced or user-specified partitioning or binning can be implemented via the option `binspos()`.

Several other options are available for the four estimation commands, including multiway fixed effects and multiway clustering adjustments. Each command has an accompanying help file with complete details.

2.2 Choosing the number of bins

From a statistical point of view, J is the main tuning parameter of a binscatter, and as usual for nonparametrics, one must assume $J \to \infty$ for consistent estimation. To provide an optimal, data-driven choice of J, CCFF developed valid integrated mean squared error (IMSE) approximations for generalized nonlinear binscatter in the context of (2). These expansions give IMSE-optimal selection of the number of bins J, depending on polynomial order p within bins and smoothness level s across bins, the target estimand set by the derivative order v, and the evaluation point of interest $\mathbf{w}$ for covariate adjustment. Specifically, the IMSE-optimal choice of J is

$$J_{\text{IMSE}} = \left\lceil \left(\frac{2(p-v+1)\mathscr{B}_n(p,s,v)}{(1+2v)\mathscr{V}_n(p,s,v)} \right)^{\frac{1}{2p+3}} n^{\frac{1}{2p+3}} \right\rceil$$

where $\lceil \cdot \rceil$ denotes the ceiling operator, $\mathscr{B}_n(p,s,v)$ and $\mathscr{V}_n(p,s,v)$ represent an approximation to the integrated (squared) bias and variance of $\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x)$, respectively, and the three integer choices must respect $p \geq s \geq 0$ and $p \geq v \geq 0$. The constants $\mathscr{B}_n(p,s,v)$ and $\mathscr{V}_n(p,s,v)$ depend on the partitioning scheme and binscatter estimator used. Note that the package `binsreg` allows for the standard option `vce()` to specify variance–covariance estimation methods, which can also affect the variance constant $\mathscr{V}_n(p,s,v)$.

For simplicity, the command `binsregselect` in the package `binsreg` implements number-of-bins selection based on the IMSE expansion for the linear least-squares binscatter. For generalized nonlinear binscatter (logistic, probit, or quantile regression), the number of bins J given by the command `binsregselect` still has the "correct" rate (the same order as that of the IMSE-optimal one). Thus, confidence bands and testing procedures based on such choices of J and the robust bias-correction strategy described below are still valid even in the general nonlinear case.

Both IMSE constants, $\mathscr{B}_n(p,s,v)$ and $\mathscr{V}_n(p,s,v)$, can be estimated consistently using a preliminary choice of J. Thus, our implementation offers two J selectors.

- $\widehat{J}_{\text{ROT}}$: applies a rule-of-thumb (ROT) approximation for $\mathscr{B}_n(p,s,v)$ and $\mathscr{V}_n(p,s,v)$, using a trimmed-from-below Gaussian reference model for the density of x_i and global polynomial approximations for the other two unknown features needed, $\mu_0^{(v)}(x)$ and $\mathbb{V}[y_i|x_i = x, \mathbf{w}_i = \mathbf{w}]$. This J selector uses the correct rate but an inconsistent constant approximation.
- $\widehat{J}_{\text{DPI}}$: applies a direct-plugin (DPI) approximation for $\mathscr{B}_n(p,s,v)$ and $\mathscr{V}_n(p,s,v)$ based on the desired binscatter, set by the choices p and s, and using a preliminary J. If a preliminary J is not provided by the user, then $J = \max\{\widehat{J}_{\text{ROT}}, \lceil ([\{2(p-v+1)\}/(1+2v)]n)^{1/(2p+3)}\rceil\}$ is used for DPI implementation. This J selector uses the correct rate and consistent estimators of the appropriate constants. The default implementation uses $\widehat{J}_{\text{DPI}}$ whenever J is not specified.

Implementing a binscatter with $J = J_{\text{IMSE}}$ is optimal from a statistical point of view (for valid estimation, testing, and uncertainty quantification), but sometimes a fixed, user-chosen number of bins, denoted by $J = \mathtt{J}$, may yield a more visually appealing binscatter (albeit with the caveat that the implied estimator may be inaccurate). Many applications in the past used round bin numbers such as 10, 20, 50, or 100. Furthermore, a fixed $J = \mathtt{J}$ can also be directly interpretable as a discretized version of x. For instance, setting $J = \mathtt{J} = 10$ yields a binscatter that allows for comparison across deciles of x, for example, comparing those in the top decile of earnings with those at the bottom. Setting $\mathtt{J} = 100$ can be used to compare across percentiles and is commonly used when studying ranks.

To balance the potential for an appealing visualization based on a fixed $J = \mathtt{J}$ with the desire for statistical validity, CCFF developed a novel method for selecting the polynomial order p (and along with it, the smoothness s) as a function of the fixed $\mathtt{J}$, as opposed to $J_{\text{IMSE}} = J_{\text{IMSE}}(p, s, v)$, which does the reverse. Specifically, CCFF proposed looking for the values p^* and s^* (in a prespecified range) such that $J_{\text{IMSE}}(p^*, s^*, v)$ is approximately equal to the researcher's chosen $\mathtt{J}$. That is, finding the p and s for which the chosen $\mathtt{J}$ would be IMSE optimal. Although $p \to \infty$ is not allowed in the theory of CCFF, making this choice somewhat ad hoc, it may effectively reduce the bias of the binscatter.

2.2.1 Main implementation details

Unless J is manually specified, all commands in the `binsreg` package use the command `binsregselect` to implement ROT and DPI data-driven, IMSE-optimal selection of J for all possible choices of $p \geq v$, $s \geq 0$, and for both quantile-spaced or evenly spaced partitioning or binning. For DPI implementation, the user can provide the initialization value of J via the option `nbinsrot()`; if this value is not provided, then $\widehat{J}_{\text{ROT}}$ is used.

As discussed above, instead of selecting the number of bins J, an alternative strategy is setting a fixed value for J and implementing (ROT or DPI) data-driven, IMSE-optimal selection of the degree of polynomial p or the number of smoothness constraints s. The command `binsregselect` implements this selection procedure if 1) the number of bins J is supplied via the option `nbins()` and 2) a range for searching for the optimal p or s is supplied via the option `pselect()` or `sselect()`.

Several other options are available for the command `binsregselect`, including the possibility of generating an output file with the IMSE-optimal partitioning or binning structure selected and the corresponding grid of evaluation points, which can be used by the other six companion commands for plotting, simulation, testing, and other calculations.

2.3 Confidence intervals

Both confidence intervals and confidence bands for the unknown function $\vartheta^{(v)}_{\mathbf{w}}(x)$ are constructed using the same type of Studentized t statistic:

$$T_p(x) = \frac{\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x) - \vartheta^{(v)}_{\mathbf{w}}(x)}{\sqrt{\widehat{\Omega}(x)/n}} \qquad 0 \leq v, s \leq p$$

The binscatter variance estimator is of the usual "sandwich" form

$$\widehat{\Omega}(x) = \widehat{\mathbf{b}}^{(v)}_{p,s}(x)'\widehat{\mathbf{Q}}^{-1}\widehat{\mathbf{\Sigma}}\widehat{\mathbf{Q}}^{-1}\widehat{\mathbf{b}}^{(v)}_{p,s}(x)\left\{\eta^{(1)}(\widehat{\mu}(x)+\mathbf{w}'\widehat{\gamma})\right\}^2$$
$$\widehat{\mathbf{Q}} = \frac{1}{n}\sum_{i=1}^{n}\widehat{\mathbf{b}}_{p,s}(x_i)\widehat{\mathbf{b}}_{p,s}(x_i)'\widehat{\Psi}_{i,1}\widehat{\eta}^2_{i,1}$$
$$\widehat{\mathbf{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}\widehat{\mathbf{b}}_{p,s}(x_i)\widehat{\mathbf{b}}_{p,s}(x_i)'\widehat{\eta}^2_{i,1}\psi\left(y_i,\widehat{\eta}_{i,0}\right)^2$$

with $\widehat{\Psi}_{i,1}$ being a consistent estimator of $\Psi_{i,1} = \frac{\partial}{\partial\eta}\mathbb{E}\{\psi(y_i,\eta)|x_i,\mathbf{w}_i\}\big|_{\eta=\eta_{i,0}}$, $\widehat{\eta}_{i,v}$ being a consistent estimator of $\eta_{i,v} = \eta^{(v)}\{\mu_0(x_i)+\mathbf{w}_i'\gamma_0\}$ for $v = 0, 1$, and $\psi(y,u)$ being the (weak) derivative of $\rho(\cdot;u)$ with respect to u. See CCFF for details and omitted formulas. In practice, these estimators are implemented using the base commands from the statistical software.

CCFF showed that $T_p(x) \to_d \mathsf{N}(0,1)$ pointwise in x, that is, for each evaluation point x on the support of x_i, provided the misspecification error introduced by binscatter is removed from the distributional approximation. Such a result justifies asymptotically valid confidence intervals for $\vartheta^{(v)}_{\mathbf{w}}(x)$, pointwise in x after bias correction. Specifically, for each x, the $(1-\alpha)\%$ confidence interval takes the form

$$\widehat{I}_p(x) = \left\{\ \widehat{\vartheta}^{(v)}_{\mathbf{w}}(x) \pm \Phi^{-1}(1-\alpha/2)\times\sqrt{\widehat{\Omega}(x)/n}\ \right\} \qquad 0 \le v, s \le p$$

where $\Phi(u)$ denotes the distribution function of a standard normal random variable (for example, $\Phi^{-1}(1-0.05/2) \approx 1.96$ for a 95% Gaussian confidence intervals), provided the choice of J is such that the misspecification error can be ignored.

However, using an IMSE-optimal binscatter (that is, setting $J = J_{\mathrm{IMSE}}$ for the selected polynomial order p) introduces a first-order misspecification error leading to invalidity of these confidence intervals and hence cannot be directly used to form the confidence intervals $\widehat{I}_p(x)$ in general. To address this problem, we rely on a simple application of robust bias correction (Calonico, Cattaneo, and Titiunik 2014; Calonico, Cattaneo, and Farrell 2018, 2022; Cattaneo, Farrell, and Feng 2020) to form valid confidence intervals based on IMSE-optimal binscatter, that is, without altering the partitioning scheme $\widehat{\Delta}$ used.

Our implementation uses robust bias-corrected binscatter confidence intervals as follows. First, for a given choice of p, select the number of bins in $\widehat{\Delta}$ according to $J = J_{\mathrm{IMSE}}$, which gives an IMSE-optimal binscatter (point estimator). Then, use the confidence interval $\widehat{I}_{p+q}(x)$ with $q \ge 1$, which gives a valid confidence interval: $\mathbb{P}\left[\vartheta^{(v)}_{\mathbf{w}}(x) \in \widehat{I}_{p+q}(x)\right] \to 1-\alpha$ for all x.

The same strategy is applied when the degree or smoothness selection described previously is implemented. First, for a fixed choice of J, select the "optimal" degree of polynomial p (that is, the resulting J_{IMSE} is the closest to the chosen J). Then, increase the degree of polynomial to construct the confidence intervals $\widehat{I}_{p+q}(x)$. This idea is also used in the construction of confidence bands and hypothesis testing procedures and can be fully implemented using the options `pselect()` and `sselect()`. Because the degree or smoothness selection is *not* the default in the package `binsreg`, our discussion focuses on robust bias correction based on IMSE-optimal J selection.

2.3.1 Main implementation details

The four estimation commands `binsreg`, `binslogit`, `binsprobit`, and `binsqreg` implement confidence intervals and report them as part of the final binned scatterplot. Specifically, the option `ci(p s)` estimates confidence intervals with the corresponding choices of p and s selected and plots them as vertical segments along the support of x_i. If `ci(T)` is specified, `ci(1 1)` is used unless the degree p or smoothness s selection is requested via the option `pselect()` or `sselect()` as described before. The confidence interval option is muted by default and needs to be set explicitly to appear in the resulting plot. The implementation is done over a grid of evaluation points, which can be modified via the option `cigrid()`, and the desired level is set by the option `level()`. Note that `dots(p s)`, `lines(p s)`, and `ci(p s)` may all take different choices of p and s, which allows for robust bias-correction implementation of the confidence intervals and permits incorporating different levels of smoothness restrictions.

2.4 Confidence bands

In many empirical applications of binscatter, the goal is to conduct inference about the entire function $\vartheta^{(v)}_{\mathbf{w}}(x)$ simultaneously, that is, uniformly over all x values of the support of x_i. This goal is fundamentally different from pointwise inference. A leading example of uniform inference is reporting confidence bands for $\vartheta_{\mathbf{w}}(x)$ and its derivatives, which are different from (pointwise) confidence intervals. The package `binsreg` offers asymptotically valid constructions of both confidence intervals, as discussed above, and confidence bands, which can be implemented with the same choices of (p, s) used to construct $\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x)$ or different ones.

Following the theoretical work in CCFF, for a choice of p and partition or binning of size J, the $(1-\alpha)\%$ confidence band for $\vartheta^{(v)}_{\mathbf{w}}(x)$ is

$$\widehat{I}_p(\cdot) = \left\{ \widehat{\vartheta}^{(v)}_{\mathbf{w}}(\cdot) \pm \mathfrak{c} \times \sqrt{\widehat{\Omega}(\cdot)/n} \right\} \qquad 0 \leq v, s \leq p$$

where the quantile value $\mathfrak{c}$ is now approximated via simulations using

$$\mathfrak{c} = \inf\left[c \in \mathbb{R}_+ : \mathbb{P}\left\{ \sup_x \left| \widehat{Z}_p(x) \right| \leq c \,\middle|\, \mathbf{D} \right\} \geq 1-\alpha \right]$$

with $\mathbf{D} = \{(y_i, x_i, \mathbf{w}_i') : 1 \leq i \leq n\}$ denoting the original data,

$$\widehat{Z}_p(x) = \frac{\widehat{\mathbf{b}}^{(v)}_{p,s}(x)'\widehat{\mathbf{Q}}^{-1}\widehat{\mathbf{\Sigma}}^{1/2}}{\sqrt{\widehat{\Omega}(x)/n}} \mathbf{N}^\star_K \qquad K = (p+1)J - (J-1)s,\ 0 \leq v, s \leq p$$

and $\mathbf{N}_K^\star \sim \mathsf{N}(\mathbf{0}, \mathbf{I})$ being a K-dimensional standard normal random vector independent of the data $\mathbf{D}$. The distribution of $\sup_x |T_p(x)|$, which is unknown, is approximated by that of $\sup_x |\widehat{Z}_p(x)|$ conditional on the data $\mathbf{D}$, which can be simulated by taking repeated samples from $\mathbf{N}_K^\star$ and recomputing the supremum each time. In other words, the quantiles used to construct confidence bands can be approximated by resampling from the standard normal random vector $\mathbf{N}_K^\star$, keeping fixed the data $\mathbf{D}$ (and hence all quantities depending on it). See CCFF for more details.

A confidence band covers the entire function, $\vartheta_{\mathbf{w}}^{(v)}(x)$, $(1-\alpha)$% of the time in repeated sampling whenever the misspecification error can be ignored. As before, we recommend using robust bias correction to remove misspecification errors introduced by binscatter, that is, following the same logic discussed above for the case of confidence interval construction. To be more precise, first, p is chosen, along with s and v, and the optimal partitioning or binning is selected according to $J = J_{\text{IMSE}}$. Then, valid confidence bands are constructed using $\widehat{I}_{p+q}(x)$ with $q \geq 1$: $\mathbb{P}\big[\vartheta_{\mathbf{w}}^{(v)}(x) \in \widehat{I}_{p+q}(x), \text{ for all } x\big] \to 1-\alpha$.

Moreover, the visual appearance of the confidence band will be impacted by the chosen evaluation point $\mathbf{w}$, and the researcher needs to be careful when using the band as visual aids in parametric specification testing. See CCFF, particularly section SA-1.2 of Cattaneo et al. (2024b).

2.4.1 Main implementation details

The four estimation commands `binsreg`, `binslogit`, `binsprobit`, and `binsqreg` implement confidence bands and report them as part of the final binned scatterplot. The option `cb(p s)` estimates an asymptotically valid confidence band with the corresponding choices of p and s selected and plots it as a shaded region along the support of x_i. If `cb(T)` is specified, `cb(1 1)` is used unless the degree p or smoothness s selection is requested via the option `pselect()` or `sselect()` as described before. The confidence band option is muted by default and needs to be set explicitly to appear in the resulting plot. The implementation is done over a grid of evaluation points, which can be modified via the option `cbgrid()`, and the desired level is set by the option `level()`. The options `dots(p s)`, `lines(p s)`, `ci(p s)`, and `cb(p s)` can all take different choices of p and s, which allows for robust bias-correction implementations and many other practically relevant possibilities.

2.5 Parametric specification testing

In addition to implementing binscatter and producing binned scatterplots, with both point estimation and uncertainty visualization, the package `binsreg` allows for formal testing of substantive hypotheses. The command `binstest` implements two types of substantive hypothesis tests about $\vartheta_{\mathbf{w}}^{(v)}(x)$: 1) parametric specification testing and 2) nonparametric shape restriction testing. This subsection discusses the former, while the next subsection discusses the latter.

For a choice of (p, s, v) and partitioning or binning scheme of size J, the implemented parametric specification testing approach contrasts a (nonparametric) binscatter approximation $\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x)$ of $\vartheta^{(v)}_{\mathbf{w}}(x)$ with a hypothesized parametric specification of the form $\vartheta_{\mathbf{w}}(x) = \eta\{M_{\mathbf{w}}(x; \boldsymbol{\theta}, \boldsymbol{\gamma}_0)\}$, where $M_{\mathbf{w}}(x; \boldsymbol{\theta}, \boldsymbol{\gamma}_0) = m(x; \boldsymbol{\theta}) + \mathbf{w}'\boldsymbol{\gamma}_0$ for some $m(\cdot)$ known up to a finite parameter $\boldsymbol{\theta}$, which can be estimated using the available data. Formally, the null and alternative hypotheses are, respectively,

$$\dot{\mathsf{H}}_0: \quad \sup_x \left| \vartheta^{(v)}_{\mathbf{w}}(x) - \frac{\partial^v}{\partial x^v} \eta \left\{ M_{\mathbf{w}}(x; \boldsymbol{\theta}, \boldsymbol{\gamma}_0) \right\} \right| = 0 \quad \text{for some } \boldsymbol{\theta} \qquad \text{versus}$$

$$\dot{\mathsf{H}}_{\mathrm{A}}: \quad \sup_x \left| \vartheta^{(v)}_{\mathbf{w}}(x) - \frac{\partial^v}{\partial x^v} \eta \left\{ M_{\mathbf{w}}(x; \boldsymbol{\theta}, \boldsymbol{\gamma}_0) \right\} \right| > 0 \quad \text{for all } \boldsymbol{\theta}$$

for a choice of v.

For example, excluding covariates $\mathbf{w}_i$, $\widehat{\mu}(x)$ is compared with $\overline{y} = (1/n)\sum_{i=1}^n y_i$ to assess whether there is a relationship between y_i and x_i or, more formally, whether $\mu_0(x)$ is a constant function. Similarly, it is possible to formally test for a linear, quadratic, or even nonlinear parametric relationship $\mu_0(x) = m(x, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ would be estimated from the data under the null hypothesis, that is, assuming that the postulated relationship is indeed correct.

However, when additional covariates $\mathbf{w}_i$ are included, CCFF showed that the special case of the test $\dot{\mathsf{H}}_0$ versus $\dot{\mathsf{H}}_{\mathrm{A}}$ with $v = 0$ could give conclusions that are sensitive to the user-selected point of evaluation $\mathbf{w}$, which implies that the common practice of visually examining a binned scatterplot compared with a specific parametric could be misleading. See section SA-1.2 of Cattaneo et al. (2024b) for further details.

To avoid this issue, and motivated by the fact that the central point of binscatter is to study how y_i relates to x_i, controlling for $\mathbf{w}_i$, we advocate reformulating the hypothesis as pertaining to the *derivative* of $\mu_0(x)$ instead of the level. For example, to test whether $\mu_0(x)$ is linear, one can test whether it has a constant first derivative, that is, $\mathsf{H}_0 : \sup_x |\mu_0^{(1)}(x) - a| = 0$ for some a, versus $\mathsf{H}_{\mathrm{A}} : \sup_x |\mu_0^{(1)}(x) - a| > 0$. This can be viewed as a special case of the test $\dot{\mathsf{H}}_0$ versus $\dot{\mathsf{H}}_{\mathrm{A}}$ above with $v = 1$ and the (inverse) link $\eta(\cdot)$ suppressed [that is, apply the test to the linear index $\mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0$ directly].

Formally, the command `binstest` uses the test statistic

$$\dot{T}_p(x) = \frac{\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x) - \frac{\partial^v}{\partial x^v} \eta \left\{ M_{\mathbf{w}} \left(x; \widetilde{\boldsymbol{\theta}}, \widetilde{\boldsymbol{\gamma}} \right) \right\}}{\sqrt{\widehat{\Omega}(x)/n}} \qquad 0 \le v, s \le p$$

where $(\widetilde{\boldsymbol{\theta}}', \widetilde{\boldsymbol{\gamma}}')'$ are consistent estimates of $(\boldsymbol{\theta}', \boldsymbol{\gamma}_0')'$ under the null hypothesis (correct parametric specification) and are "well behaved" under the alternative hypothesis (parametric misspecification). The researcher needs to carefully choose a proper v and decide whether the test should be applied to $\eta(\mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0)$ or $\mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0$, following the discussion above. Then a parametric specification hypothesis testing procedure is

$$\text{Reject } \dot{\mathsf{H}}_0 \qquad \text{if and only if} \qquad \sup_x |\dot{T}_p(x)| \ge \mathfrak{c} \tag{5}$$

where $\mathfrak{c} = \inf\{c \in \mathbb{R}_+ : \mathbb{P}[\sup_x |\widehat{Z}_p(x)| \leq c \mid \mathbf{D}] \geq 1-\alpha\}$ is again computed by simulation from a standard normal random vector, conditional on the data $\mathbf{D}$, as in the case of confidence bands already discussed. This testing procedure is an asymptotically valid $\alpha\%$-level test if the misspecification error is removed from the test statistic $\dot{T}_p(x)$.

The command `binstest` uses robust bias correction by default: First, p and s are chosen, and the partitioning or binning scheme is selected by setting $J = J_{\mathrm{IMSE}}$ for these choices. Then, using this partitioning scheme, (5) is implemented with the choice $p+q$ instead of p, with $q \geq 1$. CCFF showed that, under regularity conditions, the resulting parametric specification testing approach controls for type I errors with nontrivial power: for given p, $0 \leq v$, $s \leq p$, and $J = J_{\mathrm{IMSE}}$,

$$\lim_{n\to\infty} \mathbb{P}\Big[\sup_x \big|\dot{T}_{p+q}(x)\big| > \mathfrak{c}\Big] = \alpha \qquad \text{under } \dot{\mathsf{H}}_0$$

and

$$\lim_{n\to\infty} \mathbb{P}\Big[\sup_x \big|\dot{T}_{p+q}(x)\big| > \mathfrak{c}\Big] = 1 \qquad \text{under } \dot{\mathsf{H}}_{\mathrm{A}}$$

where $q \geq 1$. This testing approach formalizes the intuitive idea that if the confidence band for $\vartheta^{(v)}_{\mathbf{w}}(x)$ does not contain the hypothesized parametric fit entirely, then the parametric fit is incompatible with the data; that is, the null should be rejected.

2.5.1 Main implementation details

The command `binstest` implements parametric specification testing in two ways. First, polynomial regression (parametric) specification testing is implemented directly via the option `testmodelpoly(P)`, where the null hypothesis is $m(x,\boldsymbol{\theta}) = \theta_0 + x\theta_1 + \cdots + x^P\theta_P$ and $\boldsymbol{\theta} = (\theta_0, \theta_1, \ldots, \theta_P)'$ is estimated by least-squares regression. For other parameterizations of $m(x,\boldsymbol{\theta})$, the command takes as input an auxiliary array or database (`.dta` in Stata or data frame in Python and R) via the option `testmodelparfit()` containing the following columns or variables: grid of evaluation points in one column and fitted values $\eta^{(v)}(m(x,\widetilde{\boldsymbol{\theta}}) + \mathbf{w}'\widetilde{\gamma})$ (over the evaluation grid) for each parametric model considered in other columns or variables. The ordering of these variables is arbitrary, but they have to follow a naming rule: the evaluation grid has the same name as the independent variable x_i, and the names of other variables storing fitted values take the form `binsreg_fit*`.

The binscatter (nonparametric) estimate used to construct the testing procedure is set by the options `testmodel(p s)` and `deriv(v)`, and the partitioning or binning scheme selected. If `testmodel(T)` or `testmodel()` is supplied, then the default `testmodel(1 1)` is used unless the degree p and smoothness s selection is requested via the options `pselect()` and `sselect()` as described before. The option `nolink` can be used to specify whether the test should be applied to the linear index directly.

2.6 Nonparametric shape testing

The second type of hypothesis tests implemented by the command `binstest` concerns nonparametric testing of shape restrictions. For a choice of v, the null and alternative hypotheses of these testing problems are

$$\ddot{\mathsf{H}}_0: \quad \sup_x \vartheta^{(v)}_{\mathtt{w}}(x) \leq 0 \qquad \text{versus} \qquad \ddot{\mathsf{H}}_{\mathrm{A}}: \quad \sup_x \vartheta^{(v)}_{\mathtt{w}}(x) > 0$$

that is, a one-sided testing problem to the left. For example, negativity, monotonicity, and concavity of $\vartheta_{\mathtt{w}}(x)$ correspond to $\vartheta_{\mathtt{w}}(x) \leq 0$, $\vartheta^{(1)}_{\mathtt{w}}(x) \leq 0$, and $\vartheta^{(2)}_{\mathtt{w}}(x) \leq 0$, respectively. Of course, the analogous testing problem to the right is also implemented but not discussed here to avoid unnecessary repetition.

The relevant Studentized test statistic for this class of testing problems is

$$\ddot{T}_p(x) = \frac{\widehat{\vartheta}^{(v)}_{\mathtt{w}}(x)}{\sqrt{\widehat{\Omega}(x)/n}} \qquad 0 \leq v, s \leq p$$

Then the testing procedure is

$$\text{Reject } \ddot{\mathsf{H}}_0 \qquad \text{if and only if} \qquad \sup_x \ddot{T}_p(x) \geq \mathfrak{c} \tag{6}$$

with $\mathfrak{c} = \inf\{c \in \mathbb{R}_+ : \mathbb{P}[\sup_x \widehat{Z}_p(x) \leq c \mid \mathbf{D}] \geq 1-\alpha\}$. Misspecification errors of binscatter need to be accounted for to control for type I errors. CCFF showed that for given p, $0 \leq v$, $s \leq p$, and $J = J_{\mathrm{IMSE}}$ accordingly, then

$$\lim_{n\to\infty} \mathbb{P}\Big[\sup_x \ddot{T}_{p+q}(x) > \mathfrak{c}\Big] \leq \alpha \qquad \text{under } \ddot{\mathsf{H}}_0$$

and

$$\lim_{n\to\infty} \mathbb{P}\Big[\sup_x \ddot{T}_{p+q}(x) > \mathfrak{c}\Big] = 1 \qquad \text{under } \ddot{\mathsf{H}}_{\mathrm{A}}$$

for any $q \geq 1$, that is, using a robust bias-correction approach. These results imply that (6) is an asymptotically valid hypothesis test, provided that it is implemented with the choice $q \geq 1$ after the IMSE-optimal partitioning or binning scheme for binscatter of order p is selected.

2.6.1 Main implementation details

The command `binstest` implements one-sided and two-sided nonparametric shape restriction testing as follows. Option `testshapel(a)` implements one-sided testing to the left: $\ddot{\mathsf{H}}_0 : \sup_x \vartheta^{(v)}_{\mathtt{w}}(x) \leq \mathtt{a}$. Option `testshaper(a)` does the same to the right: $\ddot{\mathsf{H}}_0 : \inf_x \vartheta^{(v)}_{\mathtt{w}}(x) \geq \mathtt{a}$. Option `testshape2(a)` implements two-sided testing: $\ddot{\mathsf{H}}_0 : \sup_x |\vartheta^{(v)}_{\mathtt{w}}(x) - \mathtt{a}| = 0$. The constant `a` needs to be specified by the user.

The binscatter (nonparametric) estimate used to construct the testing procedure is set by the options `testshape(p s)` and `deriv(v)` and the chosen partitioning or binning scheme. If `testshape(T)` or `testshape()` is supplied, `testshape(1 1)` is used unless the degree p and smoothness s selection is requested via the options `pselect()` and `sselect()` as described before.

2.7 Multisample estimation and testing

The package `binsreg` also allows for comparisons of mean, quantile, and other regression functions across different groups (or treatment arms), which can be useful for estimation and inference of treatment effects that are heterogeneous in x_i, possibly after controlling for $\mathbf{w}_i$. For each subsample defined by a group indicator variable, the parameter of interest can be defined as $\vartheta^{(v)}_{\mathbf{w},\ell}(x)$, which corresponds to the parameter in (1) for specific subsample $\ell = 0, 1, 2, \dots, L$.

For example, assuming that two subsamples of the same size n are available ($L = 1$), one being a control group and the other a treatment group, all the methods discussed above can be applied to each subsample. Furthermore, the null hypothesis of no heterogeneous treatment effect is $\mathsf{H}_0^{\Delta} : \vartheta^{(v)}_{\mathbf{w},0}(x) = \vartheta^{(v)}_{\mathbf{w},1}(x)$ for all $x \in \mathcal{X}$, which captures the idea of no (heterogeneous in x_i) treatment effect across the two groups. A natural test statistic is

$$T_p^{\Delta}(x) = \frac{\widehat{\vartheta}^{(v)}_{\mathbf{w},1}(x) - \widehat{\vartheta}^{(v)}_{\mathbf{w},0}(x)}{\sqrt{\widehat{\Omega}_1(x)/n + \widehat{\Omega}_0(x)/n}} \qquad 0 \le v, s \le p$$

which compares the pairwise difference between the two groups, where $\widehat{\Omega}_\ell(x)$ is the variance estimator [of $\widehat{\vartheta}^{(v)}_{\mathbf{w},\ell}(x)$] for the subsample $\ell = 0, 1$. The testing procedure is

$$\text{Reject } \mathsf{H}_0^{\Delta} \qquad \text{if and only if} \qquad \sup_x \left| T_p^{\Delta}(x) \right| \ge \mathfrak{c}$$

with the critical value obtained as before via Gaussian approximations (resampling from a normal random vector conditional on the data). As discussed before, in practice the robust bias-corrected test statistic $T^{\Delta}_{p+q}(x)$ is used to eliminate misspecification bias and obtain a valid hypothesis testing procedure.

All the ideas and results above also apply to pairwise comparisons across multisamples. In particular, estimation, uncertainty quantification, and hypothesis testing can be conducted for each subsample at the time, and then hypothesis testing for pairwise comparisons can also be implemented following the results above. CCFF provided all the necessary theoretical background. Importantly, concerns regarding the choice of evaluation point $\mathbf{w}$ also apply to the multisample testing problems: researchers need to be careful when implementing the tests and interpreting the results.

2.7.1 Main implementation details

Estimation and uncertainty quantification across subsamples is done using the estimation commands (`binsreg`, `binslogit`, `binsprobit`, and `binsqreg`) via the option `by()`.

In addition, the command `binspwc` implements formal hypothesis testing for pairwise comparisons for the null hypothesis $\mathsf{H}_0^{\triangle}$ (and analogous one-sided problems).

2.8 Extensions and other implementation details

The package `binsreg` is implemented using the base commands in the statistical software. For example, in Stata, `binsreg` relies on `regress` (or `reghdfe` if that option is selected); `binslogit` relies on `logit`; `binsprobit` relies on `probit`; and `binsqreg` relies on `qreg` (or `bsqreg` if the bootstrapping-based standard error is selected). Furthermore, the testing commands (`binstest` and `binspwc`) also use base commands whenever possible. This approach may sacrifice some speed of implementation but improves substantially in terms of stability and replicability. Importantly, most options available in the base commands are available in the package `binsreg`.

This section reviews some specific extensions and other numerical issues of the package `binsreg` and discusses related choices made for implementation, all of which can affect speed or robustness of the package.

2.8.1 Other metrics

All the results presented above use the uniform norm; that is, they focus on the largest deviation on the support of a function. See, for example, $\dot{\mathsf{H}}_0$, $\ddot{\mathsf{H}}_0$, and $\mathsf{H}_0^{\triangle}$. Our results also apply to other metrics, such as the $L_{\mathfrak{p}}$ metric. In such a case, the null hypotheses, corresponding statistics, and simulated critical values will focus on an integral computation of the function of interest. For example, $\dot{\mathsf{H}}_0$ is replaced by

$$\int \left|\vartheta_{\mathbf{w}}^{(v)}(x) - \frac{\partial^v}{\partial x^v}\eta\left\{M_{\mathbf{w}}(x;\boldsymbol{\theta},\boldsymbol{\gamma}_0)\right\}\right|^{\mathfrak{p}} dx = 0 \quad \text{for some } \boldsymbol{\theta}$$

where $\mathfrak{p}$ is some positive integer no less than 1 (typically $\mathfrak{p} = 2$ for squared deviations) and the corresponding critical value simulation takes the form $\mathfrak{c} = \inf\{c \in \mathbb{R}_+ : \mathbb{P}[\int |\widehat{Z}_p(x)|^{\mathfrak{p}} dx \leq c \mid \mathbf{D}] \geq 1-\alpha\}$. Analogous modifications are done for other hypothesis tests. Note that by construction, the $L_{\mathfrak{p}}$ metric measures the integrated absolute deviation and thus should be applied to two-sided tests only. The choice of metric is implemented in each testing command via the option `lp()`.

2.8.2 Mass points and minimum effective sample size

The package `binsreg` incorporates specific implementation decisions to deal with mass points in the distribution of the independent variable x_i. The number of distinct values of x_i, denoted by N, is taken as the effective sample size as opposed to the total number of observations n. If x_i is continuously distributed, then $N = n$. However, in many applications, N can be substantially smaller than n, which affects some of the implementations in the package.

First, assume that J is set by the user (via the option `nbins(J)`). Then, given the choice J, the commands `binsreg`, `binslogit`, `binsprobit`, `binsqreg`, `binstest`, and `binspwc` perform a degrees-of-freedom check to decide whether the x_i data exhibit enough variation. Specifically, given p and s set by the option `dots(p s)` or `bins(p s)`, these commands check whether $N > N_2 + (p+1)J - (J-1)s$ with $N_2 = 30$ by default. If this check is not passed, then the package `binsreg` regards the data as having "too little" variation in x_i and turns off all nonparametric estimation and inference results based on large sample approximations. Thus, in this extreme case, the command `binsreg` (or `binslogit`, `binsprobit`, `binsqreg`) allows only for `dots(0 0)`, `ci(0 0)`, and `polyreg(P)` for any $P + 1 < N$, while the command `binstest` (or `binspwc`) does not return any results and issues a warning message instead.

If, on the other hand, for given J, the numerical check $N > N_2 + (p+1)J - (J-1)s$ is passed, then all nonparametric methods implemented by the commands `binsreg`, `binslogit`, `binsprobit`, `binsqreg`, `binstest`, and `binspwc` become available. However, before implementing each method (`dots(p s)`, `lines(p s)`, `ci(p s)`, `cb(p s)`, `polyreg(P)`, and the hypothesis testing procedures), a degrees-of-freedom check is performed in each individual case. Specifically, each nonparametric procedure is implemented only if $N > N_2 + (p+1)J - (J-1)s$, where p and s may change from one procedure to the next.

Second, as discussed above, whenever J is not set by the user via the option `nbins()`, the command `binsregselect` is used to select J in a data-driven way, provided there is enough variation in x_i. To determine the latter, an initial degrees-of-freedom check is performed to assess whether J selection is possible or, alternatively, whether the unique values of x_i should be used as bins directly. Specifically, if $N > N_1 + p + 1$, with p set by the option `dots(p s)` (or `bins(p s)`) and $N_1 = 20$ by default, then the data are deemed appropriate for ROT selection of J via the command `binsregselect`, so $\widehat{J}_{\mathrm{ROT}}$ is implemented. If, in addition, $N > N_2 + (p+1)\widehat{J}_{\mathrm{ROT}} - (\widehat{J}_{\mathrm{ROT}} - 1)s$, then $\widehat{J}_{\mathrm{DPI}}$ is also implemented whenever requested. Furthermore, the command `binsregselect` uses the following alternative formula for J selection:

$$J_{\mathrm{IMSE}} = \left\lceil \left\{ \frac{2(p-v+1)\mathscr{B}_n(p,s,v)}{(1+2v)\mathscr{V}_n(p,s,v)} \right\}^{\frac{1}{2p+3}} N^{\frac{1}{2p+3}} \right\rceil$$

This formula has a slightly different constant $\mathscr{V}_n(p,s,v)$, accounting for the frequency of data at each mass point. All other estimators in the package `binsreg`, including bias and standard error estimators, automatically adapt to the presence of mass points. Once the final J is estimated, the degrees-of-freedom checks discussed in the previous paragraphs are performed based on this choice.

If J is not set by the user and $N \leq N_1 + p + 1$, so that not even ROT estimation of J is possible, then N is taken as "too small". In this extreme case, the package `binsreg` sets $J = N$ and constructs a partitioning or binning structure with each bin containing one unique value of x_i. In other words, the support of the raw data is taken as the binning structure itself. In this extreme case, the follow-up degrees-of-freedom checks based on the formula $N > N_2 + (p+1)J - (J-1)s$ fail by construction, so the nonparametric methods are turned off as explained above.

Finally, the specific numerical checks and corresponding adjustments mentioned in this subsection can be modified or omitted. This is controlled by two main options: `dfcheck()` and `masspoints()`, respectively. First, the default cutoff points N_1 and N_2, corresponding to the degrees-of-freedom checks for parametric global polynomial regression and nonparametric binscatter, respectively, can be modified using the option `dfcheck(`N_1 N_2`)`. Second, the option `masspoints()` controls how the package `binsreg` handles the presence of mass points (that is, repeated values) in x_i. Specifically, setting `masspoints(noadjust)` omits mass point checks and the corresponding effective sample-size adjustments; that is, it sets $N = n$ and ignores the presence of mass points in x_i (if any). Setting `masspoints(nolocalcheck)` omits within-bin mass-point checks but still performs global mass-point checks and adjustments. The option `masspoints(off)` corresponds to setting both `masspoints(noadjust)` and `masspoints(nolocalcheck)` simultaneously. Finally, setting `masspoints(veryfew)` forces the package to proceed as if N is so small that all checks are failed, thereby treating x_i as if it has very few distinct values.

2.8.3 Clustered data and minimum effective sample size

As discussed in CCFF, the main methodological results for binscatter can be extended to accommodate clustered data. All commands in the package `binsreg` allow for clustered data via the option `vce()`. In this case, the number of clusters G is taken as the effective sample size, assuming $N = n$ (see below for the other case). The only substantive change occurs in the command `binsregselect`, which now uses the following alternative formula for J selection:

$$J_{\text{IMSE}} = \left\lceil \left\{ \frac{2(p-v+1)\mathscr{B}_n(p,s,v)}{(1+2v)\mathscr{V}_n(p,s,v)} \right\}^{\frac{1}{2p+3}} G^{\frac{1}{2p+3}} \right\rceil$$

This formula has a variance constant $\mathscr{V}_n(p,s,v)$, accounting for the clustered structure of the data. Accordingly, cluster–robust variance estimators are used in this case.

2.8.4 Minimum effective sample size

The package `binsreg` requires some minimal variation in x_i to successfully implement nonparametric methods based on large-sample approximations. The minimal variation is captured by the number of distinct values on the support of x_i, denoted by N, and the number of clusters, denoted by G. Thus, all commands in the package perform degrees-of-freedom numerical checks using $\min\{n, N, G\}$ as the general definition of effective sample size and proceeding as explained above for the case of mass points in the distribution of x_i.

2.9 Increasing speed of execution

The package `binsreg` offers many options and methods, many of which involve nonlinear estimation or simulations, thereby slowing down its speed of execution. Furthermore, to improve the stability and replicability of the package, it implements several robustness checks that may further decrease execution speed, particularly in settings with ultralarge datasets. There are, however, several options and approaches that could be used to improve the speed of execution of the package `binsreg`.

1. *Sorted data.* The core implementations of the package `binsreg` use several algorithms and procedures that require sorted data along the x dimension. If the provided data are not sorted, then the package begins by sorting the data, which slows down the execution (particularly in large datasets).

    - Speed improvement: provide sorted data in x, which may substantially increase execution speed (particularly in ultralarge datasets).

2. *Data distribution.* The methods implemented in the package `binsreg` were developed for continuously distributed data with "enough" variation (for example, enough degrees of freedom within and across bins, appropriate rank conditions for Gram matrices, etc.). Because empirical work may involve data with mass points or other irregularities that can make the default methods fail, the package `binsreg` implements a series of robustness checks before execution (see above for details).

    - Speed improvement: Use option `masspoints(off)` whenever x is known to be (close to) continuously distributed and the data exhibit "enough" regularity.

3. *Number-of-bins selection.* The package `binsreg` selects the number of bins J in a multistep, data-driven, and optimal way, whenever the user does not provide a selection manually (via the options `nbins()` or `bynbins()`). In large datasets, estimating J may be time consuming.

    - Speed improvement: Provide J manually or use option `randcut()` to speed up the process.

4. *Gtools.* The package `binsreg` is open source and, by default, relies exclusively on base commands and functions in Stata (and in Python and R). However, some parts of this algorithm (for example, `pctile`) may be slow in large datasets.

    - Speed improvement: Use the community-contributed package `gtools` (Caceres 2017) via the option `usegtools(on)`. This community-contributed package needs to be installed separately by the user.

5. *Other possibilities.* The package binsreg offers several other options for increasing speed of execution. First, the community-contributed package reghdfe (Correia 2014) could be used when using the binsreg command. Second, for uncertainty quantification and inference, the number of simulations (option nsims()) and the number of grid points for simulation (option simsgrid()) can be decreased to improve speed, which could offer a good alternative for preliminary exploration. Finally, for ultralarge datasets, it may be advisable to begin exploratory analysis with a random sample of the data if the goal is to increase speed of execution of the package binsreg.

3 Illustration of methods

We illustrate the package binsreg using a simulated dataset, which is available in the file binscatter_simdata.dta. In this dataset, y is the outcome variable, x is the independent variable for binning, w is a continuously distributed covariate, t is a binary covariate, and id is a group identifier. Summary statistics of the simulated data are as follows.

```
. use binsreg_simdata
. summarize
    Variable |        Obs        Mean    Std. dev.       Min        Max
-------------+---------------------------------------------------------
           x |      1,000    .4907072    .2932553   .0002281   .9985808
           w |      1,000    .0120224    .5799381  -.9993055   .9973198
           t |      1,000        .515     .500025          0          1
          id |      1,000       250.5    144.4095          1        500
           y |      1,000    .5283884    1.727878  -5.159858   5.751276
-------------+---------------------------------------------------------
           d |      1,000         .45    .4977427          0          1
```

3.1 Estimation, uncertainty quantification, and plotting

The basic syntax for `binsreg` is the following:

```
. binsreg y x w
Sorting dataset on x...
Note: This step is omitted if dataset already sorted by x.
Binscatter plot
Bin selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced
Derivative: 0
──────────────────────────────────────────────
# of observations                 │     1000
# of distinct values              │     1000
# of clusters                     │        .
──────────────────────────────────────────────
Bin/Degree selection:             │
         Degree of polynomial     │        0
  # of smoothness constraints     │        0
                    # of bins     │       21
                imse, bias^2      │    5.420
                  imse, var.      │    1.192
──────────────────────────────────────────────

────────────────────────────────────────
         │      p       s       df
────────────────────────────────────────
 dots    │      0       0       21
────────────────────────────────────────
```

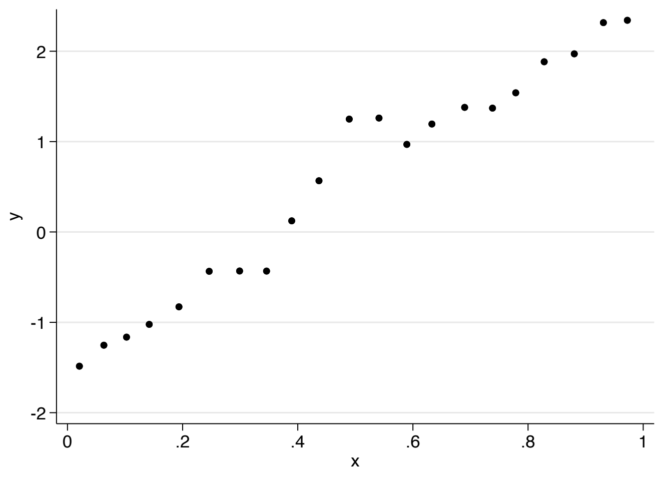


Figure 1. Canonical binned scatterplot

The main output is a binned scatterplot as shown in figure 1. By default, the (nonparametric) mean relationship between `y` and `x` is approximated by piecewise constants (`dots(0 0)`). Each dot in the figure represents the point estimate corresponding to each bin, which is the canonical binscatter plot. The number of bins, whenever not specified, is automatically selected via the companion command **binsregselect**. In this case, 21 bins are used. Other useful information is also reported, including total sample size, the number of distinct values of x, bin-selection results, and the degrees of freedom of the statistical models used.

By default, the command **binsreg** evaluates and plots the regression function of interest $\vartheta^{(v)}_{\mathbf{w}}(x)$ at the mean of the additional covariates $\mathbf{w}_i$, that is, $\mathbf{w} = \overline{\mathbf{w}}$. Users may specify a different value of $\mathbf{w}$, for example, the empirical median of each component in $\mathbf{w}_i$, via the option **at()**:

```
binsreg y x w, at(median)
```

Users may also save the values of the additional covariates at which the binscatter estimate is evaluated in another file and then specify the filename in the option **at()**. For example,

```
. tempfile evalcovar
. preserve
. clear
. set obs 1
Number of observations (_N) was 0, now 1.
. generate w = 0.2
. generate t = 1
. save `evalcovar', replace
(file /var/folders/0b/h0wl9g7d3s3dr9d_vm0qts9r0000gn/T//S_37285.000001 not found)
file /var/folders/0b/h0wl9g7d3s3dr9d_vm0qts9r0000gn/T//S_37285.000001 saved as .dta
    format
. restore
. binsreg y x w i.t, at(`evalcovar')
Sorting dataset on x...
Note: This step is omitted if dataset already sorted by x.
Binscatter plot
Bin selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced
Derivative: 0

 ─────────────────────────────────────────────────────
 # of observations              |        1000
 # of distinct values           |        1000
 # of clusters                  |           .
 ─────────────────────────────────────────────────────
 Bin/Degree selection:          |
          Degree of polynomial  |           0
   # of smoothness constraints  |           0
                     # of bins  |          22
                imse, bias^2    |       4.736
                  imse, var.    |       0.974
 ─────────────────────────────────────────────────────

 ─────────────────────────────────────────────
          |       p         s         df
 ─────────────────────────────────────────────
 dots     |       0         0         22
 ─────────────────────────────────────────────
```

In this case, we control for a continuous variable `w` and a dummy variable generated based on the binary covariate `t`. We evaluate the binscatter estimate at `w` = 0.2 and `t` = 1, and these values are saved in the temporary file `` `evalcovar' `` in advance.

Users may specify the number of bins manually rather than relying on the automatic data-driven procedures. For example, a popular ad hoc choice in practice is setting $J = 20$ quantile-spaced bins:

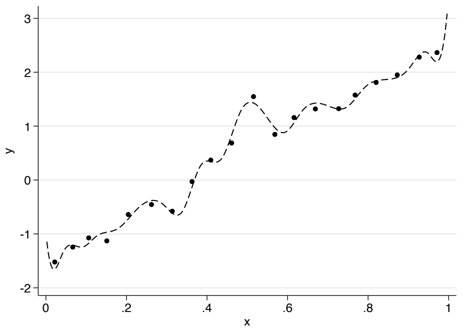
```
. binsreg y x w, nbins(20) polyreg(1)
Sorting dataset on x...
Note: This step is omitted if dataset already sorted by x.
Note: When additional covariates w are included, the polynomial fit may not
> always be close to the binscatter fit.
Binscatter plot
Bin selection method: User-specified
Placement: Quantile-spaced
Derivative: 0
──────────────────────────────────────────────
# of observations              │      1000
# of distinct values           │      1000
# of clusters                  │         .
───────────────────────────────┼──────────────
Bin/Degree selection:          │
        Degree of polynomial   │         0
  # of smoothness constraints  │         0
                   # of bins   │        20
──────────────────────────────────────────────

────────────────────────────────────────
         │       p        s        df
─────────┼──────────────────────────────
 dots    │       0        0        20
 polyreg │       1        NA       2
────────────────────────────────────────
```

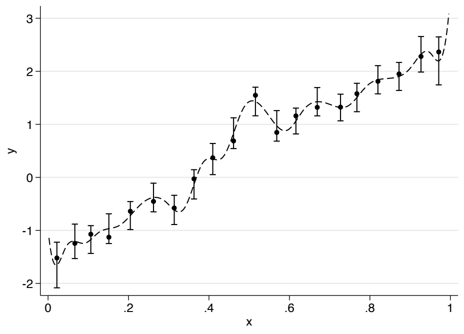
The option `polyreg(1)` adds a linear prediction line to the canonical binscatter plot, but the resulting binned scatterplot is not reported here to conserve space.

The command `binsreg` allows users to add a binscatter-based line approximating the unknown regression function, pointwise confidence intervals, a uniform confidence band, and a global polynomial regression approximation. For example, the following syntax cumulatively adds in four distinct plots a fitted line, confidence intervals, and a confidence band, all three based on cubic B-splines, and also a fitted line based on a global polynomial of degree 4. The results are shown in figure 2.

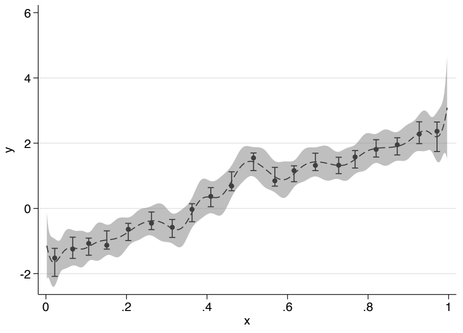
```
. quietly binsreg y x w, nbins(20) dots(0,0) line(3,3)
. quietly binsreg y x w, nbins(20) dots(0,0) line(3,3) ci(3,3)
. quietly binsreg y x w, nbins(20) dots(0,0) line(3,3) ci(3,3) cb(3,3)
. quietly binsreg y x w, nbins(20) dots(0,0) line(3,3) ci(3,3) cb(3,3) polyreg(4)
```



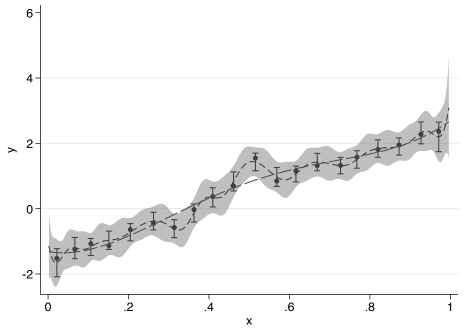
(a) Add cubic B-spline fit



(b) Add confidence intervals



(c) Add confidence band



(d) Add a polynomial fit of degree 4

Figure 2. Binned scatterplot with lines, confidence intervals, and bands

By construction, a cubic B-spline fit is a piecewise cubic polynomial function that is continuous and has continuous first- and second-order derivatives. Thus, the prediction line and confidence band generated are quite smooth. In this case, it is arguably undersmoothed because of the "large" choice of $J = 20$. The degree and smoothness of polynomials can be changed by adjusting the values of `p` and `s` in the options `dots()`, `line()`, `ci()`, and `cb()`.

The command `binsreg` also allows for the standard `vce()` options, factor variables, and `twoway` graph options, among other features. This is illustrated in the following code:

```
. binsreg y x w i.t, dots(0,0) line(3,3) ci(3,3) cb(3,3) polyreg(4)
> vce(cluster id) savedata(graphdat) replace
> title("Binned scatterplot")
Sorting dataset on x...
Note: This step is omitted if dataset already sorted by x.
Note: When additional covariates w are included, the polynomial fit may not
> always be close to the binscatter fit.
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.

Binscatter plot
Bin selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced
Derivative: 0
Output file: graphdat.dta
──────────────────────────────────────────────┬──────────────
# of observations                              |      1000
# of distinct values                           |      1000
# of clusters                                  |       500
──────────────────────────────────────────────┼──────────────
Bin/Degree selection:                          |
         Degree of polynomial                  |         0
  # of smoothness constraints                  |         0
                    # of bins                  |        20
               imse, bias^2                    |     3.588
                 imse, var.                    |     0.494
──────────────────────────────────────────────┴──────────────

─────────┬──────────────────────────────
         |      p       s       df
─────────┼──────────────────────────────
 dots    |      0       0       20
 line    |      3       3       23
 CI      |      3       3       23
 CB      |      3       3       23
 polyreg |      4       NA      5
─────────┴──────────────────────────────
```

Specifically, a dummy variable based on the binary covariate `t` is added to the estimation; standard errors are clustered at the group level indicator `id`; and a graph title is added to the resulting binned scatterplot. Note that any unrecognized options for the command `binsreg` will be understood as `twoway` options and therefore appended to the final plot command. Thus, users may easily modify, for example, axis properties, legends, etc. The option `savedata(graphdat)` saves the underlying data used in the binned scatterplot in the file `graphdat.dta`.

In addition, the command `binsreg` can be used for subgroup analysis. The following command implements binscatter estimation and inference across two subgroups separately, defined by the variable `t`, and then produces a common binned scatterplot (figure 3):

```
. binsreg y x w, by(t) dots(0,0) line(3,3) cb(3,3)
> bycolors(gs4 gs10) bysymbols(O T)
Sorting dataset on x...
Note: This step is omitted if dataset already sorted by x.
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.
Binscatter plot
Bin selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced
Derivative: 0
Group: t = 0
───────────────────────────────┬───────────────
# of observations              │        485
# of distinct values           │        485
# of clusters                  │          .
───────────────────────────────┼───────────────
Bin/Degree selection:          │
         Degree of polynomial  │          0
  # of smoothness constraints  │          0
                    # of bins  │         20
              imse, bias^2     │      7.092
                imse, var.     │      0.941
───────────────────────────────┴───────────────

─────────┬──────────────────────────────
         │       p        s        df
─────────┼──────────────────────────────
 dots    │       0        0        20
 line    │       3        3        23
 CB      │       3        3        23
─────────┴──────────────────────────────
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.
Group: t = 1
───────────────────────────────┬───────────────
# of observations              │        515
# of distinct values           │        515
# of clusters                  │          .
───────────────────────────────┼───────────────
Bin/Degree selection:          │
         Degree of polynomial  │          0
  # of smoothness constraints  │          0
                    # of bins  │         15
              imse, bias^2     │      2.861
                imse, var.     │      0.955
───────────────────────────────┴───────────────

─────────┬──────────────────────────────
         │       p        s        df
─────────┼──────────────────────────────
 dots    │       0        0        15
 line    │       3        3        18
 CB      │       3        3        18
─────────┴──────────────────────────────
```

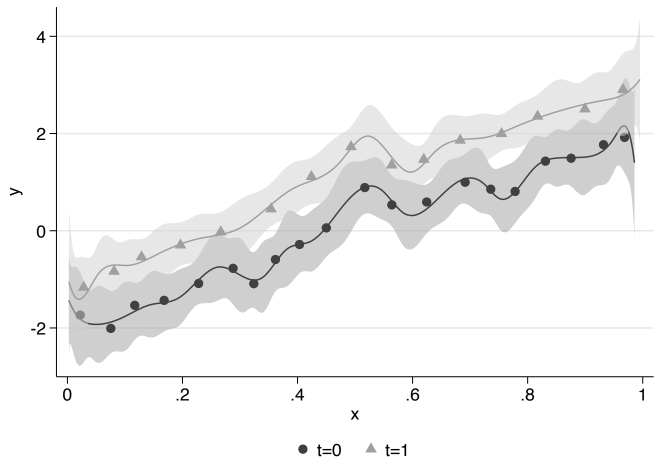


Figure 3. Binned scatterplot: Group comparison

Figure 3 highlights a difference across the two subgroups defined by the variable `t`, which corresponds to the fact that our simulated data add a 1 to the outcome variable for those units with `t` = 1. The colors, symbols, and line patterns in figure 3 can be modified via the options `bycolors()`, `bysymbols()`, and `bylpatterns()`. When the number of bins is unspecified, the command **`binsreg`** selects the number of bins for each subsample separately via the companion command **`binsregselect`**. This means that, by default, the choice of binning or partitioning structure will be different across subgroups in general. However, if the option `samebinsby` is specified, then a common binning scheme for all subgroups is constructed based on the full sample.

As described before, sometimes one would like to keep the number of bins J fixed and select the degree or smoothness of the polynomial instead. The following snippet illustrates how to implement this procedure:

```
. binsreg y x w, nbins(20) line(T) ci(T) cb(T) pselect(0/3)
Sorting dataset on x...
Note: This step is omitted if dataset already sorted by x.
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.

Binscatter plot
Bin selection method: IMSE-optimal plug-in choice (select degree and smoothness)
Placement: Quantile-spaced
Derivative: 0

──────────────────────────────────────────────
                              │
# of observations             │       1000
# of distinct values          │       1000
# of clusters                 │          .
──────────────────────────────┼───────────────
Bin/Degree selection:         │
         Degree of polynomial │          0
  # of smoothness constraints │          0
                    # of bins │         20
                imse, bias^2  │      5.422
                  imse, var.  │      1.181
──────────────────────────────────────────────

────────────────────────────────────────
         │
         │      p       s        df
─────────┼──────────────────────────────
         │
 dots    │      0       0        20
 line    │      0       0        20
 CI      │      1       1        21
 CB      │      1       1        21
────────────────────────────────────────
```

Here we let $J = 20$ and select the degree of polynomial p within the specified range $\{0, 1, 2, 3\}$. The resulting optimal p is 0. Accordingly, we construct "dots" and a "line" based on piecewise constant estimates, and we construct confidence intervals and a confidence band based on linear spline estimates.

The accompanying replication files include other illustrations. For example:

- Inference based on asymptotic variance formula:

```
binsreg y x w i.t, dots(0 0) line(3 3) ci(3 3) cb(3 3) polyreg(4) ///
    vce(cluster id) asyvar(on)
```

- Using the community-contributed command `reghdfe`:

```
binsreg y x w, absorb(t) dots(0 0) line(3 3) ci(3 3) cb(3 3) polyreg(4)
```

- Turning off data distribution robustness checks and using the community-contributed command `gtools`:

```
binsreg y x w, masspoints(off) usegtools(on)
```

Next we illustrate the command `binsqreg` for estimation and uncertainty quantification using quantile regression binscatter methods. The following code looks at the conditional 25th quantile of the outcome variable:

```
. binsqreg y x w, quantile(0.25)
Sorting dataset on x...
Note: This step is omitted if dataset already sorted by x.

Binscatter plot, quantile
Bin selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced
Derivative: 0
──────────────────────────────────────────────┬───────────────
# of observations                             │       1000
# of distinct values                          │       1000
# of clusters                                 │          .
──────────────────────────────────────────────┼───────────────
Bin/Degree selection:                         │
        Degree of polynomial                  │          0
  # of smoothness constraints                 │          0
                    # of bins                 │         21
                imse, bias^2                  │      5.420
                  imse, var.                  │      1.192
──────────────────────────────────────────────┴───────────────

─────────┬──────────────────────────────
         │       p       s       df
─────────┼──────────────────────────────
 dots    │       0       0       21
─────────┴──────────────────────────────
```

By default, quantile regression methods use an analytic variance estimator formula that may not perform well in applications. A more robust alternative is using bootstrap methods:

```
. binsqreg y x w, quantile(0.25) ci(3 3) vce(bootstrap, reps(100))
Sorting dataset on x...
Note: This step is omitted if dataset already sorted by x.

Binscatter plot, quantile
Bin selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced
Derivative: 0
──────────────────────────────┬───────────────
# of observations             │      1000
# of distinct values          │      1000
# of clusters                 │         .
──────────────────────────────┼───────────────
Bin/Degree selection:         │
         Degree of polynomial │         0
  # of smoothness constraints │         0
                    # of bins │        21
                imse, bias^2  │     5.420
                  imse, var.  │     1.192
──────────────────────────────┴───────────────

─────────┬──────────────────────────
         │       p       s       df
─────────┼──────────────────────────
 dots    │       0       0       21
 CI      │       3       3       24
─────────┴──────────────────────────
```

The replication files also illustrate how to plot together least squares and quantile regression binscatter approximations. The final output is illustrated in figure 4, which plots the conditional mean and its confidence band together with the conditional 25th and 75th quantile regressions (that is, the conditional interquartile range).

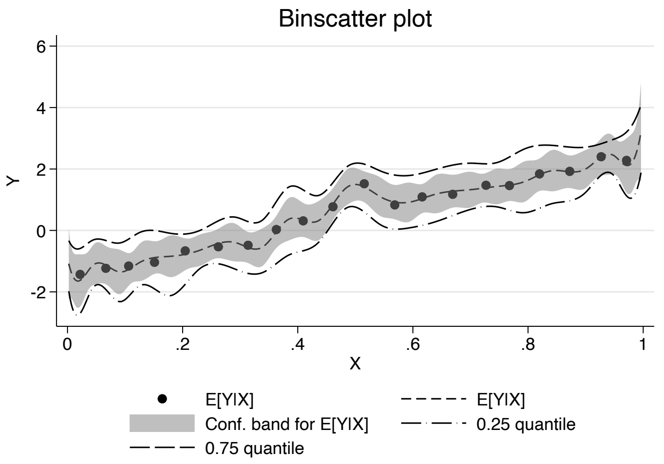


Figure 4. Binned scatterplot for means and quantiles

Moreover, we provide a convenient option, `qregopt()`, to modify the underlying quantile regression. For example, the user can control the optimization process as follows:

```
. quietly binsqreg y x w, quantile(0.25) qregopt(iterate(1000) wls(1))
```

Finally, we illustrate the command `binslogit` for binary response regression binscatter methods using logistic regression.

```
. binslogit d x w
Sorting dataset on x...
Note: This step is omitted if dataset already sorted by x.

Binscatter plot, logit model
Bin selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced
Derivative: 0
──────────────────────────────────────────────┬───────────────
# of observations                             │       1000
# of distinct values                          │       1000
# of clusters                                 │          .
──────────────────────────────────────────────┼───────────────
Bin/Degree selection:                         │
         Degree of polynomial                 │          0
  # of smoothness constraints                 │          0
                    # of bins                 │         12
                imse, bias^2                  │      0.172
                  imse, var.                  │      0.208
──────────────────────────────────────────────┴───────────────

─────────┬──────────────────────────────
         │      p        s        df
─────────┼──────────────────────────────
 dots    │      0        0        12
─────────┴──────────────────────────────
```

3.2 Hypothesis testing and statistical inference

We illustrate first the syntax and output of the command `binstest`. The basic syntax is the following:

```
. binstest y x w, testmodelpoly(1)
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.
Hypothesis tests based on binscatter estimates
Estimation method: reg
Bin selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced
Derivative: 0
───────────────────────────────────────────────────────────────
# of observations                        1000
# of distinct values                     1000
# of clusters                               .
───────────────────────────────────────────────────────────────
Bin/Degree selection:
         Degree of polynomial               0
  # of smoothness constraints               0
                    # of bins              21
───────────────────────────────────────────────────────────────

Model specification Tests:
Degree: 1     # of smoothness constraints: 1
──────────────────────────────────────────────────
H0: mu =                sup |T|          p value
──────────────────────────────────────────────────
poly. degree  1           6.566            0.000
──────────────────────────────────────────────────
```

A test for linearity of the regression function $\mu_0(x)$ is implemented using the binscatter estimator. By default, a linear B-spline is used in the inference procedure, which can be adjusted by the option `testmodel()`. In addition, when unspecified, the number of bins is selected using a data-driven procedure via the companion command `binsregselect`. The selected number of bins is IMSE optimal for piecewise constant point estimates by default. A summary of the sample and binning scheme is displayed, and then the test statistic and p-value are reported. In this case, the test statistic is the supremum of the absolute value of the t statistic evaluated over a sequence of grid points, and the p-value is calculated based on simulation. Clearly, the p-value is quite small, and thus the null hypothesis of linearity of the regression function is rejected.

As emphasized before, the parametric specification test for a null hypothesis about the level may be sensitive to the choice of evaluation point **w**. Thus, a recommended strategy to test for linearity of $\mu_0(x)$ is to check if its first derivative is a constant, which is implemented in the following:

```
. binstest y x w, testmodelpoly(1) deriv(1)
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.
Hypothesis tests based on binscatter estimates
Estimation method: reg
Bin selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced
Derivative: 1
──────────────────────────────────────────────┬───────────────
# of observations                             │         1000
# of distinct values                          │         1000
# of clusters                                 │            .
──────────────────────────────────────────────┼───────────────
Bin/Degree selection:                         │
         Degree of polynomial                 │            1
  # of smoothness constraints                 │            1
                     # of bins                │            6
──────────────────────────────────────────────┴───────────────
Model specification Tests:
Degree: 2     # of smoothness constraints: 2
───────────────────┬──────────────────────────────
H0: mu =           │ sup |T|           p value
───────────────────┼──────────────────────────────
poly. degree  1    │   4.114             0.000
───────────────────┴──────────────────────────────
```

Note that for $v = 1$, the selected number of bins is IMSE optimal for the linear B-spline estimate by default, and the test statistic based on the proposed robust bias-correction strategy is constructed using a quadratic B-spline fit.

The command `binstest` can implement testing for any parametric model specification by comparing the fitted values based on the binscatter estimator (computed by the command) and the parametric model of interest (provided by the user). For example, the following code creates an auxiliary database with a grid of evaluation points, implements a linear regression first, makes an out-of-sample prediction using the auxiliary dataset, and then tests for linearity based on the binscatter estimator by specifying the auxiliary file containing the fitted values.

```
. quietly binsregselect y x w, simsgrid(30) savegrid(parfitval) replace
. quietly regress y x w
. use parfitval, clear
. predict binsreg_fit_lm
(option xb assumed; fitted values)
. save parfitval, replace
file parfitval.dta saved
```

```
. use binsreg_simdata, clear

. binstest y x w, testmodelparfit(parfitval) lp(2) deriv(1)
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.

Hypothesis tests based on binscatter estimates
Estimation method: reg
Bin selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced
Derivative: 1
──────────────────────────────┬───────────────
# of observations             │      1000
# of distinct values          │      1000
# of clusters                 │         .
──────────────────────────────┼───────────────
Bin/Degree selection:         │
         Degree of polynomial │         1
  # of smoothness constraints │         1
                    # of bins │         6
──────────────────────────────┴───────────────

Model specification Tests:
Degree: 2    # of smoothness constraints: 2

Input file: parfitval.dta
───────────────────┬───────────────────────────
H0: mu =           │ L2 of T          p value
───────────────────┼───────────────────────────
   binsreg_fit_lm  │   4.377            0.000
───────────────────┴───────────────────────────
```

The first line above, `binsregselect y x w, simsgrid(30) savegrid(parfitval) replace`, generates the auxiliary file containing the grid of evaluation points. Because the parameter of interest is only the mean relation between `y` and `x`, that is, $\mu_0(x)$, at the out-of-sample prediction step, the testing dataset `parfitval.dta` must contain a variable `x` containing a sequence of evaluation points at which the binscatter and parametric models are compared and the covariate `w` whose values are set to be zeros. In addition, the variable containing fitted values has to follow a specific naming rule; that is, it must take the form of `binsreg_fit*`. The companion command `binsregselect` can be used to construct the required auxiliary dataset, as illustrated above. We discuss this other command further below.

In addition to model specification tests, the command `binstest` can test for nonparametric shape restrictions on the regression function. For example, the following syntax tests whether the regression function is increasing:

```
. binstest y x w, deriv(1) nbins(20) testshaper(0)
Warning: Testing procedures are valid when nbins() is much larger than the IMSE-
> optimal choice. Compare your choice with the IMSE-optimal one obtained by
> binsregselect.
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.

Hypothesis tests based on binscatter estimates
Estimation method: reg
Bin selection method: User-specified
Placement: User-specified
Derivative: 1
──────────────────────────────────────────────
# of observations              │      1000
# of distinct values           │      1000
# of clusters                  │         .
──────────────────────────────────────────────
Bin/Degree selection:          │
         Degree of polynomial  │         .
  # of smoothness constraints  │         .
                    # of bins  │        20
──────────────────────────────────────────────

Shape Restriction Tests:
Degree: 2    # of smoothness constraints: 2
──────────────────────────────────────────────────
H0: inf mu >=       │  inf T             p value
──────────────────────────────────────────────────
         0          │   -3.709             0.004
──────────────────────────────────────────────────
```

The null hypothesis here is that the infimum of the first-order derivative of the regression function is no less than 0. The output reports the test statistic, which is the infimum of the t statistic over a sequence of evaluation points and the corresponding simulation-based p-value.

The command `binstest` may implement many tests at once (given the derivative of interest). For example,

```
. binstest y x w, nbins(20) testshaper(-2 0) testshapel(4) testmodelpoly(1)
> nsims(1000) simsgrid(30)
Warning: Testing procedures are valid when nbins() is much larger than the IMSE-
> optimal choice. Compare your choice with the IMSE-optimal one obtained by
> binsregselect.
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.

Hypothesis tests based on binscatter estimates
Estimation method: reg
Bin selection method: User-specified
Placement: User-specified
Derivative: 0
──────────────────────────────────────────────
                                │
# of observations               │      1000
# of distinct values            │      1000
# of clusters                   │         .
────────────────────────────────┼─────────────
Bin/Degree selection:           │
         Degree of polynomial   │         .
  # of smoothness constraints   │         .
                     # of bins  │        20
──────────────────────────────────────────────

Shape Restriction Tests:
Degree: 1     # of smoothness constraints: 1
──────────────────────────────────────────────────
H0: sup mu <=      │ sup T            p value
───────────────────┼──────────────────────────────
         4         │  -4.131            1.000
──────────────────────────────────────────────────

──────────────────────────────────────────────────
H0: inf mu >=      │ inf T            p value
───────────────────┼──────────────────────────────
         -2        │   1.774            1.000
         0         │ -10.772            0.000
──────────────────────────────────────────────────

Model specification Tests:
Degree: 1     # of smoothness constraints: 1
──────────────────────────────────────────────────
H0: mu =           │ sup |T|          p value
───────────────────┼──────────────────────────────
poly. degree  1    │   5.972            0.000
──────────────────────────────────────────────────
```

The above syntax tests three shape restrictions and one model specification (linearity) using 1,000 random draws from $\mathbf{N}_K^\star$ and 30 evaluation points to evaluate the supremum or infimum in the simulation.

The accompanying replication files include other illustrations:

- Testing whether the median regression function is linear:

```
binstest y x w, estmethod(qreg 0.5) testmodelpoly(1)
```

- Testing whether the nonlinear logistic regression function is increasing:

```
binstest d x w, estmethod(logit) deriv(1) nbins(20) testshaper(0)
```

Next consider pairwise comparison as implemented via the command `binspwc`. Using least-squares binscatter for the two samples identified via the binary variable `t`, we have

```
. binspwc y x w, by(t)
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.

Pairwise group comparison based on binscatter estimates
Estimation method: reg
Derivative: 0
Group variable: t
Bin/Degree selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced

Group 1 vs. Group 0
─────────────────────────────────────────────────────
Group t=                      │         1  │        0
─────────────────────────────────────────────────────

# of observations             │       515  │      485
# of distinct values          │       515  │      485
# of clusters                 │         .  │        .
Degree of polynomial          │         1  │        1
# of smoothness constraints   │         1  │        1
# of bins                     │        15  │       20
─────────────────────────────────────────────────────

diff = group 1 - group 0
──────────────────────────────────────────────────
H0:                 │  sup |T|           p value
──────────────────────────────────────────────────
diff=0              │    5.790             0.000
──────────────────────────────────────────────────
```

Similarly, using quantile regression binscatter methods for the 40th conditional quantile of the outcome variable, we have

```
. binspwc y x w, by(t) estmethod(qreg 0.4)
Note: Setting at least nsims(2000) and simsgrid(50) is recommended to obtain the
> final results.

Pairwise group comparison based on binscatter estimates
Estimation method: qreg
Derivative: 0
Group variable: t
Bin/Degree selection method: IMSE-optimal plug-in choice (select # of bins)
Placement: Quantile-spaced

Group 1 vs. Group 0
----------------------------------------------------
Group t=                      |          1 |       0
----------------------------------------------------
# of observations             |        515 |     485
# of distinct values          |        515 |     485
# of clusters                 |          . |       .
Degree of polynomial          |          1 |       1
# of smoothness constraints   |          1 |       1
# of bins                     |         15 |      20
----------------------------------------------------

diff = group 1 - group 0
--------------------------------------------------
H0:                 | sup |T|            p value
--------------------------------------------------
diff=0              |   5.023              0.000
--------------------------------------------------
```

3.3 Binning selection

As already mentioned, all our estimation and inference commands rely on data-driven bin-selection procedures via the command `binsregselect` whenever the option `nbins()` is not used. Its basic syntax is as follows:

```
. binsregselect y x w
Bin selection for binscatter estimates
Method: IMSE-optimal plug-in choice (select # of bins)
Position: Quantile-spaced
───────────────────────────────────────
             # of observations │   1000
          # of distince values │   1000
                 # of clusters │      .
               eff. sample size │   1000
───────────────────────────────────────
          Degree of polynomial │      0
  # of smoothness constraint   │      0
───────────────────────────────────────

─────────────────────────────────────────────────────────────────────
    method     │  # of bins │      df    │ imse, bias^2 │  imse, var.
─────────────────────────────────────────────────────────────────────
   ROT-POLY    │      18    │      18    │     3.295    │     1.212
   ROT-REGUL   │      18    │      18    │         .    │         .
   ROT-UKNOT   │      18    │      18    │         .    │         .
      DPI      │      21    │      21    │     5.420    │     1.192
   DPI-UKNOT   │      21    │      21    │         .    │         .
─────────────────────────────────────────────────────────────────────
df: degrees of freedom.
```

The following choices for the number of bins are reported: `ROT-POLY`, the ROT choice based on global polynomial estimation; `ROT-REGUL`, the ROT choice regularized as discussed in section 2 or the user's choice specified in the option `nbinsrot()`; `ROT-UKNOT`, the ROT choice with unique knots; `DPI`, the DPI choice; and `DPI-UKNOT`, the DPI choice with unique knots.

The DPI choice is implemented based on the ROT choice, which can be set by users directly:

```
. binsregselect y x w, nbinsrot(20) binspos(es)
Bin selection for binscatter estimates
Method: IMSE-optimal plug-in choice (select # of bins)
Position: Evenly-spaced
─────────────────────────────────────────────
             # of observations │      1000
          # of distince values │      1000
                 # of clusters │         .
               eff. sample size │      1000
─────────────────────────────────────────────
           Degree of polynomial │         0
   # of smoothness constraint  │         0
─────────────────────────────────────────────

─────────────────────────────────────────────────────────────────────
     method   │  # of bins  │     df     │ imse, bias^2  │ imse, var.
─────────────────────────────────────────────────────────────────────
    ROT-POLY  │          .  │         .  │            .  │          .
   ROT-REGUL  │         20  │        20  │            .  │          .
   ROT-UKNOT  │         20  │        20  │            .  │          .
        DPI   │         22  │        22  │        5.793  │      1.194
   DPI-UKNOT  │         22  │        22  │            .  │          .
─────────────────────────────────────────────────────────────────────
df: degrees of freedom.
```

Note that in the example above, an evenly spaced rather than a quantile-spaced binning scheme is selected via the option `binspos(es)`. The binning used in other commands may be adjusted similarly.

In addition, as illustrated above, the command **binsregselect** provides a convenient option **savegrid()**, which can be used to generate the auxiliary dataset needed for parametric specification testing of user-chosen models via the command **binstest**. Specifically, the following command was (quietly) used above:

```
. binsregselect y x w, simsgrid(30) savegrid(parfitval) replace

Bin selection for binscatter estimates
Method: IMSE-optimal plug-in choice (select # of bins)
Position: Quantile-spaced
Output file: parfitval.dta
───────────────────────────────────────
          # of observations |   1000
       # of distince values |   1000
              # of clusters |      .
           eff. sample size |   1000
───────────────────────────────────────
       Degree of polynomial |      0
 # of smoothness constraint |      0
───────────────────────────────────────

────────────────────────────────────────────────────────────────────
     method    |  # of bins  |     df    | imse, bias^2 |  imse, var.
────────────────────────────────────────────────────────────────────
   ROT-POLY    |      18     |     18    |    3.295     |    1.212
   ROT-REGUL   |      18     |     18    |        .     |        .
   ROT-UKNOT   |      18     |     18    |        .     |        .
      DPI      |      21     |     21    |    5.420     |    1.192
   DPI-UKNOT   |      21     |     21    |        .     |        .
────────────────────────────────────────────────────────────────────
df: degrees of freedom.
```

The resulting file, `parfitval.dta`, includes `x` and `w`, as well as some other variables related to the binning scheme. The variable `x` contains a sequence of evaluation points, in this case set to 30 within each bin via the option **simsgrid()**, and the values of `w` are set to 0 on purpose (this is used to generate fitting the model correctly).

When an extremely large dataset is available, the data-driven procedures for selecting the binning scheme could be very time consuming. In such a scenario, one could use a small subsample to estimate the leading constants in the IMSE expansions and then extrapolate the optimal number of bins to the full sample. The following code illustrates how this method is implemented:

```
. binsregselect y x w if t==0, useeffn(1000)
Bin selection for binscatter estimates
Method: IMSE-optimal plug-in choice (select # of bins)
Position: Quantile-spaced
───────────────────────────────────────
             # of observations │    485
          # of distince values │    485
               # of clusters   │      .
             eff. sample size  │   1000
───────────────────────────────────────
          Degree of polynomial │      0
  # of smoothness constraint   │      0
───────────────────────────────────────

─────────────────────────────────────────────────────────────────────
    method     │ # of bins │     df     │ imse, bias^2 │  imse, var.
─────────────────────────────────────────────────────────────────────
   ROT-POLY    │     20    │     20     │     3.185    │    0.937
   ROT-REGUL   │     20    │     20     │         .    │        .
   ROT-UKNOT   │     20    │     20     │         .    │        .
      DPI      │     26    │     26     │     7.092    │    0.941
   DPI-UKNOT   │     26    │     26     │         .    │        .
─────────────────────────────────────────────────────────────────────
df: degrees of freedom.
```

In this example, 485 observations with $\texttt{t} = 0$ are used to compute the leading constants $\mathscr{B}_n(p, s, v)$ and $\mathscr{V}_n(p, s, v)$ in the IMSE expansion, but then the reported optimal numbers of bins are calculated based on the effective sample size specified in the option `useeffn()`. This method also applies to extrapolating the optimal number of bins to a smaller sample based on a larger one.

Alternatively, the number-of-bins selection can be implemented using only a random subsample of the observations. For example, the following command selects J using, in expectation, 30% of the observations based on uniformly distributed random numbers. Repeated application of this command will produce modestly different bin-selection choices depending on the sequence of realized uniform random variables.

```
. binsregselect y x w, randcut(0.3)
Bin selection for binscatter estimates
Method: IMSE-optimal plug-in choice (select # of bins)
Position: Quantile-spaced
───────────────────────────────────────────────
            # of observations │        1000
        # of distince values  │        1000
               # of clusters  │           .
            eff. sample size  │        1000
───────────────────────────────────────────────
          Degree of polynomial │           0
   # of smoothness constraint  │           0
───────────────────────────────────────────────

──────────────────────────────────────────────────────────────────────────────
      method     │  # of bins   │      df     │  imse, bias^2  │  imse, var.
──────────────────────────────────────────────────────────────────────────────
     ROT-POLY    │         18   │        18   │       3.714    │       1.303
     ROT-REGUL   │         18   │        18   │           .    │           .
     ROT-UKNOT   │         18   │        18   │           .    │           .
       DPI       │         23   │        23   │       7.104    │       1.289
     DPI-UKNOT   │         23   │        23   │           .    │           .
──────────────────────────────────────────────────────────────────────────────
df: degrees of freedom.
```

4 Conclusion

We have introduced the package **binsreg**, which provides general-purpose software implementations of binned scatterplots in different statistical models. The package is based on Cattaneo et al. (2024a,b), which yields a thorough treatment and broad applicability but does entail several limitations. Chiefly, **binsreg** focuses on cross-sectional settings and, by focusing on standard binscatters, univariate x for plotting purposes. However, the general idea of binning data for visualization and for inference has appeal outside these areas and would make for useful extensions of our work. Binning is a commonly used method, for example, in empirical finance, where it is used to create portfolios for factor creation and anomaly detection (Cattaneo et al. 2020). Extending **binsreg** to panel data to cover such cases would be valuable. The same is true for time-series data, where the binning may need to be adapted to accommodate or elucidate features of the dependence in the data. Finally, univariate binscatters are closely related to bivariate heat maps, a popular visualization tool in social and physical sciences, and more broadly even in data-based journalism. It is an open but important question of how to add the statistical rigor we have brought to binscatters to the world of heat maps, or beyond into multidimensional binning.

5 Acknowledgments

We thank Michael Droste, John Friedman, Andreas Fuster, Filippo Palomba, Paul Goldsmith-Pinkham, David Lucca, Xinwei Ma, Ricardo Masini, Jonah Rockoff, Jesse Rothstein, Ryan Santos, Jesse Shapiro, and Rocio Titiunik for helpful comments and discussions. We also thank the editor, Stephen Jenkins, and an anonymous reviewer for their detailed comments that improved our article. The views expressed in this article are those of the authors and do not necessarily reflect the position of the Federal Reserve Bank of New York or the Federal Reserve System. Cattaneo gratefully acknowledges financial support from the National Science Foundation through grants SES-1947805, SES-2019432, and SES-2241575. Feng gratefully acknowledges financial support from the National Natural Science Foundation of China through grants 72203122, 72133002, and 72250064.

6 Programs and supplemental material

To install the software files as they existed at the time of publication of this article, type

```
. net sj 25-1
. net install st0765      (to install program files, if available)
. net get st0765          (to install ancillary files, if available)
```

The latest version of the package `binsreg` and other related materials can be found at https://nppackages.github.io/binsreg/.

7 References

Caceres, M. 2017. gtools. GitHub. https://github.com/mcaceresb/stata-gtools/.

Calonico, S., M. D. Cattaneo, and M. H. Farrell. 2018. On the effect of bias estimation on coverage accuracy in nonparametric inference. *Journal of the American Statistical Association* 113: 767–779. https://doi.org/10.1080/01621459.2017.1285776.

———. 2022. Coverage error optimal confidence intervals for local polynomial regression. *Bernoulli* 28: 2998–3022. https://doi.org/10.3150/21-BEJ1445.

Calonico, S., M. D. Cattaneo, and R. Titiunik. 2014. Robust nonparametric confidence intervals for regression-discontinuity designs. *Econometrica* 82: 2295–2326. https://doi.org/10.3982/ECTA11757.

Cattaneo, M. D., R. K. Crump, M. H. Farrell, and Y. Feng. 2024a. Nonlinear binscatter methods. arXiv:2407.15276 [stat.EM], https://doi.org/10.48550/arXiv.2407.15276.

———. 2024b. On binscatter. *American Economic Review* 114: 1488–1514. https://doi.org/10.1257/aer.20221576.

Cattaneo, M. D., R. K. Crump, M. H. Farrell, and E. Schaumburg. 2020. Characteristic-sorted portfolios: Estimation and inference. *Review of Economics and Statistics* 102: 531–551. https://doi.org/10.1162/rest_a_00883.

Cattaneo, M. D., M. H. Farrell, and Y. Feng. 2020. Large sample properties of partitioning-based series estimators. *Annals of Statistics* 48: 1718–1741. https://doi.org/10.1214/19-AOS1865.

Correia, S. 2014. reghdfe: Stata module to perform linear or instrumental-variable regression absorbing any number of high-dimensional fixed effects. Statistical Software Components S457874, Department of Economics, Boston College. https://ideas.repec.org/c/boc/bocode/s457874.html.

Correia, S., and N. Constantine. 2014. reghdfe: Linear regressions with multiple fixed effects. GitHub. https://github.com/sergiocorreia/reghdfe/.

Droste, M. 2019. binscatter2. GitHub. https://github.com/mdroste/stata-binscatter2/.

Healy, K. 2019. *Data Visualization: A Practical Introduction*. Princeton, NJ: Princeton University Press.

Schwabish, J. 2021. *Better Data Visualizations: A Guide for Scholars, Researchers, and Wonks*. New York: Columbia University Press.

Starr, E., and B. Goldfarb. 2020. Binned scatterplots: A simple tool to make research easier and better. *Strategic Management Journal* 41: 2261–2274. https://doi.org/10.1002/smj.3199.

Stepner, M. 2013. binscatter: Binned scatterplots. GitHub. https://github.com/michaelstepner/binscatter/.

About the authors

Matias D. Cattaneo is a professor of operations research and financial engineering at Princeton University.

Richard K. Crump is a financial research advisor in macrofinance studies at the Federal Reserve Bank of New York.

Max H. Farrell is an associate professor of economics at the University of California at Santa Barbara.

Yingjie Feng is an associate professor of economics at Tsinghua University.