The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–48

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 Chicago
Chicago, IL
max.farrell@chicagobooth.edu

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

Abstract. We introduce the `Stata` package Binsreg, which implements the binscatter methods developed in Cattaneo, Crump, Farrell, and Feng (2021). The package includes seven commands: **`binsreg`**, **`binslogit`**, **`binsprobit`**, **`binsqreg`**, **`binstest`**, **`binspwc`** and **`binsregselect`**. The first four commands implement point estimation and uncertainly quantification (confidence intervals and confidence bands) procedures for canonical and extended least squares binscatter regression (**`binsreg`**) as well as generalized non-linear binscatter regression (**`binslogit`** for Logit regression, **`binsprobit`** for Probit regression, **`binsqreg`** for quantile regression). These commands also offer binned scatter plots, allowing for one- and multi-sample settings. The next two commands focus on pointwise and uniform inference: **`binstest`** implements hypothesis testing procedures for parametric specification and for nonparametric shape restrictions of the unknown regression function, while **`binspwc`** implements multi-group pairwise statistical comparisons. These two commands cover both least squares as well as generalized non-linear binscatter methods. All our methods allow for multi-sample analysis, which is useful when studying treatment effects heterogeneity in randomized and observational studies. Finally, the command **`binsregselect`** implements data-driven number of bins selectors for binscatter methods using either quantile-spaced or evenly-spaced binning/partitioning. All the commands allow for covariate adjustment, smoothness restrictions, weighting and clustering, among many other features. Companion `Python` and `R` packages with similar syntax and capabilities are also available.

Keywords: st0001, binscatter, binned scatter plot, nonparametrics, semiparametrics, partitioning estimators, B-splines, tuning parameter selection, confidence bands, shape and specification testing.

July 23, 2021

The views expressed in this paper 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.

 st0001

1 Introduction

Binscatter has become a popular methodology in applied microeconomics since its introduction (Chetty and Szeidl 2006; Chetty, Looney, and Kroft 2009; Chetty, Friedman, Olsen, and Pistaferri 2011b; Chetty, Friedman, Hilger, Saez, Schanzenbach, and Yagan 2011a; Chetty, Friedman, and Rockoff 2014, among many other references). See Starr and Goldfarb (2020) for a recent heuristic overview of these methods. Binscatter techniques offer flexible, yet parsimonious ways of visualizing and summarizing "big data" in regression settings. They can also be used for formal estimation and inference, including testing of substantive hypotheses such as linearity or monotonicity, of the regression function and its derivatives. Despite its popularity among empirical researchers, little was known about the statistical properties of binscatter until very recently: Cattaneo, Crump, Farrell, and Feng (2021) offered the first foundational and comprehensive analysis of binscatter methods, giving an array of theoretical and practical results that aid both in understanding current practices (i.e., their validity or lack thereof) and in offering theory-based guidance for future applications.

This paper introduces the (`Python`, `R` and) `Stata` package `Binsreg`, which includes seven commands implementing the main methodological results in Cattaneo, Crump, Farrell, and Feng (2021). 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 non-linear binscatter methods (i.e., 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 non-linear settings. Furthermore, all three commands allow for multi-sample estimation, uncertainty quantification, and plotting.

- *Hypothesis testing and statistical inference.* The command `binstest` implements hypothesis testing procedures for parametric specification and for nonparametric shape restrictions of the unknown regression function, while `binspwc` implements multi-group pairwise comparisons. These two commands offer the same flexibility and features as the estimation commands `binsreg`, `binslogit`, `binsprobit` and `binsqreg`, and therefore allow for linear and non-linear least squares and generalized binscatter methods with within-bin higher-order polynomial fits, across-bins smoothness restrictions, and semi-linear 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 as well as 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 number of bins selectors for binscatter implementation using either quantile-spaced or evenly-spaced binning/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 by default mass point and degrees of freedom checks and adjustments, which improve the stability of the implementation. Second, the commands allow for multi-way fixed effect and one-way clustering estimation and inference using base commands available in `Stata` (and also, as appropriate, in `Python` and `R`). Third, only in `Stata`, the commands offer the option of estimation and inference with multi-way fixed effects and multi-way clustering via the community-distributed package `reghdfe` (Correia and Constantine 2021). Third, only in `Stata`, the commands allow for using the community-distributed package `gtools` (Caceres 2020), instead of our internal implementations, to potentially increase the speed of internal computations with ultra-large datasets. Depending on the data size and structure, the commands in the package Binsreg may improve implementation speed in `Stata` when (i) mass point and degrees of freedom checks are turned off, (ii) the user-written packages `reghdfe` and `gtools` instead of our internal (open-source) implementations are used. See Section 3.6 for more discussion on the speed of execution of the package Binsreg.

There exist two other popular `Stata` commands implementing binscatter methods: `binscatter` (Stepner 2017) and `binscatter2` (Droste 2019). The package Binsreg offers several new capabilities relative to those other commands, in addition to also implementing covariate adjustment in a different, more principled way. First, the package Binsreg implements binscatter methods allowing for within-bin higher-order polynomial fitting, across-bins smoothness restrictions for both least squares and generalized non-linear binscatter regressions. These features enable derivative estimation for linear and non-linear parameters of interest (e.g., partial effects in discrete choice settings or quantile regression), in addition to producing more smooth approximations of the regression function and transformations thereof. Further, the package `Binsreg` provides formal uncertainty quantification: it implements valid confidence intervals and confidence bands for least squares and generalized non-linear binscatter. Second, while all binscatter packages available allow for covariate adjustment, each one does it in a very different way: `binscatter` and `binscatter2` employ a residualization approach, while Binsreg employs a semi-linear approach. As shown in Cattaneo, Crump, Farrell, and Feng (2021), the residualization approach is inconsistent for the target function of interest in general, while the semi-linear approach is either consistent (under correct specification) or has a clear probability limit interpretation (under misspecification). Finally, the two hypothesis testing and statistical inference commands (`binstest` and `binspwc`) as well as the binning scheme selection command (`binsregselect`) in the package Binsreg are novel implementation methods. None of these features are available in the commands `binscatter` and `binscatter2`.

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 discuss some of the main options and related syntax details for each of the commands in the package. Due to the high flexibility that each of this commands has, we focus only on those syntax options that are potentially most useful for empirical work. The help files for each command contain a detailed description of all available options. Section 4 gives a numerical illustration, while Section 5 concludes. The latest version of this software, as well as other related software and materials, can be found at:

https://nppackages.github.io/binsreg/

2 Overview of Methods and Implementation Details

This section summarizes the main methods implemented in the package Binsreg. For further methodological and theoretical details see Cattaneo, Crump, Farrell, and Feng (2021, CCFF hereafter) and references therein. For a function $f(x)$, we define $f^{(v)}(x) = d^v f(x)/dx^v$, with the usual notation $f(x) = f^{(0)}(x)$.

Given a random sample $(y_i, x_i, \mathbf{w}_i')$, $i = 1, 2, \ldots, n$, binscatter seeks to flexibly approximate the function

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

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

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

with $\rho(\cdot;\cdot)$ and $\eta(\cdot)$ user-chosen loss and (inverse) link functions, respectively, and $\mathcal{M}$ an appropriate space of functions satisfying certain conditions (see below). Several settings of applied interest are covered by this formulation:

- *Semi-linear regression*: $\rho(y; \eta) = (y-\eta)^2$ and $\eta(u) = u$. In this case, the parameter of interest becomes

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

  For example, $\vartheta_{\mathbf{w}}(x)$ (resp. $\vartheta_{\mathbf{w}}^{(1)}(x)$) corresponds to the average (partial) effect of x on y for level $\mathbf{w}_i = \mathbf{w}$. In this setting, $\vartheta_{\mathbf{0}}^{(v)}(x) = \mu_0^{(v)}(x)$, 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. In this case, the parameter of interest becomes

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

  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 τ-th quantile. In this setting, $\mu_0^{(v)}(x)$ itself is usually not of interest in some applications. In this case, the parameter of interest becomes
$$\vartheta_{\mathbf{w}}^{(v)}(x) = \frac{\partial^v}{\partial x^v} Q_\tau(y_i|x_i = x, \mathbf{w}_i = \mathbf{w}),$$
where $Q_\tau(y_i|x_i = x, \mathbf{w}_i = \mathbf{w}) = \mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0$ denotes the conditional τ-th 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 (i.e., relative to the assumptions on the probability distribution of the data $(y_i, x_i, \mathbf{w}_i')$, $i = 1, 2, \ldots, n$). 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 $\vartheta_{\mathbf{w}}^{(v)}(x)$ can only be interpreted as the solution to the minimization in (2). For example, in the semi-linear regression case under misspecification, $\vartheta_{\mathbf{w}}(x) \neq \mathbb{E}[y_i|x_i = x, \mathbf{w}_i = \mathbf{w}]$ but rather $\vartheta_{\mathbf{w}}(x) = \mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0$ corresponds to the best approximation in mean squared error (MSE) to the unknown conditional expectation $\mathbb{E}[y_i|x_i = x, \mathbf{w}_i = \mathbf{w}]$, using functions of the form $\{f(x,\mathbf{w}) = \mu(x) + \mathbf{w}'\boldsymbol{\gamma} : \mu \in \mathcal{M}, \boldsymbol{\gamma} \in \mathbb{R}^d\}$. See Angrist and Pischke (2008), and references therein, for more discussion on the role of misspecification in applied microeconometrics. As another example, in the quantile regression case, $\vartheta_{\mathbf{w}}(x)$ may not equal the true (population) conditional τ-th quantile regression function of y_i given $x_i = x, \mathbf{w}_i = \mathbf{w}$, denoted by $Q_\tau(y_i|x_i = x, \mathbf{w}_i = \mathbf{w})$ above, but nevertheless the pseudo-true parameter $\vartheta_{\mathbf{w}}^{(v)}(x)$ can be useful in applied work (e.g., Angrist, Chernozhukov, and Fernández-Val 2006, and references therein).

2.1 Binscatter Construction

Using the general setup above, we can now describe the binscatter approach in general. To approximate $\mu_0(x)$ and its derivatives in model (2), binscatter first partitions the support of x_i into J quantile-spaced bins, leading to the partitioning scheme:

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

where $x_{(i)}$ denotes the i-th 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. See below for more implementation details.

Then, given the quantile-spaced partitioning/binning scheme, for a choice of number of bins J, and a user's choice of loss function $\rho(\cdot;\cdot)$ and (inverse) link function $\eta(\cdot)$, the generalized non-linear binscatter estimator of the v-th derivative of $\mu_0(x)$ in (2), employing a p-th order polynomial approximation within each bin, imposing s-times differentiability across bins, and adjusting for additional covariates $\mathbf{w}_i$, takes the form:

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

where

$$\widehat{\mu}^{(v)}(x) = \widehat{\mathbf{b}}^{(v)}_s(x)'\widehat{\boldsymbol{\beta}}, \qquad \begin{bmatrix} \widehat{\boldsymbol{\beta}} \\ \widehat{\boldsymbol{\gamma}} \end{bmatrix} = \underset{\boldsymbol{\beta},\boldsymbol{\gamma}}{\arg\min} \sum_{i=1}^{n} \rho\Big(y_i;\ \eta\big(\widehat{\mathbf{b}}_s(x_i)'\boldsymbol{\beta} + \mathbf{w}_i'\boldsymbol{\gamma}\big)\Big), \tag{4}$$

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

$$\widehat{\mathbf{b}}(x) = \left[\ \mathbb{1}_{\widehat{\mathcal{B}}_1}(x) \quad \mathbb{1}_{\widehat{\mathcal{B}}_2}(x) \quad \cdots \quad \mathbb{1}_{\widehat{\mathcal{B}}_J}(x)\ \right]' \otimes \left[\ 1 \quad x \quad \cdots \quad x^p\ \right]',$$

being the p-th 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: $\widehat{\mathbf{b}}(x) = \widehat{\mathbf{b}}_0(x)$ is the basis used for (disjoint) piecewise p-th 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 least squares p-th 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, $s = 2$ forces these fits to be connected and continuously differentiable at the boundaries of adjacent bins, and so on for each $s = 3, 4, \ldots, p$.

Enforcing smoothness on binscatter boils down to incorporating restrictions on the basis of approximation. The resulting constrained basis, $\widehat{\mathbf{b}}_s(x)$, 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 employs $\widehat{\mathbf{T}}_s$ leading to B-splines, which tend to have very good finite sample properties.

It follows that the estimator binscatter $\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x)$ in (3) is constructed as a plug-in estimator for (2). Returning to the settings of applied interest mentioned previously, we have:

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

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

  This estimator is obtained by running the linear least squares regression of y_i on $(\widehat{\mathbf{b}}_s(x_i)', \mathbf{w}_i')$, and then constructing predicted values at $(x, \mathbf{w}')$ for $v = 0$, or

predicted values $\widehat{\mu}^{(v)}(x) = \widehat{\mathbf{b}}_s^{(v)}(x)'\widehat{\boldsymbol{\beta}}$ for $v \geq 1$.
The command `binsreg` provides implementation for this case.

- *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. This case corresponds to non-linear Logistic or Probit binscatter, where the estimator becomes

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

  This estimator is obtained by running the Logit or Probit non-linear regression of y_i on $(\widehat{\mathbf{b}}_s(x_i)', \mathbf{w}_i')$, and constructing predicted values at $(x, \mathbf{w}')$ for $v = 0$, or derivatives thereof for $v \geq 1$.
  The commands `binslogit` and `binsprobit` provide implementation for these cases.

- *Quantile regression*: $\rho(y;\eta) = \ell_\tau(y - \eta)$ and $\eta(u) = u$, where $\ell_\tau(u)$ denotes the check function associated with the τ-th quantile. This case corresponds to non-linear, non-differentiable 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 $\widehat{Q}_\tau(y_i|x_i = x, \mathbf{w}_i = \mathbf{w}) = \widehat{\mu}(x) + \mathbf{w}'\widehat{\boldsymbol{\gamma}}$ denotes the estimate of the conditional τ-th quantile function of y_i given $(x_i, \mathbf{w}_i')$. This estimator is obtained by running the quantile regression of y_i on $(\widehat{\mathbf{b}}_s(x_i)', \mathbf{w}_i')$, and constructing predicted values at $(x, \mathbf{w}')$ for $v = 0$, or derivatives thereof for $v \geq 1$.
  The command `binsqreg` provides implementation for 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 $\mathbf{w} = \mathbf{0}$, $\mathbf{w} = \bar{\mathbf{w}} = \frac{1}{n}\sum_{i=1}^n \mathbf{w}_i$ or $\mathbf{w} = \text{median}(\mathbf{w}_i)$, with $\mathbf{0}$ denoting a vector of zeros and $\mathbf{w} = \text{median}(\mathbf{w}_i)$ 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 (e.g., zero for binary variables), while for continuous variables it may be better to set those components at some other place of their distribution (e.g., mean or some quantile).

Canonical Binscatter

Canonical binscatter, as implemented in the packages `binscatter` and `binscatter2`, correspond to semi-linear least squares regression ($\rho(y;\eta) = (y-\eta)^2$ and $\eta(u) = u$) with $p = s = 0$ and without covariate adjustment (i.e., not including $\mathbf{w}_i$ in (4)). Specifically, in canonical binscatter the basis $\widehat{\mathbf{b}}(x)$ is a J-dimensional vector of orthogonal dummy variables, that is, the j-th component of $\widehat{\mathbf{b}}(x)$ records whether the evaluation point x belongs to the j-th 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: $\bar{y}_j = \frac{1}{N_j}\sum_{i=1}^n \mathbb{1}_{\widehat{\mathcal{B}}_j}(x_i) y_i$ for $j = 1, 2, \ldots, J$. Empirical work employing canonical binscatter typically plots these binned sample averages along with some other estimate(s) of the regression function $\mu_0(x)$.

Covariate-Adjusted Binscatter

Prior work employing binscatter methods, including the packages `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 shows that residualized canonical binscatter is very hard to rationalize or justify in general, 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 model (2), and is therefore principled and interpretable. Even when model (2) is misspecified, the approach to covariate adjustment employed 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.

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/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 $\mathbf{w}$. Defaults are `deriv(0)` and `at(mean)`, where the second option corresponds to $\mathbf{w} = \bar{\mathbf{w}}$.

For example, when using `binsreg`, the default implementation estimates $\widehat{\vartheta}_{\bar{\mathbf{w}}}(x) = \widehat{\mathbb{E}}[y_i|x_i = x, \mathbf{w}_i = \bar{\mathbf{w}}] = \widehat{\mu}(x) + \bar{\mathbf{w}}'\widehat{\boldsymbol{\gamma}}$. Thus, `dots(0,0)` leads to "dots" representing sample averages within each bin for the "long" regression with $\mathbf{w}_i = \bar{\mathbf{w}}$. In particular, if $\mathbf{w}_i$ are not included, then the default coincides with Canonical Binscatter (i.e., the same results would be obtained using the packages `binscatter` and `binscatter2`). 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 $\mu^{(v)}_0(x)$.

The common partitioning/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 (e.g., `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 partitioning/binning can be implemented via the option `binspos()`.

Several other options are available for the four estimation commands, including multi-way fixed effects and multi-way clustering adjustments. See Section 3 for syntax discussion and further details.

2.2 Choosing the Number of Bins

CCFF develops valid integrated mean squared error (IMSE) approximations for generalized non-linear binscatter in the context of model (2). These expansions give IMSE-optimal selection of the number bins J forming the quantile-spaced (or other) partitioning scheme $\widehat{\Delta}$, depending on polynomial order p within bins and smoothness level s across bins, the target estimand set by derivative order v, and the evaluation point of interest $\mathbf{w}$ for covariate adjustment. Specifically, the IMSE-optimal choice of J is given by:

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

where $\lceil \cdot \rceil$ denotes the ceiling operator, and $\mathscr{B}_n(p,s,v)$ 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. These constants depend on the partitioning scheme and binscatter estimator used. Recall that these three integer choices must respect $p \geq s \geq 0$ and $p \geq v \geq 0$.

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. Note that for generalized non-linear 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 non-linear case.

In practice, 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. Our main implementation offers two J selectors:

- $\widehat{J}_{\texttt{ROT}}$: implements a rule-of-thumb (ROT) approximation for the constants $\mathscr{B}_n(p,s,v)$ and $\mathscr{V}_n(p,s,v)$, employing a trimmed-from-below Gaussian reference model for the density of x_i, and global polynomial approximations for the other two unknown features needed, $\mu^{(v)}_0(x)$ and $\mathbb{V}[y_i|x_i = x]$. This J selector employs the correct rate but an inconsistent constant approximation.

- $\widehat{J}_{\texttt{DPI}}$: implements a direct-plug-in (DPI) approximation for the constants $\mathscr{B}_n(p,s,v)$ and $\mathscr{V}_n(p,s,v)$, based on the desired binscatter, set by the choices p and s, and employing a preliminary J. If a preliminary J is not provided by the user, then $J = \max\{\widehat{J}_{\texttt{ROT}}, \lceil (\frac{2(p-v+1)}{1+2v} n)^{\frac{1}{2p+3}} \rceil\}$ is used for DPI implementation. Therefore, the selector $\widehat{J}_{\texttt{DPI}}$ employs the correct rate and a nonparametric consistent constant approximation in the latter case.

The precise form of the constants changes depending on whether quantile-spaced or evenly-space partitioning/binning is used, but the two methods for selecting J, ROT and DPI, remain conceptually the same. See CCFF for more details.

Main implementation details

The command `binsregselect` implements 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/binning. For DPI implementation, the user can provide the initialization value of J or, if not provided, then $\widehat{J}_{\texttt{ROT}}$ is used.

Several other options are available for the command `binsregselect`, including the possibility of generating an output file with the IMSE-optimal partitioning/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. See Section 3 for more discussion on syntax and implementation.

2.3 Confidence Intervals

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

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

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

$$\widehat{\Omega}(x) = \widehat{\mathbf{b}}^{(v)}_s(x)'\widehat{\mathbf{Q}}^{-1}\widehat{\mathbf{\Sigma}}\widehat{\mathbf{Q}}^{-1}\widehat{\mathbf{b}}^{(v)}_s(x),$$

$$\widehat{\mathbf{Q}} = \frac{1}{n}\sum_{i=1}^{n}\widehat{\mathbf{b}}_s(x_i)\widehat{\mathbf{b}}_s(x_i)'\widehat{\Psi}_{i,1}\widehat{\eta}^2_{i,1},$$

$$\widehat{\mathbf{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}\widehat{\mathbf{b}}_s(x_i)\widehat{\mathbf{b}}_s(x_i)'\widehat{\eta}^2_{i,1}\psi(y_i - \widehat{\eta}_{i,0})^2$$

with $\widehat{\Psi}_{i,1}$ a consistent estimator of $\Psi_{i,1} = \frac{\partial}{\partial\eta}\mathbb{E}[\psi(y_i;\eta)|x_i, \mathbf{w}_i]\big|_{\eta=\eta_{i,0}}$, $\widehat{\eta}_{i,v}$ a consistent estimator of $\eta_{i,v} = \eta^{(v)}(\mu_0(x_i) + \mathbf{w}_i'\boldsymbol{\gamma}_0)$ for $v = 0, 1$, and $\psi(u)$ 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 in `Stata`.

CCFF shows that $\widehat{T}_p(x) \to_d \mathsf{N}(0, 1)$ 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:

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

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

However, employing an IMSE-optimal binscatter (i.e., setting $J = J_{\texttt{IMSE}}$ 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; Cattaneo, Farrell, and Feng 2020; Calonico, Cattaneo, and Farrell 2021) to form valid confidence intervals based on IMSE-optimal binscatter, that is, without altering the partitioning scheme $\widehat{\Delta}$ used.

Our recommended implementation employs 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 $J = J_{\texttt{IMSE}}$, which gives an IMSE-optimal binscatter (point estimator). Then, employ the confidence interval $\widehat{I}_{p+q}(x)$ with $q \ge 1$, which gives a valid confidence interval: $\mathbb{P}\big[\vartheta^{(v)}_{\mathbf{w}}(x) \in \widehat{I}_{p+q}(x)\big] \to 1-\alpha$, for all x.

Main implementation details

The four estimation commands `binsreg`, `binslogit`, `binsprobit` and `binsqreg` implement confidence intervals, and report them as part of the final binned scatter plot. 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. The implementation is done over a grid of evaluations points, which can be modified via the option `cigrid()`, and the desired level is set by the option `level()`. Notice 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. See Section 3 for the syntax and further discussion.

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/binning of size J, the $(1-\alpha)\%$ confidence band for $\vartheta^{(v)}_{\mathbf{w}}(x)$ is:

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

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,

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

and $\mathbf{N}_K \sim \mathsf{N}(\mathbf{0}, \mathbf{I})$ being a K-dimensional standard normal random vector. The distribution of $\sup_x \left|\widehat{T}_p(x)\right|$, which is unknown, is approximated by that of $\sup_x \left|\widehat{Z}_p(x)\right|$ conditional on the data $\mathbf{D}$, which can be simulated by taking repeated samples from $\mathbf{N}_K$ 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 $\mathbf{N}_{(p+1)J-(J-1)s}$, 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 $\vartheta^{(v)}_{\mathbf{w}}(x)$ $(1-\alpha)\%$ of the time in repeated sampling, whenever the misspecification error can be ignored. As before, we recommend employing robust bias correction to remove misspecification error introduced by binscatter, that is, following the same logic discussed above for the case of confidence intervals construction. To be more precise, first p is chosen, along with s and v, and the optimal partitioning/binning is selected according to $J = J_{\texttt{IMSE}}$. Then, valid confidence bands are constructed using $\widehat{I}_{p+q}(x)$ with $q \geq 1$: $\mathbb{P}\big[\vartheta^{(v)}_{\mathbf{w}}(x) \in \widehat{I}_{p+q}(x),\ \text{for all } x\big] \to 1-\alpha$.

Main implementation details

The four estimation commands `binsreg`, `binslogit`, `binsprobit` and `binsqreg` implement confidence bands, and report them as part of the final binned scatter plot. 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. The implementations is done over a grid of evaluations 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, as well as many other practically relevant possibilities. See Section 3 for the syntax discussion and other implementation details.

2.5 Parametric Specification Testing

In addition to implementing binscatter and producing binned scatter plots, with both point and uncertainty estimators, the package Binsreg also allows for formal testing of substantive hypotheses. The command `binstest` implements all hypothesis tests available. This command can be used as a stand-alone command.

The command `binstest` implements two types of substantive hypothesis tests about $\vartheta^{(v)}_{\mathbf{w}}(x)$: (i) parametric specification testing, and (ii) nonparametric shape restriction testing. This subsection discusses the first type of hypothesis testing, while the next subsection discusses the second one.

For a choice of (p, s, v), and partitioning/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 $\vartheta_{\mathbf{w}}(x) = M_{\mathbf{w}}(x; \boldsymbol{\theta}, \boldsymbol{\gamma})$ when $M_{\mathbf{w}}(x; \boldsymbol{\theta}, \boldsymbol{\gamma}) = m(x; \boldsymbol{\theta}) + \mathbf{w}'\boldsymbol{\gamma}$ 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 hypothesis are, respectively,

$$\ddot{\mathsf{H}}_0 : \quad \sup_x \left| \vartheta^{(v)}_{\mathbf{w}}(x) - M^{(v)}_{\mathbf{w}}(x; \boldsymbol{\theta}, \boldsymbol{\gamma}) \right| = 0, \qquad \text{for some } \boldsymbol{\theta} \text{ and } \boldsymbol{\gamma}, \qquad vs.$$

$$\ddot{\mathsf{H}}_{\mathrm{A}} : \quad \sup_x \left| \vartheta^{(v)}_{\mathbf{w}}(x) - M^{(v)}_{\mathbf{w}}(x; \boldsymbol{\theta}, \boldsymbol{\gamma}) \right| > 0, \qquad \text{for all } \boldsymbol{\theta} \text{ and } \boldsymbol{\gamma},$$

for a choice of v, and for $M^{(v)}_{\mathbf{w}}(x; \boldsymbol{\theta}, \boldsymbol{\gamma}) = \partial^v M_{\mathbf{w}}(x; \boldsymbol{\theta}, \boldsymbol{\gamma}) / \partial x^v$ with $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^{d_\theta}$.

For example, excluding additional covariates $\mathbf{w}_i$ for simplicity, $\widehat{\mu}(x)$ is compared to $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ in order 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 non-linear 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.

Following CCFF, the command `binstest` employs the test statistic

$$\ddot{T}_p(x) = \frac{\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x) - M^{(v)}_{\widehat{\mathbf{w}}}(x; \widetilde{\boldsymbol{\theta}}, \widetilde{\gamma})}{\sqrt{\widehat{\Omega}(x)/n}}, \qquad 0 \leq v, s \leq p,$$

where $(\widetilde{\boldsymbol{\theta}}', \widetilde{\boldsymbol{\gamma}}')'$ are consistently estimates of $(\boldsymbol{\theta}', \boldsymbol{\gamma}')'$ under the null hypothesis (correct parametric specification), and are "well behaved" under the alternative hypothesis (parametric misspecification). Then, a parametric specification hypothesis testing procedure is:

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

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 Gaussian random vector, conditional on the data $\mathbf{D}$, as in the case of confidence bands already discussed. This testing procedure is an asymptotically valid α%-level test if the misspecification error is removed from the test statistic $\ddot{T}_p(x)$.

The command `binstest` employs robust bias correction by default: first p and s are chosen, and the partitioning/binning scheme is selected by setting $J = J_{\texttt{IMSE}}$ for these choices. Then, using this partitioning scheme, the testing procedure (5) is implemented with the choice $p+q$ instead of p, with $q \geq 1$. CCFF shows that, under regularity conditions, the resulting parametric specification testing approach controls Type I error with non-trivial power: for given p, $0 \leq v, s \leq p$, and $J = J_{\texttt{IMSE}}$,

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

and

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

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 parametric fit considered entirely, then such parametric fit is incompatible with the data, i.e., should be rejected.

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 `polyreg(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 parametrizations of $m(x,\boldsymbol{\theta})$, the command takes as input an auxiliary array/database (`dta` in Stata, or data frame in `R`) via the option `testmodelparfit(`*`filename`*`)` containing the following columns/variables: grid of evaluation points in one column, and fitted values $m(x,\widehat{\boldsymbol{\theta}})$ (over the evaluation grid) for each parametric model considered in other columns/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/binning scheme selected. See Section 3 for other options and more details on the syntax and implementation.

2.6 Nonparametric Shape Testing

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

$$\dot{\mathsf{H}}_0: \quad \sup_x \vartheta^{(v)}_{\mathbf{w}}(x) \leq 0, \qquad vs. \qquad \dot{\mathsf{H}}_{\mathrm{A}}: \quad \sup_x \vartheta^{(v)}_{\mathbf{w}}(x) > 0,$$

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

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

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

Then, the testing procedure is:

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

with $\mathfrak{c} = \inf\{c \in \mathbb{R}_+ : \mathbb{P}[\sup_x \widehat{Z}_p(x) \leq c \mid \mathbf{D}] \geq 1 - \alpha\}$. As before, misspecification errors of binscatter need to be taken into account in order to control Type I error. As in previous cases, CCFF shows that for given p, $0 \leq v, s \leq p$, and $J = J_{\mathtt{IMSE}}$ accordingly, then

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

and

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

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

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: $\dot{\mathsf{H}}_0 : \ \sup_x \vartheta^{(v)}_{\mathbf{w}}(x) \leq \mathtt{a}$. Option `testshaper(a)` for one-sided to the right: $\dot{\mathsf{H}}_0 : \ \inf_x \vartheta^{(v)}_{\mathbf{w}}(x) \geq \mathtt{a}$. Option `testshape2(a)` for two-sided testing: $\dot{\mathsf{H}}_0 : \sup_x |\vartheta^{(v)}_{\mathbf{w}}(x) - \mathtt{a}| = 0$. 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/binning scheme. See Section 3 for more details on the syntax and implementation.

2.7 Multi-Sample 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 is 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 $\ell = 0, 1, 2, \ldots, L$.

For example, assuming that two sub-samples 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:

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

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

$$\text{Reject } \mathsf{H}_0^{\Delta} \qquad \text{if and only if} \qquad \sup_x T_p^{\Delta}(x) \ge \mathfrak{c}, \tag{7}$$

with the critical value obtained as before via Gaussian strong approximations. As discussed before, in practice the robust bias-corrected test statistics $T^{\Delta}_{p+q}(x)$ 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 multi-samples. 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 provides all the necessary theoretical background.

Main implementation details

Estimation and uncertainty quantification across subsample 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 H_0^{Δ}. See Section 3 for more details on the syntax and implementation.

2.8 Extensions and Other Implementation Details

The package Binsreg is implemented using the base commands in `Stata`. For example, `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 bootstrapping-based standard error is selected). Furthermore, the testing commands (`binstest` and `binspwc`) also employ base commands in `Stata` whenever possible. (A similar approach is used in `Python` and `R`, based on the available base functions/commands therein.) This approach may sacrifice some speed of implementation, but improves substantially in terms of stability and replicability. Importantly, essentially most options available in the `Stata` 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 and/or robustness of the package.

Other metrics

All the results presented above employ the uniform norm, that is, focus on the the largest deviation on the support of a function. See, for example, $\dot{\mathsf{H}}_0$, $\ddot{\mathsf{H}}_0$ and H_0^{Δ}. Our results also apply to other metrics, such as the $L_{\mathfrak{p}}$ metric. In such case, the null hypotheses, the 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

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

where $\mathfrak{p}$ is some integer (typically $\mathfrak{p} = 2$ for squared deviations), and the corresponding critical value simulation takes the form $\mathfrak{c} = \inf\{c \in \mathbb{R}_+ : \mathbb{P}[\int |\widehat{Z}_p(x)|^{\mathfrak{p}} dx \le c \mid \mathbf{D}] \ge 1-\alpha\}$. Analogous modifications are done for other hypothesis tests. The choice of metric is implemented in each testing command via the option `lp()`.

Mass points and minimum effective sample size

All seven commands in the package Binsreg incorporate 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, and this 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`) only allows 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 recall that 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 employed 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, if 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)` and $N_1 = 20$ by default, then the data are deemed appropriate for ROT selection of J via the command `binsregselect`, and hence $\widehat{J}_{\texttt{ROT}}$ is implemented. If, in addition, $N > N_2 + (p+1)\widehat{J}_{\texttt{ROT}} - (\widehat{J}_{\texttt{ROT}} - 1)s$, then $\widehat{J}_{\texttt{DPI}}$ is also implemented whenever requested. Furthermore, the command `binsregselect` employs the following alternative formula for J selection:

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

with a slightly different constant $\mathscr{V}_n(p,s,v)$, taking into account 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/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, and hence the nonparametric asymptotic 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(`$\texttt{N}_1$ $\texttt{N}_2$`)`. Second, the option `masspoints()` controls how the package Binsreg handles the presence of mass points (i.e., 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.

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 three 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 employs the following alternative formula for J selection:

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

with 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.

Minimum effective sample size

The package Binsreg requires some minimal variation in x_i in order 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 three 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.

3 Syntax and Implementation Details

The package Binsreg includes seven commands: (i) `binsreg`, `binslogit`, `binsprobit` and `binsqreg` for point estimation, uncertainty quantification and binned scatter plotting; (ii) `binstest` and `binspwc` for hypothesis testing and statistical inference; and (iii) `binsregselect` for data-driven IMSE-optimal construction of the binning/partitioning scheme. We first introduce the general syntax for each of these commands, and then we discuss some of their common and distinct features. We group the commands in three categories: estimation, testing, and binning selection.

All seven commands employ the following inputs:

depvar is the dependent variable (y_i).

indvar is the independent variable (x_i).

othercovs is a varlist for covariate adjustment ($\mathbf{w}_i$).

p, s and v are integers satisfying $0 \leq s, v \leq p$.

weights allow for `fweights`, `aweights` and `pweights`; see `weights` in Stata for more details. (In `R`, *weights* allows for the equivalent of `fweights` only; see `lm()` help for more details.)

3.1 Estimation Commands

`binsreg` syntax

The command `binsreg` implements least squares binscatter estimators, accompanying confidence intervals and confidence bands, and also a global polynomial approximation for completeness. It also implements binned scatter plots. This command employs the base linear regression command (`regress`) whenever possible. A partitioning/binning structure is required but, if not provided, then one is selected in a data-driven way using the companion command `binsregselect`.

`binsreg` *depvar* *indvar* [*othercovs*] [*if*] [*in*] [*weight*] [`,`

`deriv(`*v*`)` `at(`*position*`)`

`absorb(`*absvars*`)` `reghdfeopt(`*reghdfe_option*`)`

`dots(`*p s*`)` `dotsgrid(`*dotsgridopt*`)` `dotsplotopt(`*string*`)`

`line(`*p s*`)` `linegrid(`*numeric*`)` `lineplotopt(`*string*`)`

`ci(`*p s*`)` `cigrid(`*cigridopt*`)` `ciplotopt(`*string*`)`

`cb(`*p s*`)` `cbgrid(`*numeric*`)` `cbplotopt(`*string*`)`

`polyreg(`*p*`)` `polyreggrid(`*numeric*`)` `polyregcigrid(`*numlist*`)`

`polyregplotopt(`*string*`)`

`by(`*varname*`)` `bycolors(`*colorstylelist*`)` `bysymbols(`*symbolstylelist*`)`

`bylpatterns(`*linepatternstylelist*`)`

`nbins(`*J*`)` `binspos(`*numlist*`)` `binsmethod(`*string*`)`

`nbinsrot(`*numeric*`)` `samebinsby` `randcut(`*numeric*`)`

`nsims(`*S*`)` `simsgrid(`*numeric*`)` `simsseed(`*num*`)`

`dfcheck(`*n1 n2*`)` `masspoints(`*string*`)`

`vce(`*vcetype*`)` `asyvar(`*on/off*`)`

`level(`*numeric*`)` `usegtools(`*on/off*`)`

`noplot` `savedata(`*filename*`)` `replace`

`plotxrange(min max)` `plotyrange(min max)` *twoway_options*]

Most of the options above are common to all estimation commands in the package Binsreg, and hence they are discussed after we present the syntax of all those commands. Important exceptions are `absorb(`*absvars*`)` and `reghdfeopt(`*reghdfe_option*`)`, which implement multi-way fixed effect and multi-way clustering least squares binscatter estimators via the module `reghdfe` (Correia and Constantine 2021). These options are only available for `binsreg` and `binsregselect` (and `binstest` and `binspwc` if testing procedures are based on least squares binscatter). The command `binsreg` relies on the base command `regress`, or the command `reghdfe` when specified by the user, and therefore it uses the `regress` or `reghdfe` syntax whenever possible.

`binslogit/binsprobit` syntax

The commands `binslogit` and `binsprobit` implement binary outcome binscatter estimators, accompanying confidence intervals and confidence bands, and also a global polynomial approximation for completeness. They also implement binned scatter plots. These commands employ the base commands `logit` and `probit` whenever possible (or the function `glm()` in `R`). A partitioning/binning structure is required but, if not provided, then one is selected in a data-driven way using the companion command `binsregselect`.

`binslogit/binsprobit` *depvar indvar* [*othercovs*] [*if*] [*in*] [*weight*] [`,`

`deriv(`*v*`)` `at(`*position*`)`

`nolink`

`dots(`*p s*`)` `dotsgrid(`*dotsgridopt*`)` `dotsplotopt(`*string*`)`

`line(`*p s*`)` `linegrid(`*numeric*`)` `lineplotopt(`*string*`)`

`ci(`*p s*`)` `cigrid(`*cigridopt*`)` `ciplotopt(`*string*`)`

`cb(`*p s*`)` `cbgrid(`*numeric*`)` `cbplotopt(`*string*`)`

`polyreg(`*p*`)` `polyreggrid(`*numeric*`)` `polyregcigrid(`*numlist*`)`
`polyregplotopt(`*string*`)`

`by(`*varname*`)` `bycolors(`*colorstylelist*`)` `bysymbols(`*symbolstylelist*`)`
`bylpatterns(`*linepatternstylelist*`)`

`nbins(`*J*`)` `binspos(`*numlist*`)` `binsmethod(`*string*`)`
`nbinsrot(`*numeric*`)` `samebinsby` `randcut(`*numeric*`)`

`nsims(`*S*`)` `simsgrid(`*numeric*`)` `simsseed(`*num*`)`

`dfcheck(`*n1 n2*`)` `masspoints(`*string*`)`

`vce(`*vcetype*`)` `asyvar(`*on/off*`)`

`level(`*numeric*`)` `usegtools(`*on/off*`)`

`noplot` `savedata(`*filename*`)` `replace`
`plotxrange(min max)` `plotyrange(min max)` *twoway_options*]

When compared to `binsreg`, the only new option is `nolink`, which reports the prediction for the linear single-index (i.e., without using the inverse link function $\eta(\cdot)$). The options `absorb(`*absvars*`)` and `reghdfeopt(`*reghdfe_option*`)` are no longer available. All other options are common to all estimation commands in the package Binsreg, and hence they are discussed below. The commands `binslogit` and `binsprobit` implement the base commands `logit` and `probit`, and thus the same syntax is used whenever possible.

`binsqreg` syntax

The command `binsqreg` implements quantile regression binscatter estimators, accompanying confidence intervals and confidence bands, and also a global polynomial approximation for completeness. It also implements binned scatter plots. This command employs the base quantile regression command (`qreg` or `bsqreg`) whenever possible (or the function `rq()` in `R`). A partitioning/binning structure is required but, if not provided, then one is selected in a data-driven way using the companion command `binsregselect`.

`binsqreg` *depvar indvar* [*othercovs*] [*if*] [*in*] [*weight*] [,

`deriv(`*v*`)` `at(`*position*`)`

`quantile(`*numeric*`)`

`dots(`*p s*`)` `dotsgrid(`*dotsgridopt*`)` `dotsplotopt(`*string*`)`

`line(`*p s*`)` `linegrid(`*numeric*`)` `lineplotopt(`*string*`)`

`ci(`*p s*`)` `cigrid(`*cigridopt*`)` `ciplotopt(`*string*`)`

`cb(`*p s*`)` `cbgrid(`*numeric*`)` `cbplotopt(`*string*`)`

`polyreg(`*p*`)` `polyreggrid(`*numeric*`)` `polyregcigrid(`*numlist*`)`

`polyregplotopt(`*string*`)`

`by(`*varname*`)` `bycolors(`*colorstylelist*`)` `bysymbols(`*symbolstylelist*`)`

`bylpatterns(`*linepatternstylelist*`)`

`nbins(`*J*`)` `binspos(`*numlist*`)` `binsmethod(`*string*`)`

`nbinsrot(`*numeric*`)` `samebinsby` `randcut(`*numeric*`)`

`nsims(`*S*`)` `simsgrid(`*numeric*`)` `simsseed(`*num*`)`

`dfcheck(`*n1 n2*`)` `masspoints(`*string*`)`

`vce(`*vcetype*`)` `asyvar(`*on/off*`)`

`level(`*numeric*`)` `usegtools(`*on/off*`)`

`noplot` `savedata(`*filename*`)` `replace`

`plotxrange(min max)` `plotyrange(min max)` *twoway_options*]

When compared to `binsreg`, the only new option is `quantile(`*numeric*`)`, which sets the quantile of interest, and the options `absorb(`*absvars*`)` and `reghdfeopt(`*reghdfe_option*`)` are no longer available. All other options are common to all estimation commands in the package Binsreg, and hence they are discussed below. The command `binsqreg` implements the base command `qreg` or `bsqreg` (if bootstrapping-based standard error is requested via `vce()`), and thus the same syntax and options are allowed for whenever possible/appropriate.

3.2 Estimation Commands Syntax

Estimand

`deriv(`v`)` specifies the derivative order for estimation of $\vartheta^{(s)}_{\mathbf{w}}(x)$, testing and plotting. The default is `deriv(0)`, which corresponds to the function itself, $\vartheta_{\mathbf{w}}(x)$.

`at(`*position*`)` specifies the value of the additional covariates $\mathbf{w}$ (if supplied) where the desired function $\vartheta^{(v)}_{\mathbf{w}}(x)$ is evaluated.

`absorb(`*absvars*`)` enables `reghdfe`, where *absvars* corresponds to the variables to be "absorbed". In this case, the option `vce(`*vcetype*`)` specifies the `vce` options (e.g., multi-way clustering) for `reghdfe`. All other options in `reghdfe` are specified via the option `reghdfeopt(`*reghdfe_option*`)`. This method may change the estimand, and is only available for least squares binscatter (`binsreg`).

`nolink` specifies the estimand to be the linear index $\mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0$, instead of the conditional probability $\eta(\mu_0(x) + \mathbf{w}'\boldsymbol{\gamma}_0)$. This option is only available for binary outcome regression binscatter (`binslogit`/`binsprobit`).

`quantile(`q`)` specifies the quantile q for estimation, testing and plotting. This option is only available for quantile regression binscatter (`binsqreg`).

Dots

`dots(`p s`)` sets a piecewise polynomial of degree p with s smoothness constraints when constructing $\widehat{\vartheta}^{(v)}_{\mathbf{w}}(x)$ for point estimation and plotting as "dots". The default is `dots(0 0)`, which corresponds to piecewise constant (canonical binscatter).

`dotsgrid(`*dotsgridopt*`)` specifies the number and location of dots within each bin to be plotted. Two options are available: *mean* and a *numeric* non-negative integer. The option `dotsgrid(mean)` adds the sample average of *indvar* within each bin to the grid of evaluation points for each bin. The option `dotsgrid(`*numeric*`)` adds *numeric* number of evenly-spaced points to the grid of evaluation points. Both options can be used simultaneously: for example, `dotsgrid(mean 5)` generates six evaluation points within each bin containing the sample mean of *indvar* within each bin and five evenly-spaced points. Given this choice, the dots are point estimates evaluated over the selected grid within each bin. The default is `dotsgrid(mean)`, which corresponds to one dot per bin evaluated at the sample average of *indvar* within each bin (canonical binscatter).

`dotsplotopt(`*string*`)` standard graphs options to be passed on to the `twoway` command to modify the appearance of the plotted dots.

Line

`line(`p s`)` sets a piecewise polynomial of degree p with s smoothness constraints when constructing $\widehat{\vartheta}^{(v)}(x)$ for point estimation and plotting as a "line". By default, the line is not included in the plot unless explicitly specified. Recommended specification is `line(3 3)`, which adds a cubic B-spline estimate of the regression function of interest to the binned scatter plot.

`linegrid(`*numeric*`)` specifies the number of evaluation points of an evenly-spaced grid within each bin used for evaluation of the point estimate set by the `line(p s)` option. The default is `linegrid(20)`, which corresponds to 20 evenly-spaced evaluation points within each bin for fitting/plotting the line.

`lineplotopt(`*string*`)` standard graphs options to be passed on to the `twoway` command to modify the appearance of the plotted line.

Confidence Intervals

`ci(`p s`)` specifies the piecewise polynomial of degree p with s smoothness constraints used for constructing confidence intervals $\widehat{I}_p(x) = [\ \widehat{\vartheta}^{(v)}_{\mathbf{w}}(x) \pm \Phi^{-1}(1-\alpha/2) \cdot \sqrt{\widehat{\Omega}(x)/n}\ ]$. By default, the confidence intervals are not included in the plot unless explicitly specified. Recommended specification is `ci(3 3)`, which adds confidence intervals based on a cubic B-spline estimate of the regression function of interest to the binned scatter plot.

`cigrid(`*numeric*`)` specifies the number and location of evaluation points in the grid used to construct the confidence intervals set by the `ci(p s)` option. Two options are available: *mean* and a *numeric* non-negative integer. The option `cigrid(mean)` adds the sample average of *indvar* within each bin to the grid of evaluation points for each bin. The option `cigrid(`*numeric*`)` adds *numeric* number of evenly-spaced points to the grid of evaluation points. Both options can be used simultaneously: for example, `cigrid(mean 5)` generates six evaluation points within each bin containing the sample mean of *indvar* within each bin and five evenly-spaced points. The default is `cigrid(mean)`, which corresponds to one evaluation point set at the sample average of *indvar* within each bin for confidence interval construction.

`ciplotopt(`*string*`)` standard graphs options to be passed on to the `twoway` command to modify the appearance of the confidence intervals.

Confidence Band

`cb(`p s`)` specifies the piecewise polynomial of degree p with s smoothness constraints used for constructing the confidence band $\widehat{I}_p(\cdot) = [\ \widehat{\vartheta}^{(v)}_{\mathbf{w}}(\cdot) \pm \mathfrak{c} \cdot \sqrt{\widehat{\Omega}(\cdot)/n}\ ]$. By default, the confidence band is not included in the plot unless explicitly specified. Recommended specification is `cb(3 3)`, which adds a confidence band based on a cubic B-spline estimate of the regression function of interest to the binned scatter plot.

`cbgrid(`*numeric*`)` specifies the number of evaluation points of an evenly-spaced grid within each bin used for evaluation of the point estimate set by the `cb(p s)` option.

The default is `cbgrid(20)`, which corresponds to 20 evenly-spaced evaluation points within each bin for confidence band construction.

`cbplotopt(`*string*`)` standard graphs options to be passed on to the `twoway` command to modify the appearance of the confidence band.

Global Polynomial Regression

`polyreg(`*p*`)` sets the degree P of a global polynomial regression model for plotting. By default, this fit is not included in the plot unless explicitly specified. Recommended specification is `polyreg(3)`, which adds a cubic polynomial fit of the regression function of interest to the binned scatter plot.

`polyreggrid(`*numeric*`)` specifies the number of evaluation points of an evenly-spaced grid within each bin used for evaluation of the point estimate set by the `polyreg(p)` option. The default is `polyreggrid(20)`, which corresponds to 20 evenly-spaced evaluation points within each bin for fitting/plotting.

`polyregcigrid(`*numeric*`)` specifies the number of evaluation points of an evenly-spaced grid within each bin used for constructing confidence intervals based on polynomial regression set by the `polyreg(p)` option. The default is `polyregcigrid(0)`, which corresponds to not plotting confidence intervals for the global polynomial regression approximation.

`polyregplotopt(`*string*`)` standard graphs options to be passed on to the `twoway` command to modify the appearance of the global polynomial regression fit.

Subgroup Analysis

`by(`*varname*`)` specifies the variable containing the group indicator to perform subgroup analysis; both numeric and string variables are supported. When `by(`*varname*`)` is specified, `binsreg`, `binslogit`, `binsprobit` or `binsqreg` implements estimation and inference for each subgroup separately, but produces a common binned scatter plot. By default, the binning structure is selected for each subgroup separately, but see the option `samebinsby` below for imposing a common binning structure across subgroups.

`bycolors(`*colorstylelist*`)` specifies an ordered list of colors for plotting each subgroup series defined by the option `by()`.

`bysymbols(`*symbolstylelist*`)` specifies an ordered list of symbols for plotting each subgroup series defined by the option `by()`.

`bylpatterns(`*linepatternstylelist*`)` specifies an ordered list of line patterns for plotting each subgroup series defined by the option `by()`.

Partitioning/Binning Selection

`nbins(`*J*`)` sets the number of bins J for partitioning/binning of *indvar*. If not specified, the number of bins is selected via the companion command `binsregselect` in a data-driven, optimal way whenever possible.

`binspos(`*numlist*`)` specifies the position of binning knots. The default is `binspos(qs)`, which corresponds to quantile-spaced binning (canonical binscatter). Other options are: `es` for evenly-spaced binning, or a *numlist* for manual specification of the positions of inner knots (which must be within the range of *indvar*).

`binsmethod(`*string*`)` specifies the method for data-driven selection of the number of bins via the companion command `binstest`. The default is `binsmethod(dpi)`, which corresponds to the IMSE-optimal direct plug-in rule $\widehat{J}_{\mathtt{DPI}}$. The other option is: `rot` for rule of thumb implementation, $\widehat{J}_{\mathtt{ROT}}$.

`nbinsrot(`*numeric*`)` specifies an initial number of bins value used to construct the DPI number of bins selector via the the companion command `binstest`. If not specified, the data-driven ROT selector is used instead.

`samebinsby` forces a common partitioning/binning structure across all subgroups specified by the option `by()`. The knots positions are selected according to the option `binspos()` and using the full sample. If `nbins()` is not specified, then the number of bins is selected via the companion command `binsregselect` and using the full sample.

`randcut(`*numeric*`)` specifies the upper bound on a uniformly distributed variable used to draw a subsample for bins selection (J). Only observations for which `runiform()`$\leq$ *numeric* are used for estimating the IMSE-optimal number of bins.

Simulation

`nsims(`*S*`)` specifies the number of random draws S for constructing confidence bands and hypothesis testing. The default is `nsims(500)`, which corresponds to 500 draws from a standard Gaussian random vector of size $[(p+1)\cdot J-(J-1)\cdot s]$.

`simsgrid(`*numeric*`)` specifies the number of evaluation points of an evenly-spaced grid within each bin used for evaluation of the supremum (infimum, or $L_{\mathtt{p}}$ metric) operation needed to construct confidence bands and hypothesis testing procedures. The default is `simsgrid(20)`, which corresponds to 20 evenly-spaced evaluation points within each bin for approximating the supremum (or infimum) operator.

`simsseed(`*numeric*`)` sets the seed for simulations.

Mass Points and Degrees of Freedom

`dfcheck(`*n1 n2*`)` sets cutoff values for minimum effective sample size checks, which take into account the number of unique values of *indvar* (i.e., adjusting for the number of mass points), number of clusters, and degrees of freedom of the different statistical models considered. Specifically, $N_1 = n1$ and $N_2 = n2$. The default is `dfcheck(20 30)`, as discussed above.

`masspoints(`*string*`)` specifies how mass points in *indvar* are handled. By default, all mass point and degrees of freedom checks are implemented. Available options:

noadjust	omits mass point checks and the corresponding effective sample size adjustments.
nolocalcheck	omits within-bin mass point and degrees of freedom checks.
off	sets `masspoints(noadjust)` and `masspoints(nolocalcheck)` simultaneously.
veryfew	forces the command to proceed as if *indvar* has only a few number of mass points (i.e., distinct values). In other words, forces the command to proceed as if the mass point and degrees of freedom checks were failed.

Other Options

`vce(`*vcetype*`)` specifies the *vcetype* for variance estimation used. The default is `vce(robust)`.

`asyvar(on/off)` specifies the method used to compute standard errors. If `asyvar(on)` is specified, the standard error of the nonparametric component is used and the uncertainty related to other control variables *othercovs* is omitted. Default is `asyvar(off)`, that is, the uncertainty related to *othercovs* is taken into account.

`level(`*numeric*`)` sets the nominal confidence level $(1 - \alpha)$ for confidence interval and confidence band estimation.

`noplot` omits binscatter plotting.

`savedata(`*filename*`)` specifies a *filename* for saving all data underlying the binscatter plot (and more).

`replace` overwrites the existing file when saving the graph data.

`plotxrange(min max)` specifies the range of the x-axis for plotting. Observations outside the range are dropped in the plot.

`plotyrange(min max)` specifies the range of the y-axis for plotting. Observations outside the range are dropped in the plot.

twoway_options any unrecognized options are appended to the end of the `twoway` command generating the binned scatter plot.

3.3 Testing Commands

`binstest` **syntax**

The main purpose of the command `binstest` is to conduct hypothesis testing of parametric specifications and nonparametric shape restrictions for $\vartheta_{\mathbf{w}}^{(v)}(x)$ using binscatter methods. A partitioning/binning structure is required but, if not provided, then one is selected in a data-driven way using the companion command `binsregselect`.

`binstest` *depvar indvar* [*othercovs*] [*if*] [*in*] [*weight*] [,
    `estmethod(`*cmdname*`)` `deriv(`*v*`)` `at(`*position*`)` `nolink`
    `absorb(`*absvars*`)` `reghdfeopt(`*reghdfe_option*`)`
    `testmodel(`*p s*`)` `testmodelparfit(`*filename*`)` `testmodelpoly(`*p*`)`
    `testshape(`*p s*`)` `testshapel(`*numlist*`)` `testshaper(`*numlist*`)`
        `testshape2(`*numlist*`)` `lp(`*metric*`)`
    `bins(`*p s*`)` `nbins(`*J*`)` `binspos(`*numlist*`)` `binsmethod(`*string*`)`
        `nbinsrot(`*numeric*`)` `randcut(`*numeric*`)`
    `nsims(`*S*`)` `simsgrid(`*numeric*`)` `simsseed(`*num*`)`
    `dfcheck(`*n1 n2*`)` `masspoints(`*string*`)`
    `vce(`*vcetype*`)` `asyvar(`*on/off*`)` `usegtools(`*on/off*`)`]

`binspwc` **syntax**

The main purpose of the command `binspwc` is to conduct pairwise comparisons across samples/groups of observations using binscatter methods. A partitioning/binning structure is required but, if not provided, then one is selected in a data-driven way using the companion command `binsregselect`.

`binspwc` *depvar indvar* [*othercovs*] [*if*] [*in*] [*weight*] , `by(`*varname*`)` [
    `estmethod(`*cmdname*`)` `deriv(`*v*`)` `at(`*position*`)` `nolink`
    `absorb(`*absvars*`)` `reghdfeopt(`*reghdfe_option*`)`
    `pwc(`*p s*`)` `testtype(`*type*`)` `lp(`*metric*`)`
    `bins(`*p s*`)` `bynbins(`*numlist*`)` `binspos(`*numlist*`)` `binsmethod(`*string*`)`
        `nbinsrot(`*numeric*`)` `samebinsby` `randcut(`*numeric*`)`
    `nsims(`*S*`)` `simsgrid(`*numeric*`)` `simsseed(`*num*`)`
    `dfcheck(`*n1 n2*`)` `masspoints(`*string*`)`
    `vce(`*vcetype*`)` `asyvar(`*on/off*`)` `usegtools(`*on/off*`)`]

3.4 Testing Commands Syntax

The two testing commands have several options already discussed for the estimation commands. Therefore, we focus exclusively on those options that are new.

Basic Setup

`estmethod(`*cmdname*`)` specifies the binscatter model. The default is `estmethod(reg)`, which corresponds to the binscatter least squares regression. Other options are: `estmethod(qreg` *quantile*`)` for binscatter quantile regression where *quantile* is the quantile to be estimated, `estmethod(logit)` for binscatter logistic regression and `estmethod(probit)` for binscatter probit regression.

`lp(`*metric*`)` specifies an $L_{\mathtt{p}}$ metric used for testing. The default is `lp(inf)`, which corresponds to the sup-norm. Other options are $L_{\mathtt{p}}$(*numeric*) metric for a positive integer *numeric*.

`by(`*varname*`)` specifies the variable *varname* containing the group indicator to perform subgroup analysis; both numeric and string variables are supported. Estimation is done for each subgroup separately, and then all pairwise comparison tests are implemented. By default, the binning structure is selected for each subgroup separately, but see the option `samebinsby` above for imposing a common binning structure across subgroups. This option is required by the command `binspwc`.

Parametric Model Specification Testing

`testmodel(`*p s*`)` sets a piecewise polynomial of degree p with s smoothness constraints for parametric model specification testing. The default is `testmodel(3 3)`, which corresponds to a cubic B-spline estimate of the regression function of interest for testing against the fitting from a parametric model specification.

`testmodelparfit(`*filename*`)` specifies a dataset which contains the evaluation grid and fitted values of the model(s) to be tested against. The file must have a variable with the same name as *indvar*, which contains a series of evaluation points at which the binscatter model and the parametric model of interest are compared with each other. Each parametric model is represented by a variable named as *binsreg_fit**, which must contain the fitted values at the corresponding evaluation points.

`testmodelpoly(`*p*`)` specifies the degree of a global polynomial model P to be tested against.

Nonparametric Shape Restriction Testing

`testshape(`*p s*`)` sets a piecewise polynomial of degree p with s smoothness constraints for nonparametric shape restriction testing. The default is `testmodel(3 3)`, which corresponds to a cubic B-spline estimate of the regression function of interest for one-sided or two-sided testing.

`testshapel(`*numlist*`)` specifies a *numlist* of null boundary values for hypothesis testing. Each number a in the *numlist* corresponds to one boundary of a one-sided hypothesis test to the left.

`testshaper(`*numlist*`)` specifies a *numlist* of null boundary values for hypothesis testing. Each number a in the *numlist* corresponds to one boundary of a one-sided hypothesis test to the right.

`testshape2(`*numlist*`)` specifies a *numlist* of null boundary values for hypothesis testing. Each number a in the *numlist* corresponds to one boundary of a two-sided hypothesis test.

Pairwise Comparison Testing

`pwc(`*p s*`)` sets a piecewise polynomial of degree p with s smoothness constraints for pairwise group comparison. The default is `pwc(3 3)`, which corresponds to a cubic B-spline estimate of the function of interest for each group.

`testtype(`*type*`)` specifies the type of pairwise comparison test. The default is `testtype(2)`, which corresponds to a two-sided test of the form $\mathsf{H}_0 : \vartheta^{(v)}_{\mathbf{w},0}(x) = \vartheta^{(v)}_{\mathbf{w},1}(x)$. Other options are: `testtype(l)` for the one-sided test of the form $\mathsf{H}_0 : \vartheta^{(v)}_{\mathbf{w},0}(x) \leq \vartheta^{(v)}_{\mathbf{w},1}(x)$ and `testtype(r)` for the one-sided test of the form $\mathsf{H}_0 : \vartheta^{(v)}_{\mathbf{w},0}(x) \geq \vartheta^{(v)}_{\mathbf{w},1}(x)$.

`bynbins(`*numlist*`)` sets a *numlist* of numbers of bins for partitioning/binning of *indvar*, which is applied to the binscatter estimation for each group. The ordering of the group follows the result of `tabulate`. If a single number of bins is specified, it applies to the estimation for all groups. If not specified, the number of bins is selected via the companion command `binsregselect` in a data-driven, optimal way whenever possible.

3.5 Binning Section

`binsregselect` syntax

The command `binsregselect` implements data-driven (IMSE-optimal) selection of partitioning/binning structure for binscatter. This command is used by the companion commands whenever the user does not specify the binning structure manually.

`binsregselect` *depvar indvar* [*othercovs*] [*if*] [*in*] [*weight*] [`, deriv(`*v*`)`
`absorb(`*absvars*`)` `reghdfeopt(`*reghdfe_option*`)`
`bins(`*p s*`)` `binspos(`*numlist*`)` `binsmethod(`*string*`)` `nbinsrot(`*numeric*`)`
`simsgrid(`*numeric*`)` `savegrid(`*filename*`)` `replace`
`dfcheck(`*n1 n2*`)` `masspoints(`*string*`)`
`vce(`*vcetype*`)` `usegtools(`*on/off*`)` `useeffn(`*numeric*`)` `randcut(`*numeric*`)`]

Most of the options for this command were already explained. The only new options are as follows.

Evaluation Points Grid Generation

`simsgrid(`*numeric*`)` specifies the number of evaluation points of an evenly-spaced grid within each bin used for evaluation of the supremum (infimum or $L_\mathtt{p}$ metric) operation needed to construct confidence bands and hypothesis testing procedures. The default is `simsgrid(20)`, which corresponds to 20 evenly-spaced evaluation points within each bin for approximating the supremum (infimum, or $L_\mathtt{p}$ metric) operator.

`savegrid(`*filename*`)` specifies a *filename* for storing the simulation grid of evaluation points. It contains the following variables: *indvar*, which is a sequence of evaluation points used in approximation; all control variables in *othercovs*, which take values of zero for prediction purpose; *binsreg_isknot*, indicating whether the grid is an inner knot; and *binsreg_bin*, indicating which bin an evaluation point belongs to.

`replace` overwrites the existing file when saving the grid.

Other Options

`useeffn(`*numeric*`)` specifies the effective sample size to be used when computing the (IMSE-optimal) number of bins. This option is useful for extrapolating the optimal number of bins to larger (or smaller) datasets than the one used to compute it.

3.6 Increasing Speed of Execution

The package Binsreg offers a large array of options and methods, many of which involve non-linear estimation and/or simulations, thereby slowing down its speed of execution. Furthermore, in order to improve the stability and replicability of the package, it implements several robustness checks that may further decrease execution speed, particularly in settings with ultra-large 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 employ several algorithms and procedures that require sorted data along the x dimension. If the provided data is 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 ultra-large datasets).

2. *Data Distribution.* The methods implemented in the package Binsreg were developed for continuously distributed data with "enough" variation (e.g., enough degrees of freedom within and across bins, appropriate rank conditions for Gram matrices, etc.). Because empirical work may involve data with mass points and/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 exhibits "enough" regularity.

3. *Number of Bins Selection.* The package Binsreg selects the number of bins J in a multi-step, 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(`*numeric*`)` 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` (as well as in `Python` and `R`). However, some of this algorithm (e.g., `pctile`) may be slow in large datasets.

   - Speed improvement: employ the community-distributed package gtools (Caceres 2020) via the option `gtools(on)`. This community-distributed 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-distributed package reghdfe (Correia and Constantine 2021) could be used when employing 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 to early exploration. Finally, for ultra-large 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.

4 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, and `t` is a binary covariate, and `id` is a group identifier. Summary statistics of the simulated data are as follows.

```
. use binsreg_simdata, clear
. sum
    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
```

4.1 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
Placement: Quantile-spaced
Derivative: 0
─────────────────────────────────────────────────────────────
# of observations                          1000
# of distinct values                       1000
# of clusters                                 .
─────────────────────────────────────────────────────────────
Bin selection:
         Degree of polynomial                 0
  # of smoothness constraints                 0
                     # of bins               21
─────────────────────────────────────────────────────────────

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

The main output is a binned scatter plot 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 model(s) employed.

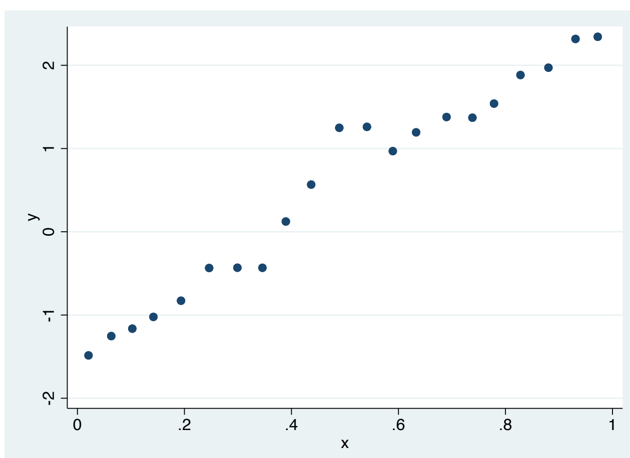


Figure 1: Canonical Binned Scatter Plot.

By default, the command `binsreg` evaluates and plots the regression function of interest $\vartheta_{\mathbf{w}}^{(v)}(x)$ at the mean of the additional covariates $\mathbf{w}_i$, i.e., $\mathbf{w} = \bar{\mathbf{w}}$. 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 file name in the option `at()`. For example,

```
. tempfile evalcovar
. preserve
. clear
. set obs 1
number of observations (_N) was 0, now 1
. gen w=0.2
. gen t=1
. save `evalcovar´, replace
(note: file /var/folders/85/ghqhbjfx4ysg9_s5gs0byqlw0000gn/T//S_00671.000001 not found)
file /var/folders/85/ghqhbjfx4ysg9_s5gs0byqlw0000gn/T//S_00671.000001 saved
. 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
Placement: Quantile-spaced
Derivative: 0
```

# of observations	1000
# of distinct values	1000
# of clusters	.
Bin selection:	
Degree of polynomial	0
# of smoothness constraints	0
# of bins	22

	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 save 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:

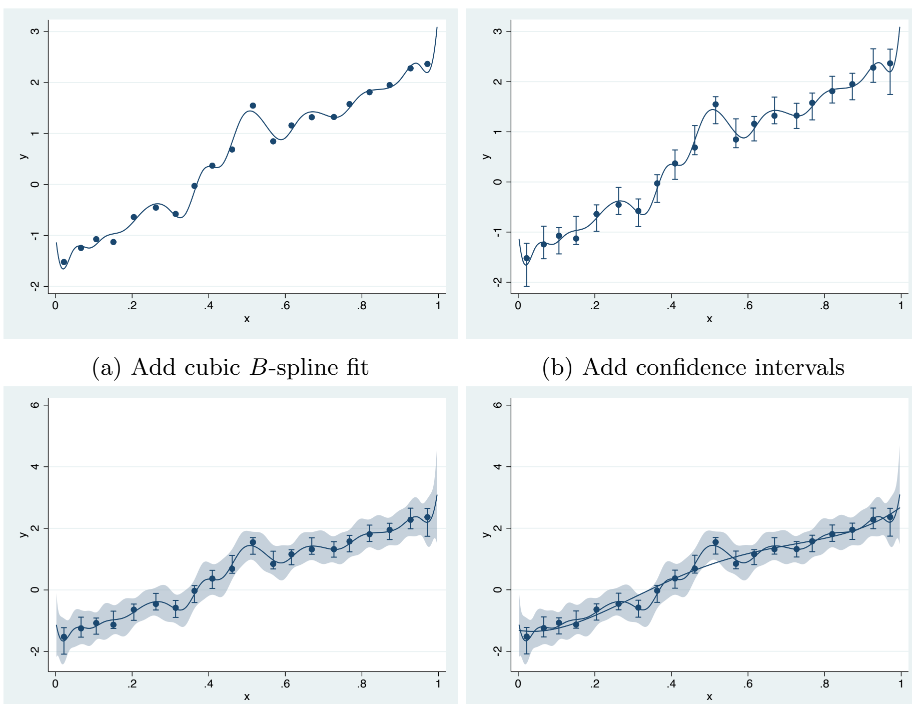
```
. binsreg y x w, nbins(20) polyreg(1)
Sorting dataset on x...
Note: This step is omitted if dataset already sorted by x.
Binscatter plot
Bin selection method: IMSE-optimal plug-in choice
```

```
Placement: Quantile-spaced
Derivative: 0
-----------------------------------------------------
# of observations              |         1000
# of distinct values           |         1000
# of clusters                  |            .
-----------------------------------------------------
Bin 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 scatter plot is not reported here to conserve space.

Figure 2: Binned Scatter Plot with Lines, Confidence Intervals and Bands.



(a) Add cubic B-spline fit

(b) Add confidence intervals

(c) Add confidence band

(d) Add a polynomial fit of degree 4

The command `binsreg` allows users to add a binscatter-based line approximating the unknown regression function, pointwise confidence intervals, a 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.

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

By construction, a cubic B-spline fit is a piecewise cubic polynomial function which 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(output/graphdat) replace ///
>                                    title("Binned Scatter Plot")
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
Placement: Quantile-spaced
Derivative: 0
Output file: output/graphdat.dta
```

# of observations	1000
# of distinct values	1000
# of clusters	500
Bin selection:	
Degree of polynomial	0
# of smoothness constraints	0
# of bins	20

	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 scatter plot. 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 scatter plot 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 scatter plot (Figure 3):

```
. binsreg y x w, by(t) dots(0,0) line(3,3) cb(3,3) ///
>                 bycolors(blue red) bysymbols(O T)
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
Placement: Quantile-spaced
Derivative: 0
Group: t = 0
──────────────────────────────────────────────
# of observations              │        485
# of distinct values           │        485
# of clusters                  │          .
───────────────────────────────┼──────────────
Bin selection:                 │
         Degree of polynomial  │          0
  # of smoothness constraints  │          0
                    # of bins  │         20
──────────────────────────────────────────────

────────────────────────────────────────
         │      p       s       df
─────────┼──────────────────────────────
 dots    │      0       0       20
 line    │      3       3       23
 CB      │      3       3       23
────────────────────────────────────────

Group: t = 1
──────────────────────────────────────────────
# of observations              │        515
# of distinct values           │        515
# of clusters                  │          .
───────────────────────────────┼──────────────
Bin selection:                 │
         Degree of polynomial  │          0
  # of smoothness constraints  │          0
                    # of bins  │         15
──────────────────────────────────────────────

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

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/partitioning structure will be different across subgroups in general. However, if the option `samebinsby` is specified, then a common

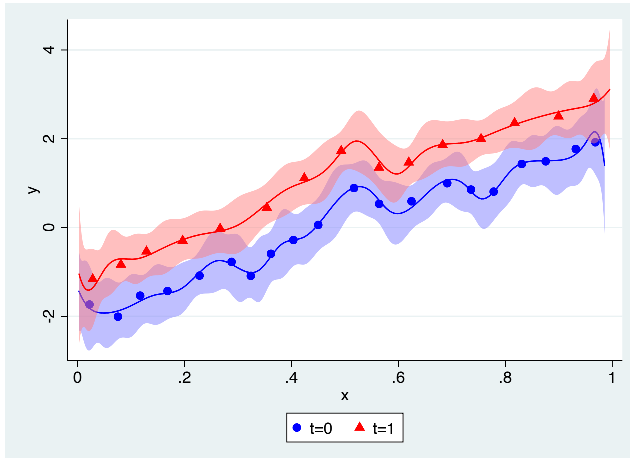


Figure 3: Binned Scatter Plot: Group Comparison

binning scheme for all subgroups is constructed based on the full sample.

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 module `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 module `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 25-th 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
Placement: Quantile-spaced
Derivative: 0

# of observations          |     1000
# of distinct values       |     1000
# of clusters              |        .

Bin selection:
        Degree of polynomial |        0
```

```
  # of smoothness constraints |        0
                    # of bins |       21
-----------------------------------------

-------------------------------------
         |    p        s       df
-------------------------------------
dots     |    0        0       21
-------------------------------------
```

By default, quantile regression methods employ an analytic variance estimator formula, which may not perform well in applications. A more robust alternative is employing 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
Placement: Quantile-spaced
Derivative: 0
-----------------------------------------
# of observations             |     1000
# of distinct values          |     1000
# of clusters                 |        .
-----------------------------------------
Bin selection:                |
         Degree of polynomial |        0
  # of smoothness constraints |        0
                    # of bins |       21
-----------------------------------------

-------------------------------------
         |    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 illustrated in Figure 4, which plots the conditional mean and its confidence band, together with the conditional 25-th and 75-th quantile regressions (i.e., conditional inter-quartile range).

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
Placement: Quantile-spaced
Derivative: 0
-----------------------------------------
# of observations             |     1000
# of distinct values          |     1000
# of clusters                 |        .
-----------------------------------------
Bin selection:                |
         Degree of polynomial |        0
  # of smoothness constraints |        0
                    # of bins |       12
-----------------------------------------
```

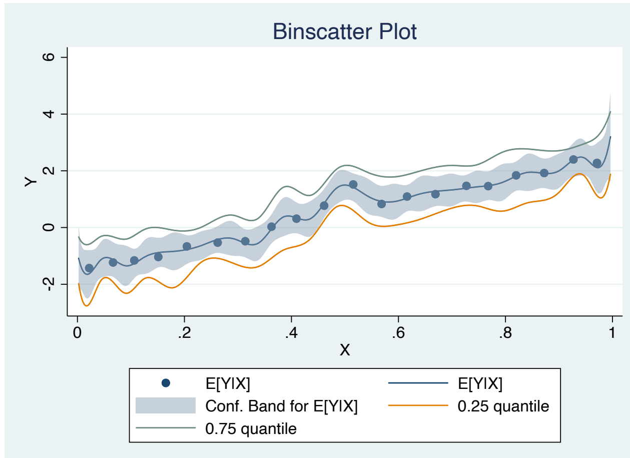


Figure 4: Binned Scatter Plot for Means and Quantiles.

	p	s	df
dots	0	0	12

4.2 Hypothesis Testing and Statistical Inference

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

```
. binstest y x w, testmodelpoly(1)
Hypothesis tests based on binscatter estimates
Estimation method: reg
Bin selection method: IMSE-optimal plug-in choice
Placement: Quantile-spaced
Derivative: 0
─────────────────────────────────────────────────
# of observations                  |     1000
# of distinct values               |     1000
# of clusters                      |        .
─────────────────────────────────────────────────
Bin selection:                     |
         Degree of polynomial      |        0
  # of smoothness constraints      |        0
                     # of bins     |       21
─────────────────────────────────────────────────

Model specification Tests:
Degree: 3     # of smoothness constraints: 3
─────────────────────────────────────────────────────
H0: mu =            |  sup |T|            p value
─────────────────────────────────────────────────────
poly. degree  1     |   6.503               0.000
─────────────────────────────────────────────────────
```

A test for linearity of the regression function $\mu_0(x)$ is implemented using the binscatter estimator. By default, a cubic B-spline is employed 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.

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.

```
. qui binsregselect y x w, simsgrid(30) savegrid(output/parfitval) replace
. qui reg y x w
. use output/parfitval, clear
. predict binsreg_fit_lm
(option xb assumed; fitted values)
. save output/parfitval, replace
file output/parfitval.dta saved
. use binsreg_simdata, clear
. binstest y x w, testmodelparfit(output/parfitval) lp(2)
Hypothesis tests based on binscatter estimates
Estimation method: reg
Bin selection method: IMSE-optimal plug-in choice
Placement: Quantile-spaced
Derivative: 0
─────────────────────────────────────────────
# of observations                 │    1000
# of distinct values              │    1000
# of clusters                     │       .
─────────────────────────────────────────────
Bin selection:                    │
         Degree of polynomial     │       0
  # of smoothness constraints     │       0
                    # of bins     │      21
─────────────────────────────────────────────

Model specification Tests:
Degree: 3     # of smoothness constraints: 3
Input file: output/parfitval.dta
─────────────────────────────────────────────────
H0: mu =           │  L2 of T          p value
─────────────────────────────────────────────────
   binsreg_fit_lm  │   1.760             0.000
─────────────────────────────────────────────────
```

The first line, `binsregselect y x w, simsgrid(30) savegrid(output/parfitval) replace`, generates the auxiliary file containing the grid of evaluation points. Since the parameter of interest is only the mean relation between `y` and `x`, i.e., $\mu_0(x)$, 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 as zeros. In addition, the variable containing fitted values has to follow a specific naming rule, i.e., takes 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)
Hypothesis tests based on binscatter estimates
Estimation method: reg
Bin selection method: IMSE-optimal plug-in choice
Placement: Quantile-spaced
Derivative: 1
─────────────────────────────────────────────────────────
# of observations                   │        1000
# of distinct values                │        1000
# of clusters                       │           .
─────────────────────────────────────────────────────────
Bin selection:                      │
         Degree of polynomial       │           0
  # of smoothness constraints       │           0
                    # of bins       │          20
─────────────────────────────────────────────────────────

Shape Restriction Tests:
Degree: 3     # of smoothness constraints: 3
─────────────────────────────────────────────────────────
H0: inf mu >=       │  inf T            p value
─────────────────────────────────────────────────────────
         0          │  -2.680             0.194
─────────────────────────────────────────────────────────
```

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 simultaneously (given the derivative of interest). For example,

```
. binstest y x w, nbins(20) testshaper(-2 0) testshapel(4) testmodelpoly(1) ///
>                    nsims(1000) simsgrid(30)
Hypothesis tests based on binscatter estimates
Estimation method: reg
Bin selection method: IMSE-optimal plug-in choice
Placement: Quantile-spaced
Derivative: 0
─────────────────────────────────────────────────────────
# of observations                   │        1000
# of distinct values                │        1000
# of clusters                       │           .
─────────────────────────────────────────────────────────
Bin selection:                      │
         Degree of polynomial       │           0
  # of smoothness constraints       │           0
```

```
                   # of bins |      20
______________________________________________
Shape Restriction Tests:
Degree: 3     # of smoothness constraints: 3
_________________________________________________
H0: sup mu <=      | sup T              p value
_________________________________________________
         4         |  -1.660              1.000
_________________________________________________

_________________________________________________
H0: inf mu >=      | inf T              p value
_________________________________________________
        -2         |   1.514              1.000
         0         |  -9.622              0.000
_________________________________________________
Model specification Tests:
Degree: 3     # of smoothness constraints: 3
_________________________________________________
H0: mu =           | sup |T|            p value
_________________________________________________
poly. degree  1    |   6.108              0.000
_________________________________________________
```

The above syntax tests three shape restrictions and one model specification (linearity), employing 1000 random draws from $\mathbf{N}_K$ and 30 evaluation points to evaluate the supremum/infimum in the simulation.

The accompanying replication files include other illustrations. For example:

- Testing whether the median regression function is linear:
  `.  binstest y x w, estmethod(qreg 0.5) testmodelpoly(1)`
- Testing whether the non-linear 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 testing command `binspwc`. Using least square binscatter for the two samples identified via the binary variable `t`, we have:

```
. binspwc y x w, by(t)
Pairwise group comparison based on binscatter estimates
Estimation method: reg
Group variable: t
Bin selection method: IMSE-optimal plug-in choice
Placement: Quantile-spaced
Derivative: 0
_____________________________________________
Bin selection:                  |
         Degree of polynomial   |      0
  # of smoothness constraints   |      0
_____________________________________________
Hypothesis test:                |
         Degree of polynomial   |      3
  # of smoothness constraints   |      3
_____________________________________________
Group 1 vs. Group 0
____________________________________________________
Group t=                        |     1     |    0
____________________________________________________
                                |           |
```

```
# of observations            |    515 |     485
# of distinct values         |    515 |     485
# of clusters                |      . |       .
# of bins                    |     15 |      20
-----------------------------------------------

diff = group 1 - group 0
---------------------------------------------
HO:                |  sup |T|      p value
---------------------------------------------
diff=0             |    5.944        0.000
---------------------------------------------
```

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

```
. binspwc y x w, by(t) estmethod(qreg 0.4)
Pairwise group comparison based on binscatter estimates
Estimation method: qreg
Group variable: t
Bin selection method: IMSE-optimal plug-in choice
Placement: Quantile-spaced
Derivative: 0
---------------------------------------------
Bin selection:                 |
        Degree of polynomial   |     0
  # of smoothness constraints  |     0
---------------------------------------------
Hypothesis test:               |
        Degree of polynomial   |     3
  # of smoothness constraints  |     3
---------------------------------------------

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

diff = group 1 - group 0
---------------------------------------------
HO:                |  sup |T|      p value
---------------------------------------------
diff=0             |    5.157        0.000
---------------------------------------------
```

4.3 Binning Section

As already mentioned, the commands **binsreg** and **binstest** rely on data-driven bin selection procedures via the command **binsregselect** whenver the option **nbins()** is not employed by the user. Its basic syntax is as follows:

```
. binsregselect y x w
Bin selection for binscatter estimates
Method: IMSE-optimal: plug-in choice
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	.	.

The following choices of number of bins are reported: `ROT-POLY`, the rule-of-thumb (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 direct plug-in (DPI) choice; and `DPI-UKNOT`, the DPI choice with unique knots.

The direct plug-in choice is implemented based on the rule-of-thumb 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
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	.	.

Notice that in the example above an even-spaced, rather than quantile-spaced, binning scheme is selected via the option `binspos(es)`. The binning used in the commands `binsreg` and `binstest` may be adjusted similarly.

In addition, as illustrated above, the command `binsregselect` also 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(output/parfitval) replace
Bin selection for binscatter estimates
```

```
Method: IMSE-optimal: plug-in choice
Position: Quantile-spaced
Output file: output/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	.	.

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 zero on purpose (this is used to generate the fitting 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 integrated mean squared error (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
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	.	.

In this example 485 observations with $\mathtt{t} == \mathtt{0}$ 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 sub-sample of the observations. For example, the following command selects J using roughly 30% of the observations.

```
. binsregselect y x w, randcut(0.3)
Bin selection for binscatter estimates
Method: IMSE-optimal: plug-in choice
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	21	21	4.244	1.233
ROT-REGUL	21	21	.	.
ROT-UKNOT	21	21	.	.
DPI	24	24	7.397	1.271
DPI-UKNOT	24	24	.	.

5 Conclusion

We introduced the `Stata` package Binsreg, which provides general-purpose software implementations of binscatter via seven commands: (i) `binsreg`, `binslogit`, `binsprobit`, `binsqreg` for point estimation, uncertainty quantification and binned scatter plotting in least squares, Logit, Probit and quantile regression settings; (ii) `binstest` for hypothesis testing and statistical inference for parametric specification and nonparametric shape restrictions, and `binspwc` for multi-group pairwise statistical comparisons; and (iii) `binsregselect` for binning scheme selection. All our methods allow for multi-sample comparisons, which is useful when studying treatment effects heterogeneity in randomized and observational studies. Companion `Python` and `R` packages with similar syntax and capabilities are also available.

6 Acknowledgments

We thank Michael Droste, John Friedman, Andreas Fuster, Paul Goldsmith-Pinkham, David Lucca, Xinwei Ma, Jonah Rockoff, Jesse Rothstein, Ryan Santos, Jesse Shapiro and Rocio Titiunik for helpful comments and discussions.

Cattaneo gratefully acknowledges financial support from the National Science Foundation through grants SES-1947805 and SES-2019432.

7 References

Angrist, J., V. Chernozhukov, and I. Fernández-Val. 2006. Quantile regression under misspecification, with an application to the US wage structure. *Econometrica* 74(2): 539–563.

Angrist, J. D., and J. S. Pischke. 2008. *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press.

Caceres, M. 2020. `Stata` Module `Mauricio` 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(522): 767–779.

———. 2021. Coverage Error Optimal Confidence Intervals for Local Polynomial Regression. arXiv:1808.01398 .

Calonico, S., M. D. Cattaneo, and R. Titiunik. 2014. Robust Nonparametric Confidence Intervals for Regression-Discontinuity Designs. *Econometrica* 82(6): 2295–2326.

Cattaneo, M. D., R. K. Crump, M. H. Farrell, and Y. Feng. 2021. On Binscatter. arXiv:1902.09608 .

Cattaneo, M. D., M. H. Farrell, and Y. Feng. 2020. Large sample properties of partitioning-based series estimators. *Annals of Statistics* 48(3): 1718–1741.

Chetty, R., J. N. Friedman, N. Hilger, E. Saez, D. W. Schanzenbach, and D. Yagan. 2011a. How Does Your Kindergarten Classroom Affect Your Earnings? Evidence from Project STAR. *Quarterly Journal of Economics* 126(4): 1593–1660.

Chetty, R., J. N. Friedman, T. Olsen, and L. Pistaferri. 2011b. Adjustment Costs, Firm Responses, and Micro vs. Macro Labor Supply Elasticities: Evidence from Danish Tax Records. *Quarterly Journal of Economics* 126(2): 749–804.

Chetty, R., J. N. Friedman, and J. Rockoff. 2014. Measuring the Impacts of Teachers II: Teacher Value-Added and Student Outcomes in Adulthood. *American Economic Review* 104(9): 2633–2679.

Chetty, R., A. Looney, and K. Kroft. 2009. Salience and Taxation: Theory and Evidence. *American Economic Review* 99(4): 1145–1177.

Chetty, R., and A. Szeidl. 2006. Marriage, Housing, and Portfolio Choice: A Test of Grossman-Laroque. Working Paper, UC-Berkeley .

Correia, S., and N. Constantine. 2021. `Stata` Module `reghdfe` https://github.com/sergiocorreia/reghdfe.

Droste, M. 2019. `Stata` Module `Binscatter2` https://github.com/mdroste/stata-binscatter2/.

Starr, E., and B. Goldfarb. 2020. Binned Scatterplots: A Simple Tool to Make Research Easier and Better. *Strategic Management Journal* 41(12): 2261–2274.

Stepner, M. 2017. `Stata` Module `Binscatter` https://github.com/michaelstepner/binscatter/.

About the Authors

Matias D. Cattaneo is a Professor of Operations Research and Financial Engineering at the University of Michigan.

Richard K. Crump is a Vice President and the Function Head of the Capital Markets Function at the Federal Reserve Bank of New York.

Max H. Farrell is an Associate Professor of Statistics and Econometrics at the University of Chicago Booth School of Business.

Yingjie Feng is an Assistant Professor of Economics at Tsinghua University.