Non-Parametric Regression and Trees

Econometría Aplicada y Ciencia de Datos - CIDE Fall 2026

Carlos Brito

Inspired by material from Francisco Cabrera:

`github.com/fcabrerahz/EconometricsDS`

Motivation

- Regression trees, or CART for Classification and Regression Trees.
  - They have reduced interpretability as **we do not observe a** β effect, but they are **useful to predict**.
  - Especially useful (for prediction) when there is a combination of continuous and discrete regressors.
- They are an **extension of econometric procedures**: e.g. a nonparametric regression using a large number of step functions. So we start with it.

Outline

- Non-parametric Regression
- Series and Spline Regression
- Regression Trees
- Bagging and Random Forest

Motivation: Why Go Beyond Linear Models?

So far we estimate $E[Y \mid X] = X'\beta$. But economic relationships are often nonlinear, and we rarely know their true shape.

Motivation: Why Go Beyond Linear Models?

So far we estimate $E[Y \mid X] = X'\beta$. But economic relationships are often nonlinear, and we rarely know their true shape.

Examples:

- **Wages and experience (Mincer):** earnings rise fast early in the career, then flatten. Is a quadratic in experience really the right shape?

Motivation: Why Go Beyond Linear Models?

So far we estimate $E[Y \mid X] = X'\beta$. But economic relationships are often nonlinear, and we rarely know their true shape.

Examples:

- **Wages and experience (Mincer):** earnings rise fast early in the career, then flatten. Is a quadratic in experience really the right shape?
- **Engel curves:** the share of spending on food falls with income, but not necessarily linearly

Motivation: Why Go Beyond Linear Models?

So far we estimate $E[Y \mid X] = X'\beta$. But economic relationships are often nonlinear, and we rarely know their true shape.

Examples:

- **Wages and experience (Mincer):** earnings rise fast early in the career, then flatten. Is a quadratic in experience really the right shape?
- **Engel curves:** the share of spending on food falls with income, but not necessarily linearly
- **Regression discontinuity:** to estimate the effect of a program at an eligibility cutoff, we fit flexible local regressions on each side

Motivation: Why Go Beyond Linear Models?

So far we estimate $E[Y \mid X] = X'\beta$. But economic relationships are often nonlinear, and we rarely know their true shape.

Examples:

- **Wages and experience (Mincer):** earnings rise fast early in the career, then flatten. Is a quadratic in experience really the right shape?
- **Engel curves:** the share of spending on food falls with income, but not necessarily linearly
- **Regression discontinuity:** to estimate the effect of a program at an eligibility cutoff, we fit flexible local regressions on each side

$\Rightarrow$ **Nonparametric regression** lets the data choose the shape of $m(x) = E[Y \mid X = x]$.

Non-Parametric Regression

- Nonparametric estimation of the conditional expectation function:

$$\mathbb{E}[Y \mid X = x] = m(x)$$

- An economic model restricts the form of $m(x)$ to a parametric function. But if $m(x)$ can take any nonlinear shape it is therefore **nonparametric**.

- **We will discuss nonparametric kernel smoothing estimators of $m(x)$.**

Non-Parametric Regression

Consider a single real-valued regressor X.

- The nonparametric regression model is:

$$Y = m(X) + e, \qquad \mathbb{E}[e \mid X] = 0, \qquad \mathbb{E}[e^2 \mid X] = \sigma^2(X).$$

Non-Parametric Regression

Consider a single real-valued regressor X.

- The nonparametric regression model is:

$$Y = m(X) + e, \qquad \mathbb{E}[e \mid X] = 0, \qquad \mathbb{E}[e^2 \mid X] = \sigma^2(X).$$

- We have n observations for the pair (Y, X).
- The goal is to estimate $m(x)$ either at a single point x or at a set of points.
  - For most of the theory, we focus on estimation at a single point x.

Non-Parametric Regression

Consider a single real-valued regressor X.

- The nonparametric regression model is:

$$Y = m(X) + e, \qquad \mathbb{E}[e \mid X] = 0, \qquad \mathbb{E}[e^2 \mid X] = \sigma^2(X).$$

- We have n observations for the pair (Y, X).
- The goal is to estimate $m(x)$ either at a single point x or at a set of points.
  - For most of the theory, we focus on estimation at a single point x.
- In addition to the Gauss–Markov assumptions, we assume that both $m(x)$ and $f(x)$ (the marginal density of X) are continuous in x.

Binned Means Estimator

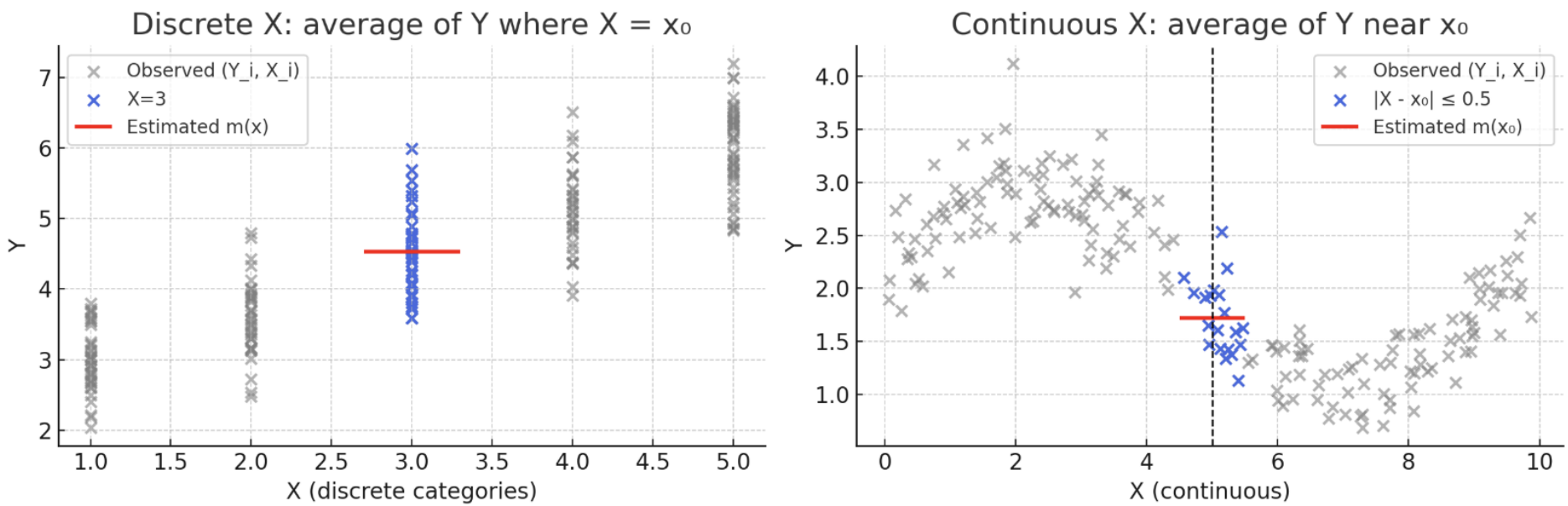
Fix the point x and consider estimation of $m(x)$:

- We get the mean of Y for random pairs (Y, X) such that $X = x$.
  - X discrete: we take the average of the sub-sample of observations Y_i with $X_i = x$.
  - X continuous: we take the average of the observations for which X_i is close to x. Say: $|X_i - x| \leq h$ for some small $h \geq 0$. h is the bandwidth.

Binned Means Estimator

Fix the point x and consider estimation of $m(x)$:

- We get the mean of Y for random pairs (Y, X) such that $X = x$.
  - X discrete: we take the average of the sub-sample of observations Y_i with $X_i = x$.
  - X continuous: we take the average of the observations for which X_i is close to x. Say: $|X_i - x| \leq h$ for some small $h \geq 0$. h is the bandwidth.



Notes: Discrete X: we compute $m(x) = \mathbb{E}[Y \mid X = x]$ by taking the mean of Y for all observations where $X_i = x$. For continuous X we estimate $m(x)$ using the average of Y values for observations where X_i is close to x (within a chosen bandwidth).

Binned Means Estimator

The **nonparametric estimator** of $m(x)$ using a local (uniform) window of half-width h is

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} 1\{|X_i - x| \leq h\} Y_i}{\sum_{i=1}^{n} 1\{|X_i - x| \leq h\}}$$

- **Numerator:** sum of Y_i for observations inside the window $[x - h, x + h]$.
- **Denominator:** number of observations inside the window.
- So $\hat{m}(x)$ is simply the **average of Y among observations close to x**.

Binned Means Estimator

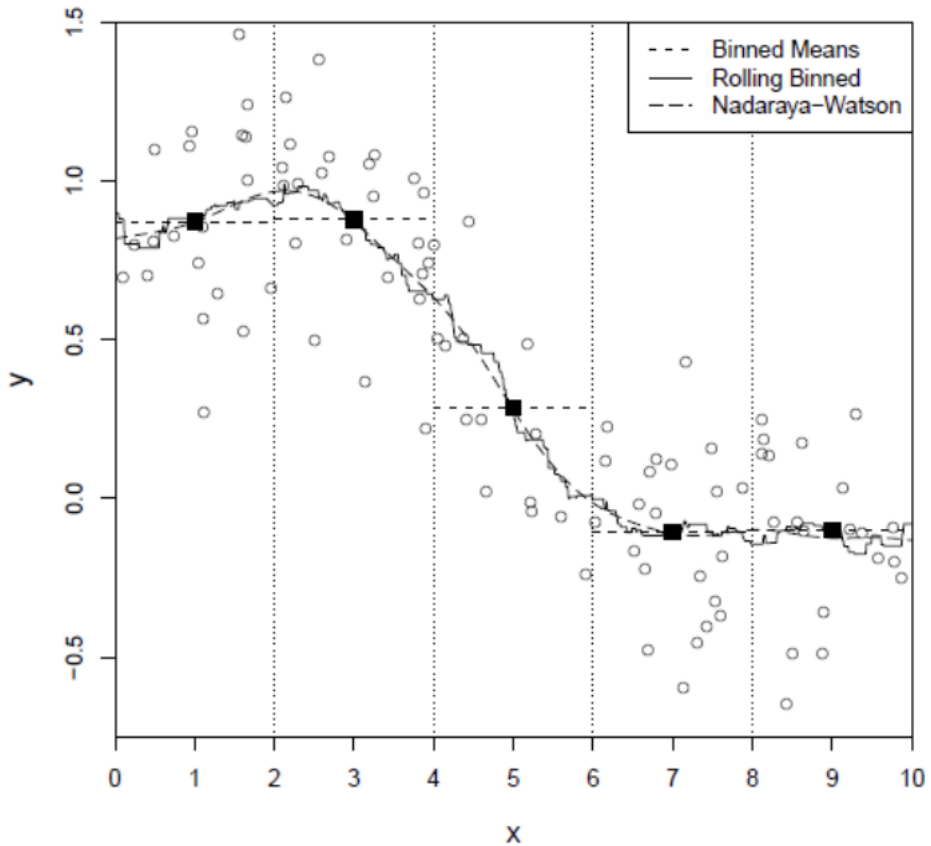
The **nonparametric estimator** of $m(x)$ using a local (uniform) window of half-width h is

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} 1\{|X_i - x| \leq h\} Y_i}{\sum_{i=1}^{n} 1\{|X_i - x| \leq h\}}$$

- **Numerator:** sum of Y_i for observations inside the window $[x - h, x + h]$.
- **Denominator:** number of observations inside the window.
- So $\hat{m}(x)$ is simply the **average of Y among observations close to x**.

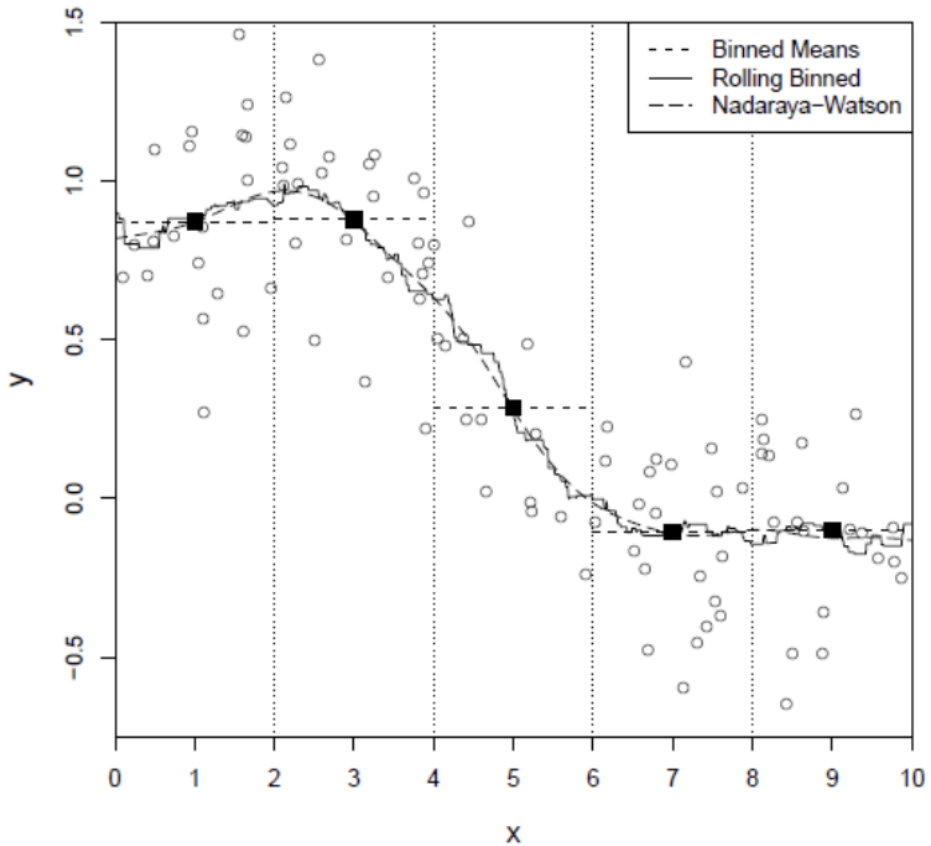
This is a **step function** estimator of $m(x)$: as x moves, the estimate jumps whenever an observation enters or leaves the window.

Binned Means Estimator



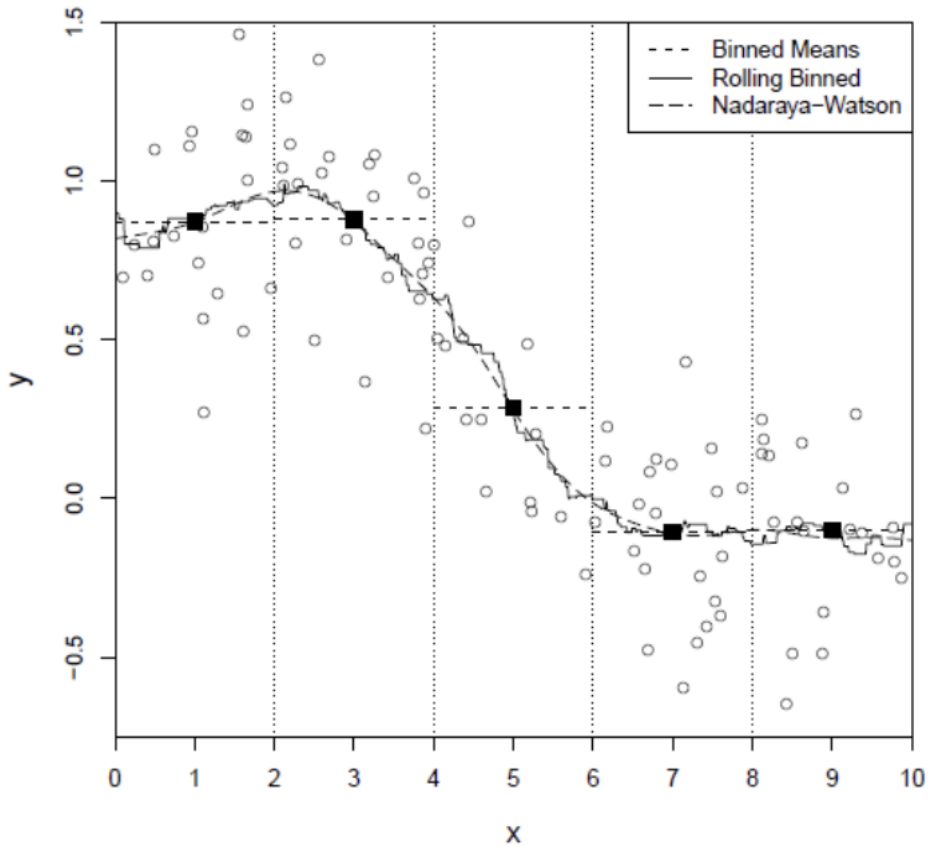
- In panel (a): one flat average per bin.
- The estimate jumps at the edges of each bin.

Binned Means Estimator



- In panel (a): one flat average per bin.
- The estimate jumps at the edges of each bin.

Binned Means Estimator



- In panel (a): one flat average per bin.
- The estimate jumps at the edges of each bin.
- **Smooth weights:** give observations closer to x more weight, and let the weight fade gradually (kernel regression, local linear regression). This produces a smooth $\hat{m}(x)$. $\Rightarrow$ Next slides.

Kernel Regression

- The source of the discontinuity in the step function is that the weights are discontinuous indicator functions:
  - i.e. the function $1\{|X_i - x| \leq h\}$ assigns equal weight $= 1$ to all points in the window $[x - h, x + h]$ and 0 to all others.

Kernel Regression

- The source of the discontinuity in the step function is that the weights are discontinuous indicator functions:
  - i.e. the function $1\{|X_i - x| \leq h\}$ assigns equal weight $= 1$ to all points in the window $[x - h, x + h]$ and 0 to all others.
- If instead the weights are continuous functions: $\hat{m}(x)$ will also be continuous in x.

Kernel Regression

- The source of the discontinuity in the step function is that the weights are discontinuous indicator functions:
  - i.e. the function $1\{|X_i - x| \leq h\}$ assigns equal weight $= 1$ to all points in the window $[x - h, x + h]$ and 0 to all others.
- If instead the weights are continuous functions: $\hat{m}(x)$ will also be continuous in x.
- e.g. the **Nadaraya–Watson** estimator uses a smooth kernel function $K(\cdot)$, which gives more weight to points closer to x and vice versa.
  - A kernel function is a bounded probability density function which is symmetric about zero satisfying:

$$\int_{-\infty}^{\infty} u^2 K(u)\, du = 1.$$

Nadaraya–Watson Estimator

The Nadaraya–Watson method estimates $m(x) = \mathbb{E}[Y|X = x]$ as a **weighted local average** of nearby Y_i values:

$$\hat{m}(x) = \frac{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right) Y_i}{\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)}$$

- $X_i - x$ measures how close observation i is to the point of interest.
- $K(\cdot)$: kernel function (smooth, symmetric, positive) ensures scale invariance.
- h: bandwidth (controls smoothness). The larger, the smoother.
- **Kernels**: Rectangular, **Gaussian**, Epanechnikov, Triangular.
- Generalizes the discrete-window estimator by assigning **smooth weights** that decline with distance from x.

Local Linear Regression (LLR)

- The Nadaraya–Watson estimator estimates a local constant that works well if $m(x)$ is roughly constant within the bandwidth h. **The local slope is ignored!**

- The local linear estimator fits a local line at each point x:

$$\min_{a,b} \sum_{i=1}^{n} \left(Y_i - a - b(X_i - x)\right)^2 K\left(\frac{X_i - x}{h}\right).$$

- If $b = 0$:

Local Linear Regression (LLR)

- The Nadaraya–Watson estimator estimates a local constant that works well if $m(x)$ is roughly constant within the bandwidth h. **The local slope is ignored!**

- The local linear estimator fits a local line at each point x:

$$\min_{a,b} \sum_{i=1}^{n} \left(Y_i - a - b(X_i - x)\right)^2 K\left(\frac{X_i - x}{h}\right).$$

- If $b = 0$: we get the minimization problem of Nadaraya–Watson.

Local Linear Regression (LLR)

- The Nadaraya–Watson estimator estimates a local constant that works well if $m(x)$ is roughly constant within the bandwidth h. **The local slope is ignored!**

- The local linear estimator fits a local line at each point x:

$$\min_{a,b} \sum_{i=1}^{n} \left(Y_i - a - b(X_i - x)\right)^2 K\left(\frac{X_i - x}{h}\right).$$

- If $b = 0$: we get the minimization problem of Nadaraya–Watson.

- if $h \to \infty$:

Local Linear Regression (LLR)

- The Nadaraya–Watson estimator estimates a local constant that works well if $m(x)$ is roughly constant within the bandwidth h. **The local slope is ignored!**

- The local linear estimator fits a local line at each point x:

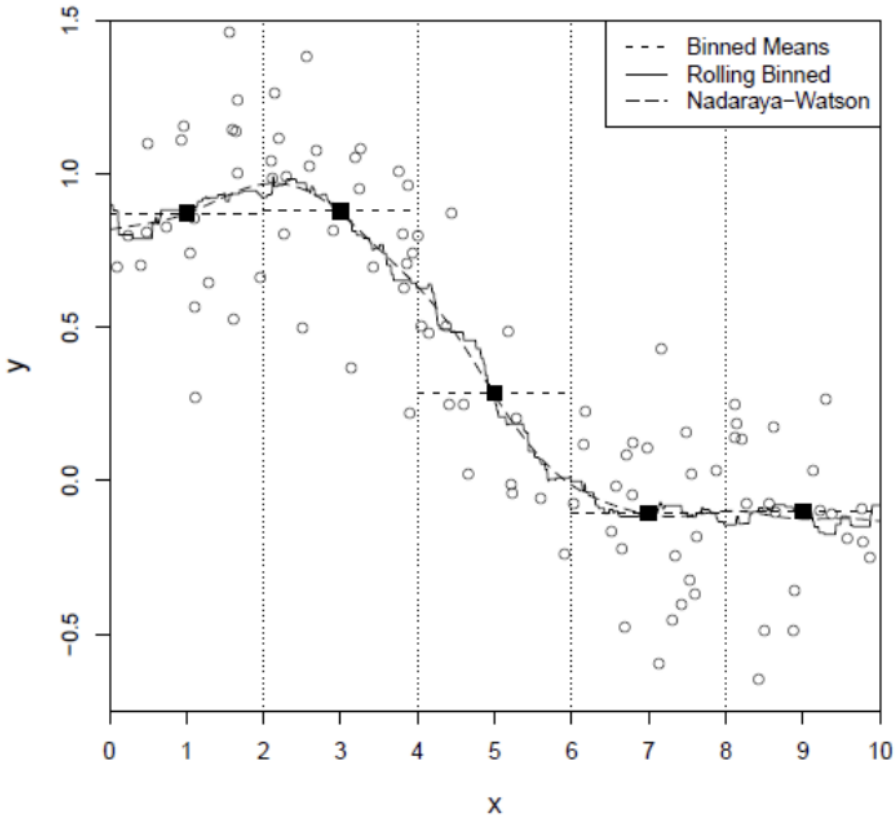
$$\min_{a,b} \sum_{i=1}^{n} \left(Y_i - a - b(X_i - x)\right)^2 K\left(\frac{X_i - x}{h}\right).$$

- If $b = 0$: we get the minimization problem of Nadaraya–Watson.

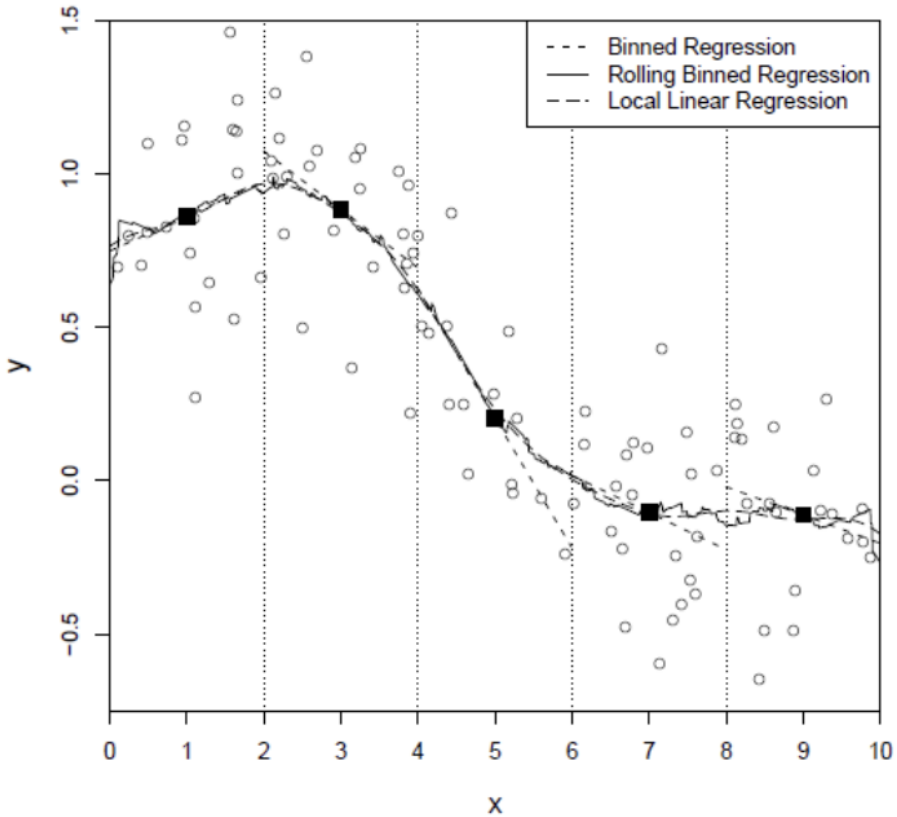
- if $h \to \infty$: we simply get the OLS estimator including all observations with equal weight.

- Hence it provides estimators of the slope at x.

Nadaraya–Watson Estimator and Local Linear Regression



(a) Nadaraya-Watson



(b) Local Linear

Local Polynomial Estimator

- Nadaraya–Watson fits a local **constant** ($p = 0$) and LLR fits a local **line** ($p = 1$).

Local Polynomial Estimator

- Nadaraya–Watson fits a local **constant** ($p = 0$) and LLR fits a local **line** ($p = 1$).
- **Natural generalization:** fit a local polynomial of degree p:

Local Polynomial Estimator

- Nadaraya–Watson fits a local **constant** ($p = 0$) and LLR fits a local **line** ($p = 1$).
- **Natural generalization:** fit a local polynomial of degree p:

$$m(X_i) \approx a_0 + a_1(X_i - x) + a_2(X_i - x)^2 + \cdots + a_p(X_i - x)^p.$$

Estimate the coefficients $(a_0, \ldots, a_p)$ by:

$$\min_{a_0, \ldots, a_p} \sum_{i=1}^{n} \left[Y_i - \sum_{j=0}^{p} a_j (X_i - x)^j \right]^2 K\left(\frac{X_i - x}{h} \right).$$

The fitted value at x is $\hat{m}_{LP}(x) = \hat{a}_0$.

- Higher $p \to$ lower bias, higher variance due to more parameters.
- $p = 1$:

Local Polynomial Estimator

- Nadaraya–Watson fits a local **constant** ($p = 0$) and LLR fits a local **line** ($p = 1$).
- **Natural generalization:** fit a local polynomial of degree p:

$$m(X_i) \approx a_0 + a_1(X_i - x) + a_2(X_i - x)^2 + \cdots + a_p(X_i - x)^p.$$

Estimate the coefficients $(a_0, \ldots, a_p)$ by:

$$\min_{a_0,\ldots,a_p} \sum_{i=1}^{n} \left[Y_i - \sum_{j=0}^{p} a_j (X_i - x)^j \right]^2 K\left(\frac{X_i - x}{h} \right).$$

The fitted value at x is $\hat{m}_{LP}(x) = \hat{a}_0$.

- Higher $p \to$ lower bias, higher variance due to more parameters.
- $p = 1$: local linear

Local Polynomial Estimator

- Nadaraya–Watson fits a local **constant** ($p = 0$) and LLR fits a local **line** ($p = 1$).
- **Natural generalization:** fit a local polynomial of degree p:

$$m(X_i) \approx a_0 + a_1(X_i - x) + a_2(X_i - x)^2 + \cdots + a_p(X_i - x)^p.$$

Estimate the coefficients $(a_0, \ldots, a_p)$ by:

$$\min_{a_0,\ldots,a_p} \sum_{i=1}^{n} \left[Y_i - \sum_{j=0}^{p} a_j (X_i - x)^j \right]^2 K\left(\frac{X_i - x}{h} \right).$$

The fitted value at x is $\hat{m}_{LP}(x) = \hat{a}_0$.

- Higher $p \to$ lower bias, higher variance due to more parameters.
- $p = 1$: local linear
- $p = 2$: local quadratic

The Bandwidth

The bandwidth h determines how *local* the nonparametric regression is.

Bandwidth	Effect	Bias	Variance
Small h	Very local fit; follows noise	Low bias	High variance
Large h	Very smooth fit; averages widely	High bias	Low variance

The bandwidth controls the **bias–variance tradeoff**.

Rule of Thumb (ROT) for NW

- A **rule-of-thumb bandwidth** gives a practical starting point based on theoretical approximations.

For a Gaussian kernel ($K(u) = \phi(u)$, the standard normal density) and roughly normal X:

$$h_{\text{rot}} = 1.06\, n^{-1/5}\, \hat{\sigma}_X$$

where n is the sample size and $\hat{\sigma}_X$ is the sample standard deviation of X.

Rule of Thumb (ROT) for NW

- A **rule-of-thumb bandwidth** gives a practical starting point based on theoretical approximations.

For a Gaussian kernel ($K(u) = \phi(u)$, the standard normal density) and roughly normal X:

$$h_{\text{rot}} = 1.06\, n^{-1/5}\, \hat{\sigma}_X$$

where n is the sample size and $\hat{\sigma}_X$ is the sample standard deviation of X.

- Larger samples: smaller h (more local detail).
- More dispersed X: larger h (wider smoothing window).

Rule of Thumb (ROT) for NW

- A **rule-of-thumb bandwidth** gives a practical starting point based on theoretical approximations.

For a Gaussian kernel ($K(u) = \phi(u)$, the standard normal density) and roughly normal X:

$$h_{\text{rot}} = 1.06\, n^{-1/5}\, \hat{\sigma}_X$$

where n is the sample size and $\hat{\sigma}_X$ is the sample standard deviation of X.

- Larger samples: smaller h (more local detail).
- More dispersed X: larger h (wider smoothing window).
- It provides a reasonable default but does not adapt to the actual data pattern (it uses only the spread of X ($\hat{\sigma}_X$) and ignores how Y actually varies with X).

Cross-Validation (Data-Driven)

- The ROT ignores how Y varies with X. **Cross-validation (CV)** instead lets the data choose h, by asking which bandwidth *predicts* Y best.

Leave-one-out CV criterion:

$$CV(h) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{m}_{-i}(X_i; h))^2$$

Where $\hat{m}_{-i}(X_i; h)$ is estimated without observation i.

As described before:

- Pick a grid of h values.

Cross-Validation (Data-Driven)

- The ROT ignores how Y varies with X. **Cross-validation (CV)** instead lets the data choose h, by asking which bandwidth *predicts* Y best.

Leave-one-out CV criterion:

$$CV(h) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{m}_{-i}(X_i; h))^2$$

Where $\hat{m}_{-i}(X_i; h)$ is estimated without observation i.

As described before:

- Pick a grid of h values.
- Compute $CV(h)$ for each (leaving one out — fold or observation).

Cross-Validation (Data-Driven)

- The ROT ignores how Y varies with X. **Cross-validation (CV)** instead lets the data choose h, by asking which bandwidth *predicts* Y best.

Leave-one-out CV criterion:

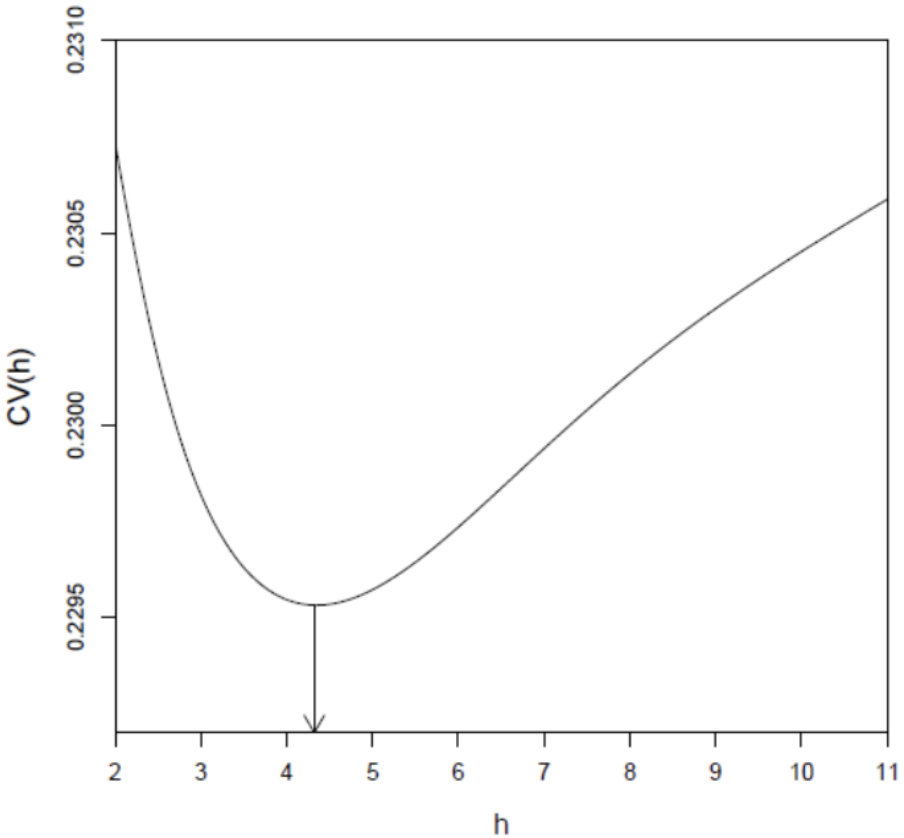
$$CV(h) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{m}_{-i}(X_i; h))^2$$

Where $\hat{m}_{-i}(X_i; h)$ is estimated without observation i.

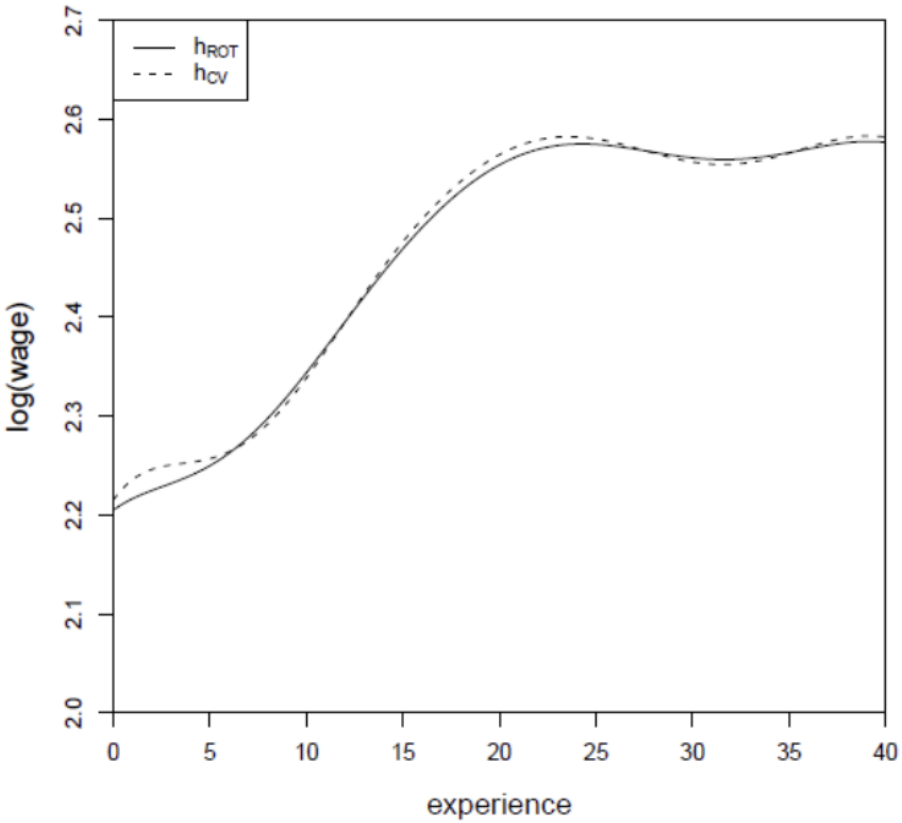
As described before:

- Pick a grid of h values.
- Compute $CV(h)$ for each (leaving one out — fold or observation).
- Choose $\hat{h}$ that minimizes $CV(h)$.

Mincer Non-Parametric Regression



(a) Cross-Validation Criterion



(b) Local Linear Regression

Figure 19.4: log(*wage*) regression on *experience*

Conditional Variance

LLR can be expressed with regressor vector $Z_i(x) : Z = [X_i - x]$ and coefficient vector $\beta(x)$.

- Applying weighted least squares with the kernel weights we obtain the LL estimator:

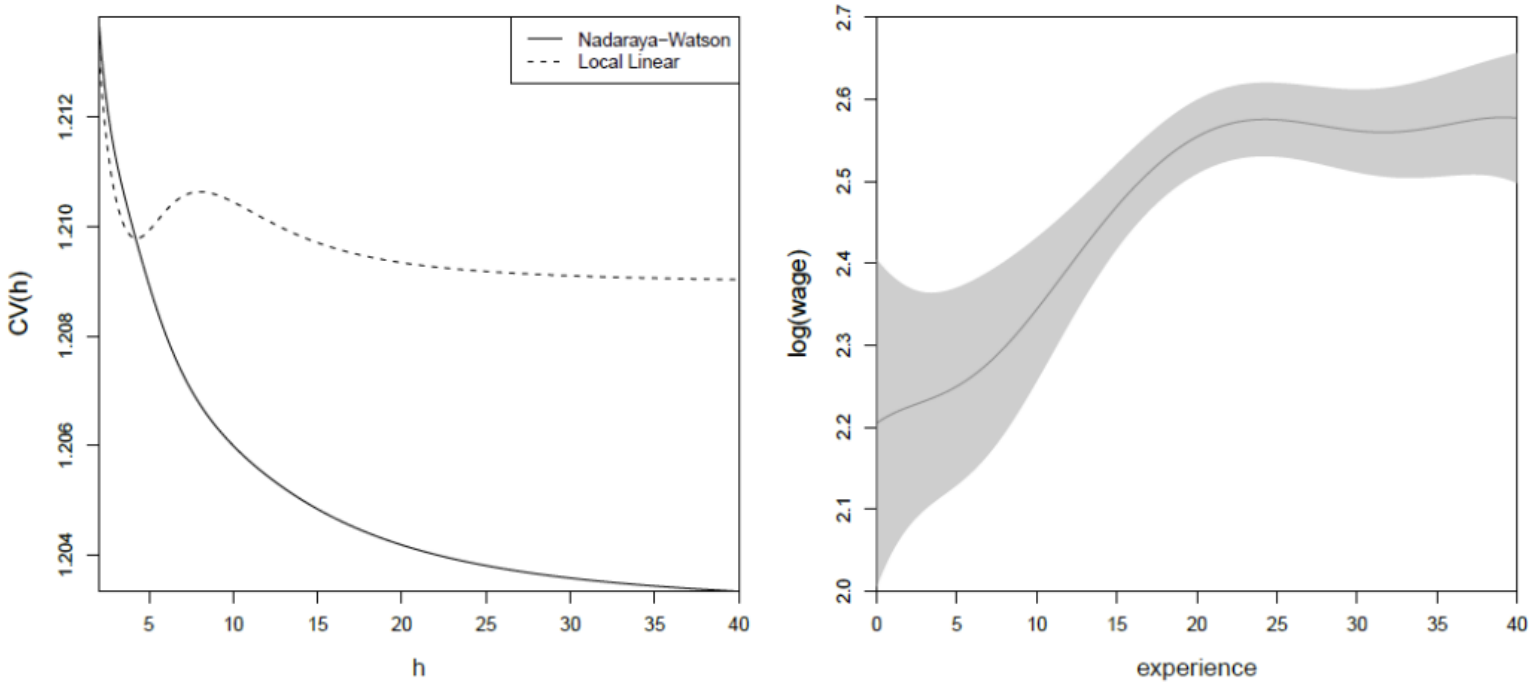
$$\widehat{\beta}_{LL}(x) = \left(\sum_{i=1}^{n} K\left(\frac{X_i - x}{h} \right) Z_i(x) Z_i(x)' \right)^{-1} \sum_{i=1}^{n} K\left(\frac{X_i - x}{h} \right) Z_i(x) Y_i$$
$$= (Z' K Z)^{-1} Z' K Y$$

- where: $\mathrm{K} = \mathrm{diag}\{K((X_1 - x)/h), \ldots, K((X_n - x)/h)\}$
- Notice that the matrices Z and K depend on x and h.

Conditional Variance

In this course we do not study in detail conditional nor asymptotic variance, but note that it is also a weighted "sandwich":

$$\widehat{V}_{\hat{\beta}}(x) = (Z'KZ)^{-1}\left(\sum_{i=1}^{n} K\left(\frac{X_i - x}{h}\right)^2 Z_i(x)Z_i(x)'\hat{e}_i^2\right)(Z'KZ)^{-1}.$$



(a) Cross-Validation for Conditional Variance (b) Regression with Confidence Bands

Figure 19.5: Confidence Band Construction

R code

Outline

- Non-parametric Regression
- Series and Spline Regression
- Regression Trees
- Bagging and Random Forest

Series Regression

- **Kernel methods (NW, LLR) are local:** they run a separate weighted regression at each point x, using only nearby observations.

Series Regression

- **Kernel methods (NW, LLR) are local:** they run a separate weighted regression at each point x, using only nearby observations.
- Now we will see: **Series methods** - they approximate the whole function $m(x)$ at once with a flexible combination of K basis functions of X, estimated by OLS on all observations.

Series Regression

- **Kernel methods (NW, LLR) are local:** they run a separate weighted regression at each point x, using only nearby observations.
- Now we will see: **Series methods** - they approximate the whole function $m(x)$ at once with a flexible combination of K basis functions of X, estimated by OLS on all observations.
  - Common choices for the basis are **polynomials** $(1, X, X^2, \ldots, X^{K-1})$ or **splines** (piecewise polynomials joined at knots).

Series Regression

- **Kernel methods (NW, LLR) are local:** they run a separate weighted regression at each point x, using only nearby observations.
- Now we will see: **Series methods** - they approximate the whole function $m(x)$ at once with a flexible combination of K basis functions of X, estimated by OLS on all observations.
  - Common choices for the basis are **polynomials** $(1, X, X^2, \ldots, X^{K-1})$ or **splines** (piecewise polynomials joined at knots).

Linear series regression models take the form

$$Y = X_K' \beta_K + e_K$$

where $X_K = X_K(X)$ is the $K \times 1$ vector of transformations of X, and β_K is the coefficient vector.

Series Regression

Instead of assuming a simple linear model:

$$m(x) = \beta_0 + \beta_1 X$$

we represent $m(x)$ as a **series expansion** that allows nonlinear shapes:

$$m_K(x) = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_p X^p.$$

Here:

- $K = p + 1$ is the number of basis functions (constant + p polynomial terms).

Series Regression

Instead of assuming a simple linear model:

$$m(x) = \beta_0 + \beta_1 X$$

we represent $m(x)$ as a **series expansion** that allows nonlinear shapes:

$$m_K(x) = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_p X^p.$$

Here:

- $K = p + 1$ is the number of basis functions (constant + p polynomial terms).
- Increasing K makes the model more flexible.

Series Regression

Instead of assuming a simple linear model:

$$m(x) = \beta_0 + \beta_1 X$$

we represent $m(x)$ as a **series expansion** that allows nonlinear shapes:

$$m_K(x) = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_p X^p.$$

Here:

- $K = p + 1$ is the number of basis functions (constant + p polynomial terms).
- Increasing K makes the model more flexible.
- The series regression approximates $m(x)$ by fitting a weighted (by β_p) combination of these basis functions.

Series Regression

We build an $n \times K$ **design matrix** $\mathbf{X}_K$, whose columns are transformations (basis functions) of X:

$$\mathbf{X}_K = \overbrace{\begin{bmatrix} 1 & X_1 & X_1^2 & \cdots & X_1^p \\ 1 & X_2 & X_2^2 & \cdots & X_2^p \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_n & X_n^2 & \cdots & X_n^p \end{bmatrix}}^{K=p+1 \text{ columns (basis functions } 1, X, \ldots, X^p)} \Bigg\} \; n \text{ rows (observations)}$$

Then we estimate the coefficients by OLS:

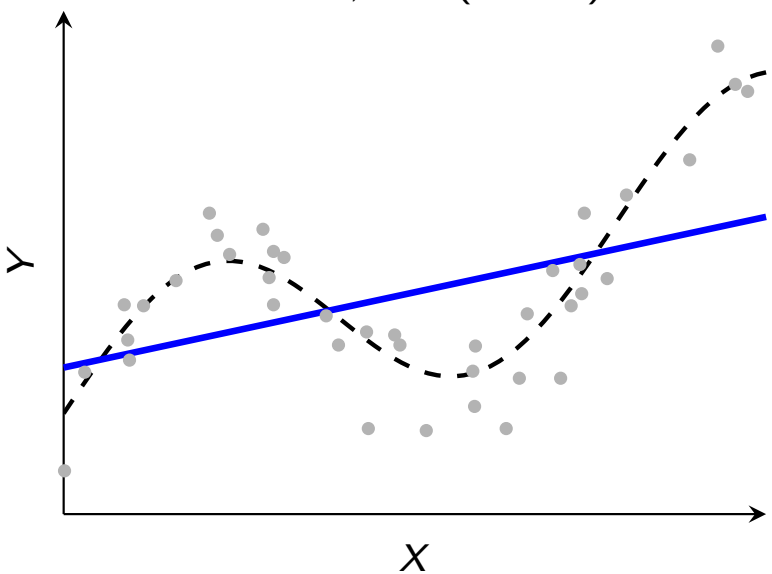
$$\underset{K\times 1}{\hat{\beta}_K} = \underset{K\times K}{\left(\mathbf{X}_K'\mathbf{X}_K\right)^{-1}} \underset{K\times n}{\mathbf{X}_K'} \underset{n\times 1}{\mathbf{Y}}$$

- That is, we run a single OLS regression of Y on all polynomial terms of X.
- Row i is $X_K(X_i)'$: observation i evaluated at every basis function.
- The estimator of $m(x)$ at any point x is $\hat{m}_K(x) = X_K(x)'\hat{\beta}_K$, where $X_K(x) = (1, x, x^2, \ldots, x^p)'$.

It is called series regression because $m(x)$ is approximated by a series of terms $\beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_p x^p$

Series Regression: Choosing K

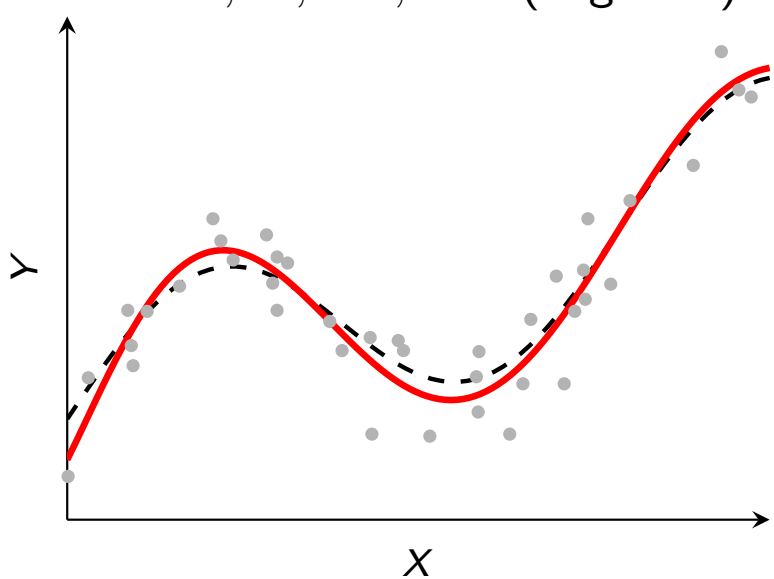
$K = 2$: $1, X$ (linear)



Too rigid: misses the curvature

$\Rightarrow$ **high bias**, low variance

$K = 7$: $1, X, \ldots, X^6$ (degree 6)

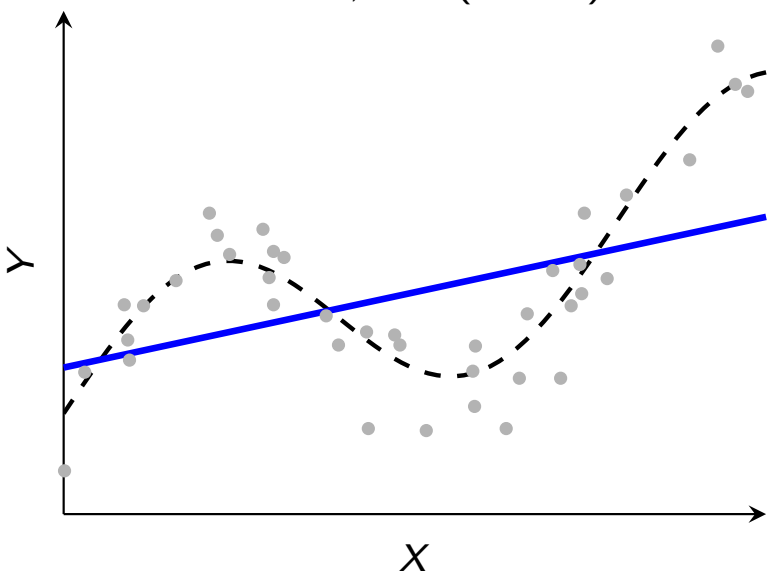


Flexible: tracks the true $m(x)$

$\Rightarrow$ lower bias, **higher variance**

Series Regression: Choosing K

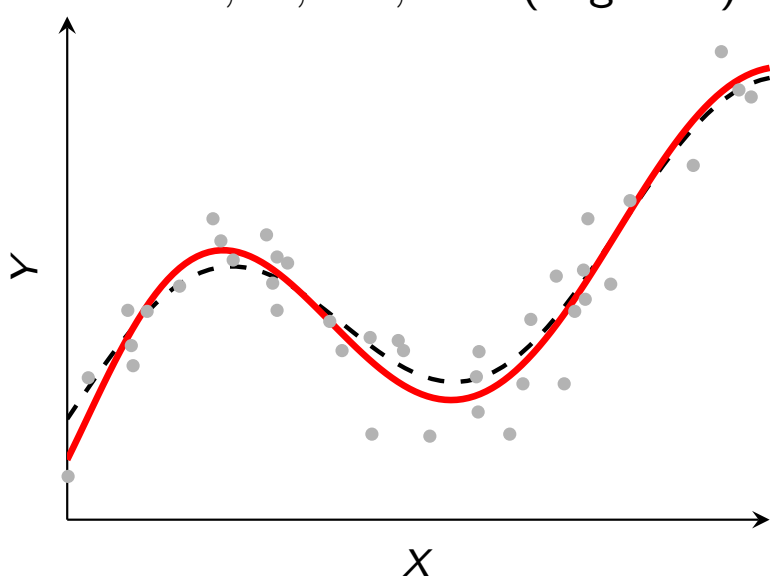
$K = 2$: $1, X$ (linear)



Too rigid: misses the curvature

$\Rightarrow$ **high bias**, low variance

$K = 7$: $1, X, \ldots, X^6$ (degree 6)



Flexible: tracks the true $m(x)$

$\Rightarrow$ lower bias, **higher variance**

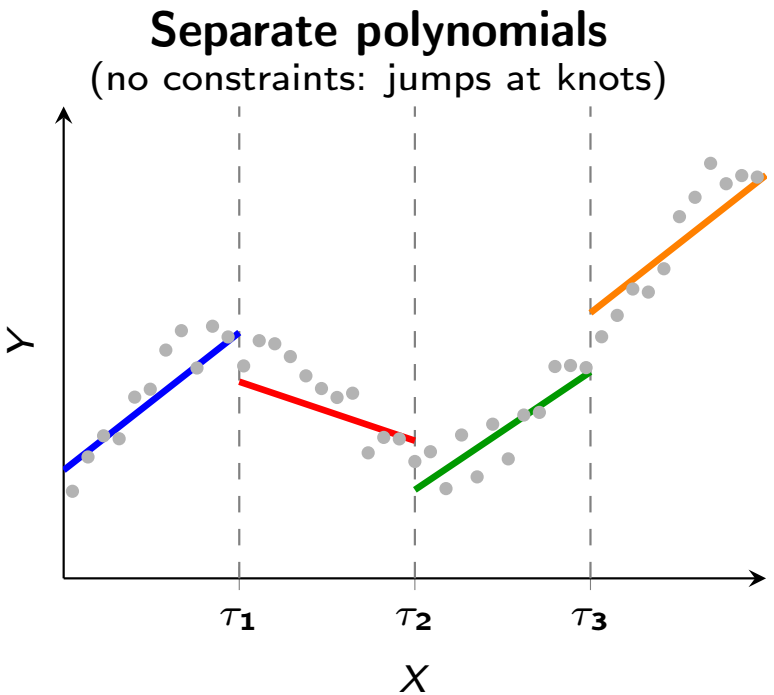
- Same data, same OLS; only the number of basis functions K changes. Dashed line: true $m(x)$.
- K plays the role of h in kernel regression: it must be chosen (e.g., by CV) to balance bias and variance.

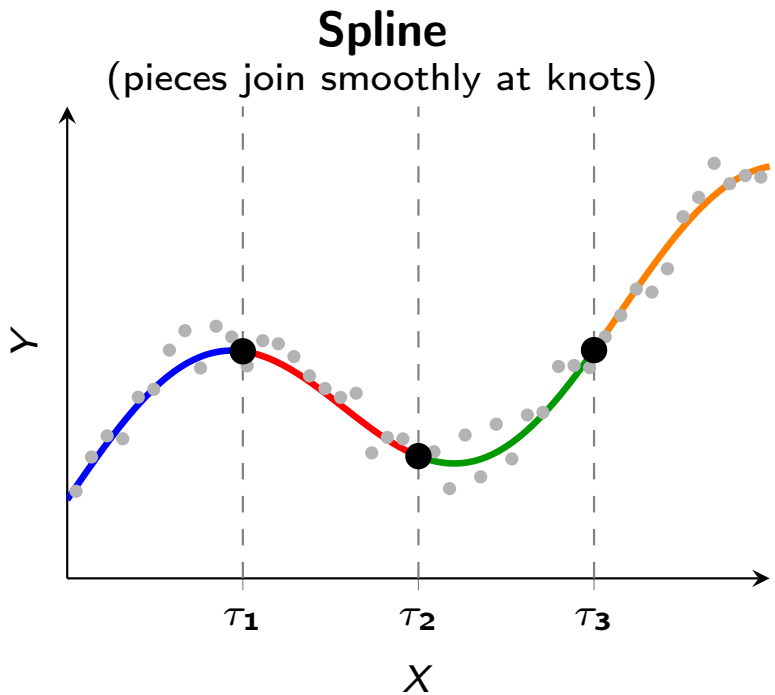
Spline Regression

A **spline** is a piecewise polynomial.

- Split the range of X into segments and fit a polynomial in each one.
- The join points between segments are called **knots** $\tau_1 < \tau_2 < \cdots < \tau_J$.
- Flexibility is governed by the **number of knots**: more knots means more segments and a more flexible fit.
- To impose smoothness and parsimony, a spline of degree p is constrained to be continuous with continuous derivatives up to order $p - 1$ at each knot, so the pieces join with no jumps or kinks.
- The degree is usually pre-selected: linear ($p = 1$), quadratic ($p = 2$), or cubic ($p = 3$, the most common).

Spline Regression: Visual Intuition





- Each color is a different polynomial. The **knots** τ_j (dashed lines) mark where one piece ends and the next begins.
- The spline constraints remove the jumps, giving one smooth curve that is still flexible locally.

Spline Regression

A **linear spline** with one knot τ is

$$m_K(x) = \beta_0 + \beta_1 x + \beta_2 (x - \tau)\, 1\{x \geq \tau\}.$$

- For $x \leq \tau$:

Spline Regression

A **linear spline** with one knot τ is

$$m_K(x) = \beta_0 + \beta_1 x + \beta_2 (x - \tau)\, 1\{x \geq \tau\}.$$

- For $x \leq \tau$: the function $m_K(x) = \beta_0 + \beta_1 x$ is linear with slope β_1;

Spline Regression

A **linear spline** with one knot τ is

$$m_K(x) = \beta_0 + \beta_1 x + \beta_2 (x - \tau)\, 1\{x \geq \tau\}.$$

- For $x \leq \tau$: the function $m_K(x) = \beta_0 + \beta_1 x$ is linear with slope β_1;
- For $x \geq \tau$:

A **linear spline** with one knot τ is

$$m_K(x) = \beta_0 + \beta_1 x + \beta_2 (x - \tau)\, 1\{x \geq \tau\}.$$

- For $x \leq \tau$: the function $m_K(x) = \beta_0 + \beta_1 x$ is linear with slope β_1;
- For $x \geq \tau$: the function $m_K(x)$ is linear with slope $\beta_1 + \beta_2$;

Spline Regression

A **linear spline** with one knot τ is

$$m_K(x) = \beta_0 + \beta_1 x + \beta_2 (x - \tau)\, 1\{x \geq \tau\}.$$

- For $x \leq \tau$: the function $m_K(x) = \beta_0 + \beta_1 x$ is linear with slope β_1;
- For $x \geq \tau$: the function $m_K(x)$ is linear with slope $\beta_1 + \beta_2$;
- Note that β_2 is the change in the slope at τ.

Spline Regression

A **linear spline** with one knot τ is

$$m_K(x) = \beta_0 + \beta_1 x + \beta_2 (x - \tau)\, 1\{x \geq \tau\}.$$

- For $x \leq \tau$: the function $m_K(x) = \beta_0 + \beta_1 x$ is linear with slope β_1;
- For $x \geq \tau$: the function $m_K(x)$ is linear with slope $\beta_1 + \beta_2$;
- Note that β_2 is the change in the slope at τ.
- Why $(x - \tau)$?

Spline Regression

A **linear spline** with one knot τ is

$$m_K(x) = \beta_0 + \beta_1 x + \beta_2 (x - \tau)\, 1\{x \geq \tau\}.$$

- For $x \leq \tau$: the function $m_K(x) = \beta_0 + \beta_1 x$ is linear with slope β_1;
- For $x \geq \tau$: the function $m_K(x)$ is linear with slope $\beta_1 + \beta_2$;
- Note that β_2 is the change in the slope at τ.
- Why $(x - \tau)$? The term $(x - \tau)1\{x \geq \tau\}$ equals 0 at $x = \tau$, so the two lines **meet at the knot** and only the slope changes. With $x\,1\{x \geq \tau\}$, the function would **jump** by $\beta_2 \tau$ at τ.

A **linear spline** with one knot τ is

$$m_K(x) = \beta_0 + \beta_1 x + \beta_2 (x - \tau)\, 1\{x \geq \tau\}.$$

- For $x \leq \tau$: the function $m_K(x) = \beta_0 + \beta_1 x$ is linear with slope β_1;
- For $x \geq \tau$: the function $m_K(x)$ is linear with slope $\beta_1 + \beta_2$;
- Note that β_2 is the change in the slope at τ.
- Why $(x - \tau)$? The term $(x - \tau)1\{x \geq \tau\}$ equals 0 at $x = \tau$, so the two lines **meet at the knot** and only the slope changes. With $x\, 1\{x \geq \tau\}$, the function would **jump** by $\beta_2 \tau$ at τ.

A **linear spline** with two knots $\tau_1 < \tau_2$ is

$$m_K(x) = \beta_0 + \beta_1 x + \beta_2 (x - \tau_1)\, 1\{x \geq \tau_1\} + \beta_3 (x - \tau_2)\, 1\{x \geq \tau_2\}.$$

A **quadratic spline** with one knot is

$$m_K(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 (x - \tau)^2 \, 1\{x \geq \tau\}.$$

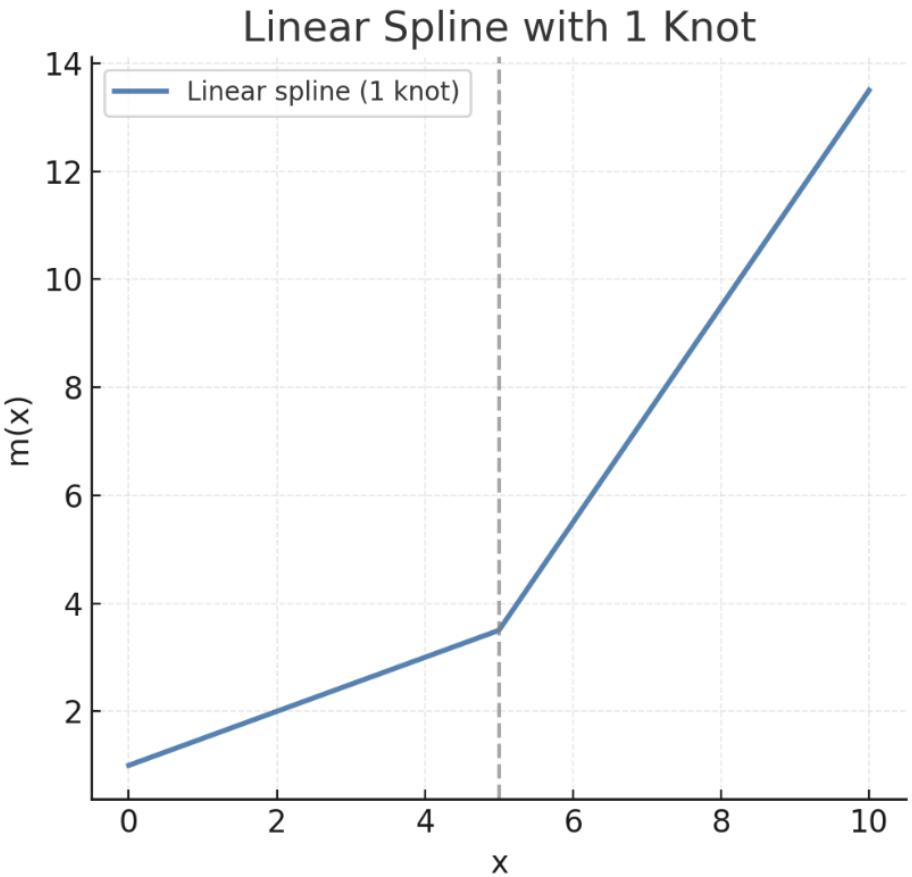
- Observe that for $x \leq \tau$ the function is the quadratic $\beta_0 + \beta_1 x + \beta_2 x^2$ with second derivative $m''_K(\tau) = 2\beta_2$;
- For $x \geq \tau$ the second derivative is $m''_K(\tau) = 2(\beta_2 + \beta_3)$; so $2\beta_3$ is the change in the second derivative at τ.
- The first derivative at $x = \tau$ is the continuous function $m'_K(\tau) = \beta_1 + 2\beta_2\tau$.

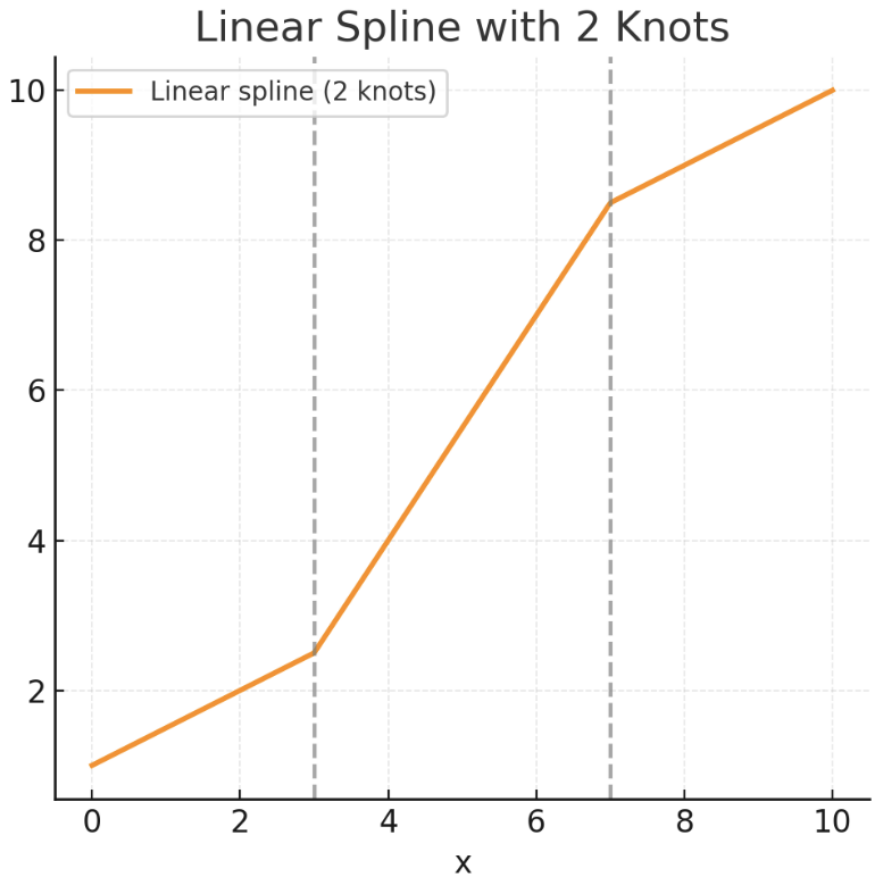
Spline Regression

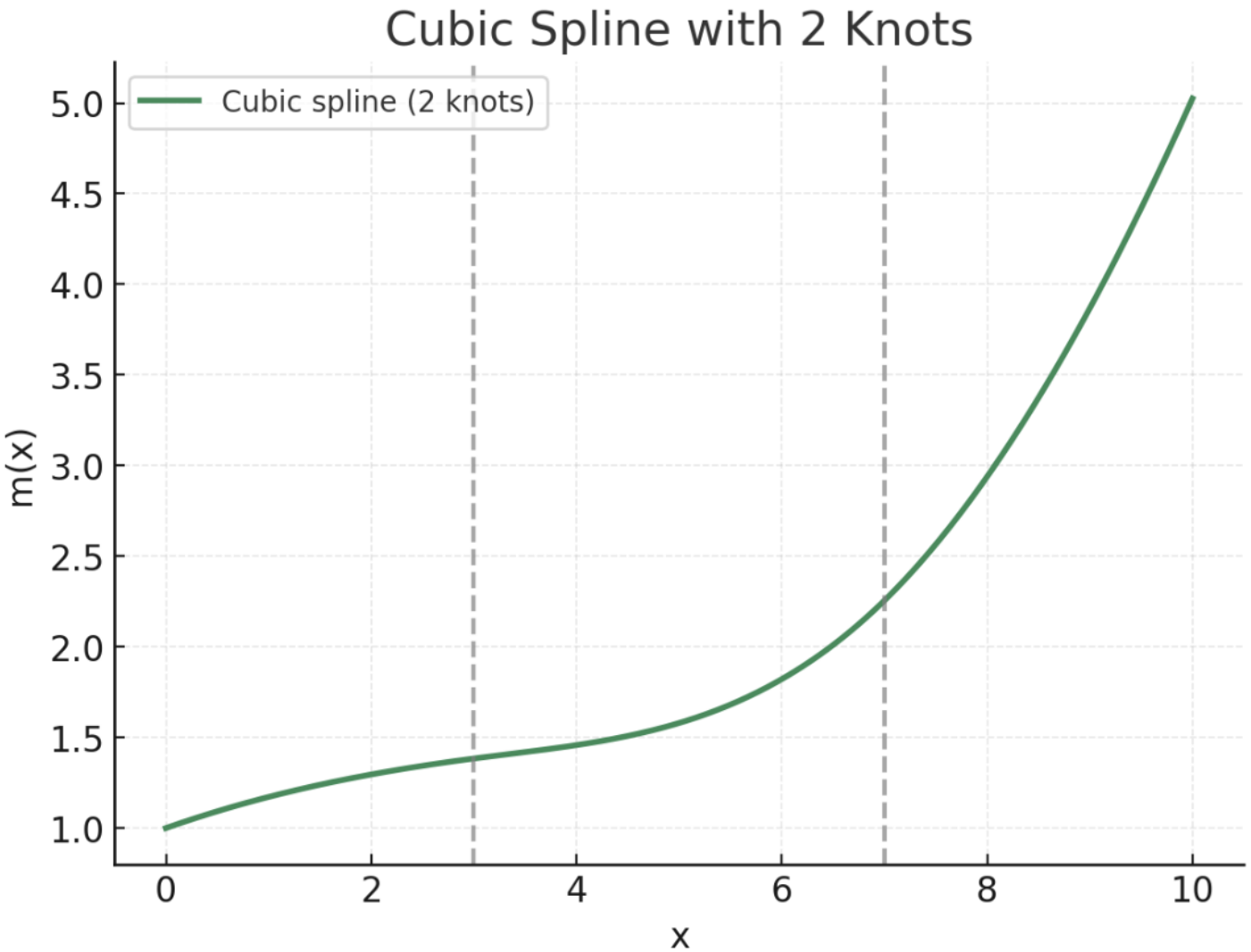
In general, a p^{th}-order spline with N knots $\tau_1 < \tau_2 < \cdots < \tau_N$ is

$$m_K(x) = \sum_{j=0}^{p} \beta_j x^j + \sum_{k=1}^{N} \beta_{p+k} (x - \tau_k)^p \, 1\{x \geq \tau_k\}$$

Splines


Linear Spline with 1 Knot
Linear spline (1 knot)
m(x)
x
14
12
10
8
6
4
2
0
2
4
6
8
10


Linear Spline with 2 Knots
Linear spline (2 knots)
x
10
8
6
4
2
0
2
4
6
8
10



Cubic spline with two knots — smoother and continuously differentiable across segments.

Splines

- In practice a spline will depend critically on the choice of the knots τ_k.

Splines

- In practice a spline will depend critically on the choice of the knots τ_k.
- When X has an approximately **uniform distribution** it is common to space the knots **evenly** so all segments have the same length.

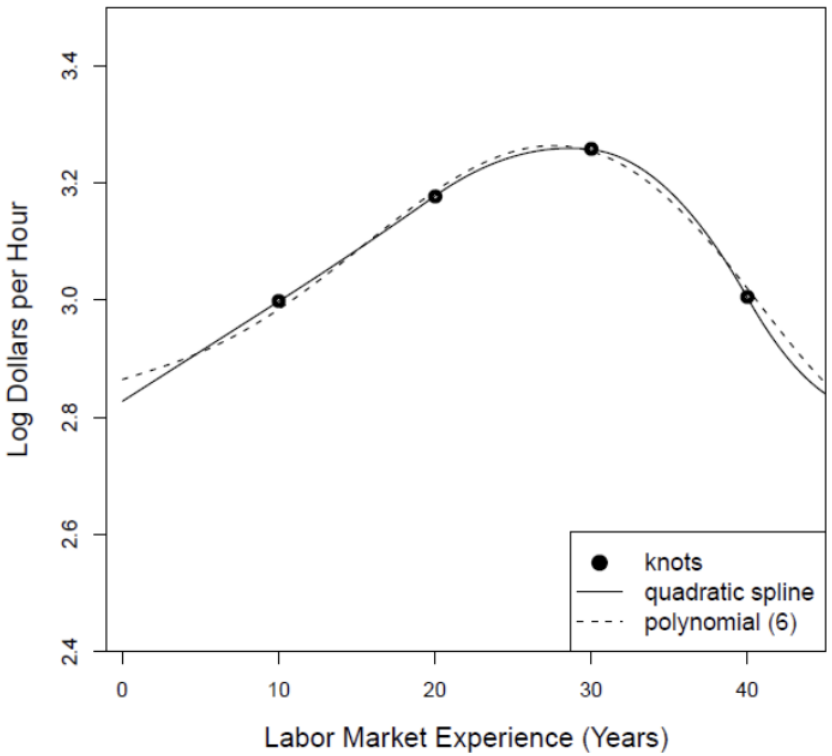
Splines

- In practice a spline will depend critically on the choice of the knots τ_k.
- When X has an approximately **uniform distribution** it is common to space the knots **evenly** so all segments have the same length.
- When the distribution of X is not uniform, an alternative is to set the knots at the quantiles $j/(N+1)$.

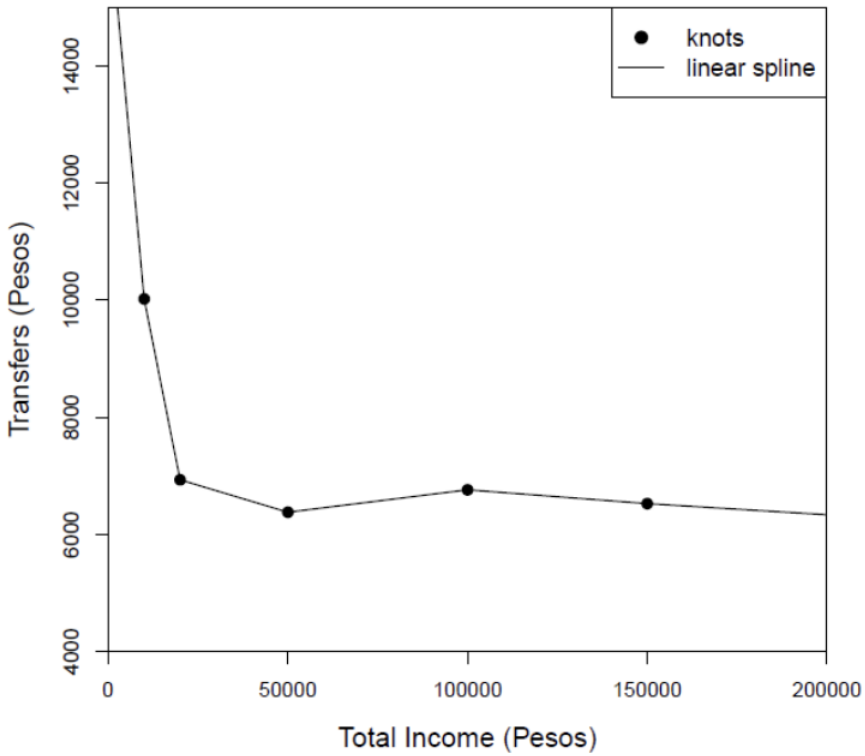
Splines

- In practice a spline will depend critically on the choice of the knots τ_k.
- When X has an approximately **uniform distribution** it is common to space the knots **evenly** so all segments have the same length.
- When the distribution of X is not uniform, an alternative is to set the knots at the quantiles $j/(N+1)$.
- A third alternative is to set the knots at the points where $m(x)$ has the greatest change in curvature.

Regression Spline Example



(a) Experience Profile



(b) Effect of Income on Transfers

Figure 20.2: Spline Regression Estimates

Outline

- Non-parametric Regression
- Series and Spline Regression
- Regression Trees
- Bagging and Random Forest

Motivation: Flexible Regression with Many Regressors

Goal: estimate $E[Y \mid X]$ without imposing a functional form.

Problem: X has many regressors, both continuous (income, age) and discrete (sex, region).

Motivation: Flexible Regression with Many Regressors

Goal: estimate $E[Y \mid X]$ without imposing a functional form.

Problem: X has many regressors, both continuous (income, age) and discrete (sex, region).

- Kernels: few observations *near* any point.
- Series: too many basis terms and interactions.

Motivation: Flexible Regression with Many Regressors

Goal: estimate $E[Y \mid X]$ without imposing a functional form.

Problem: X has many regressors, both continuous (income, age) and discrete (sex, region).

- Kernels: few observations *near* any point.
- Series: too many basis terms and interactions.

Wish list: a method that

- selects relevant variables and split points from the data,
- treats continuous and discrete regressors alike,
- is easy to interpret.

Motivation: Flexible Regression with Many Regressors

Goal: estimate $E[Y \mid X]$ without imposing a functional form.

Problem: X has many regressors, both continuous (income, age) and discrete (sex, region).

- Kernels: few observations *near* any point.
- Series: too many basis terms and interactions.

Wish list: a method that

- selects relevant variables and split points from the data,
- treats continuous and discrete regressors alike,
- is easy to interpret.

Idea: partition the regressor space into rectangles and predict with the mean of Y in each one.

Motivation: Flexible Regression with Many Regressors

Goal: estimate $E[Y \mid X]$ without imposing a functional form.

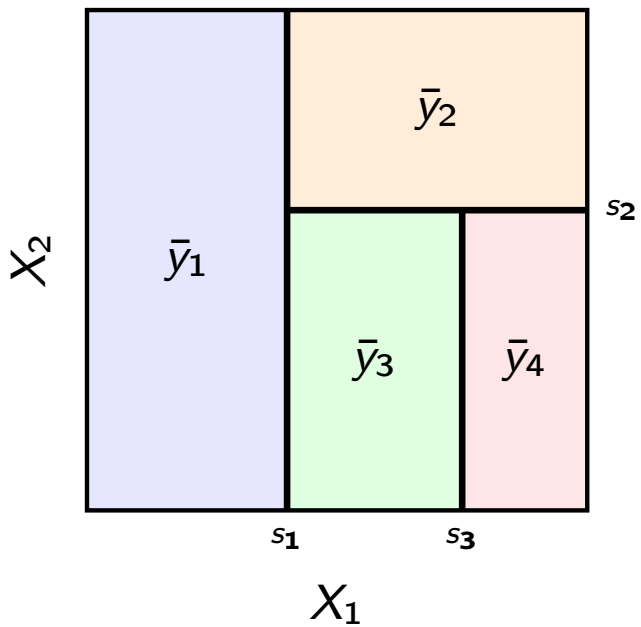
Problem: X has many regressors, both continuous (income, age) and discrete (sex, region).

- Kernels: few observations *near* any point.
- Series: too many basis terms and interactions.

Wish list: a method that

- selects relevant variables and split points from the data,
- treats continuous and discrete regressors alike,
- is easy to interpret.

Idea: partition the regressor space into rectangles and predict with the mean of Y in each one.



Split points chosen by the data; prediction is a **step function**.

Regression Trees

CART (Classification and Regression Trees) turns the idea into an estimator:

$$\hat{m}(x) \;=\; \sum_{m=1}^{M} \bar{y}_m \, 1\{x \in R_m\},$$

where $R_1, \ldots, R_M$ are rectangles and $\bar{y}_m$ is the mean of Y in R_m.

Regression Trees

CART (Classification and Regression Trees) turns the idea into an estimator:

$$\hat{m}(x) \;=\; \sum_{m=1}^{M} \bar{y}_m \, 1\{x \in R_m\},$$

where $R_1, \ldots, R_M$ are rectangles and $\bar{y}_m$ is the mean of Y in R_m.

- The rectangles come from **recursive binary splits** $\{X_j < s\}$ vs. $\{X_j \geq s\}$, which a tree records.
- $\hat{m}$ is a **step function**. With enough splits it can approximate any regression function.

Regression Trees

CART (Classification and Regression Trees) turns the idea into an estimator:

$$\hat{m}(x) \;=\; \sum_{m=1}^{M} \bar{y}_m \, 1\{x \in R_m\},$$

where $R_1, \ldots, R_M$ are rectangles and $\bar{y}_m$ is the mean of Y in R_m.

- The rectangles come from **recursive binary splits** $\{X_j < s\}$ vs. $\{X_j \geq s\}$, which a tree records.
- $\hat{m}$ is a **step function**. With enough splits it can approximate any regression function.
- It is like a 0^{th}**-order spline with free knots**: the data choose the knots, which are the split variables and the split points.
- Discrete regressors need no special treatment: a split just groups categories.

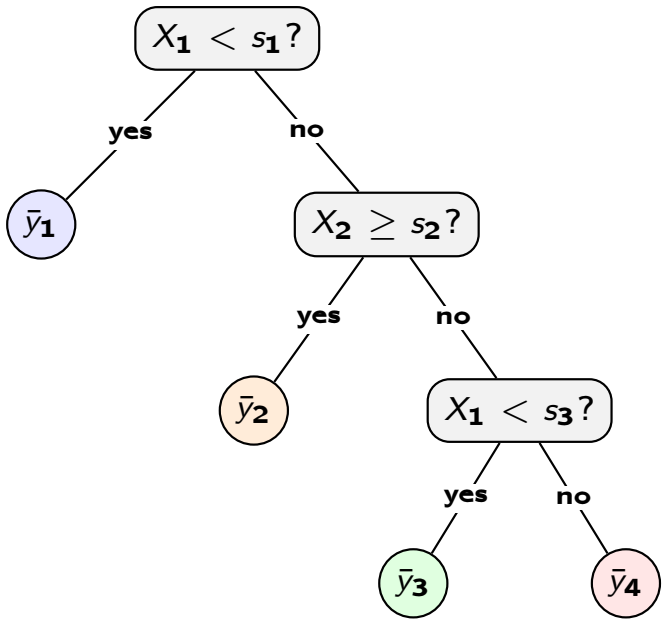
Regression Trees

CART (Classification and Regression Trees) turns the idea into an estimator:

$$\hat{m}(x) \;=\; \sum_{m=1}^{M} \bar{y}_m \, 1\{x \in R_m\},$$

where $R_1, \ldots, R_M$ are rectangles and $\bar{y}_m$ is the mean of Y in R_m.

- The rectangles come from **recursive binary splits** $\{X_j < s\}$ vs. $\{X_j \geq s\}$, which a tree records.
- $\hat{m}$ is a **step function**. With enough splits it can approximate any regression function.
- It is like a 0^{th}**-order spline with free knots**: the data choose the knots, which are the split variables and the split points.
- Discrete regressors need no special treatment: a split just groups categories.



Same partition as before, written as a tree.

Each leaf is one region R_m.

Regression Trees: Setup

Goal: estimate $m(x) = E[Y \mid X = x]$ for scalar Y and vector $X = (X_1, \ldots, X_p)$.

Each X_j must have an **order**, so that it can be split:

Continuous age, income	**Binary** female	**Ordinal** schooling level
split: $X_j < s$	split: $X_j = 0$ vs. 1	split: $X_j \leq k$

Regression Trees: Setup

Goal: estimate $m(x) = E[Y \mid X = x]$ for scalar Y and vector $X = (X_1, \ldots, X_p)$.

Each X_j must have an **order**, so that it can be split:

Continuous age, income	**Binary** female	**Ordinal** schooling level
split: $X_j < s$	split: $X_j = 0$ vs. 1	split: $X_j \leq k$

Categorical regressors have no natural order: transform them into a **set of binary variables**

Region $\in \{N, C, S\}$ → **one dummy per category**

Region	D_N	D_C	D_S
N	1	0	0
C	0	1	0
S	0	0	1

Regression Trees: Terminology

Language based on the metaphor of a living tree:

Regression Trees: Terminology

Language based on the metaphor of a living tree:

- **Branch**: a subsample, meaning a node and everything below it.

Regression Trees: Terminology

Language based on the metaphor of a living tree:

- **Branch**: a subsample, meaning a node and everything below it.
- **Leaves** (terminal nodes): branches that are not split further. Each leaf is one region R_m.

Regression Trees: Terminology

Language based on the metaphor of a living tree:

- **Branch**: a subsample, meaning a node and everything below it.
- **Leaves** (terminal nodes): branches that are not split further. Each leaf is one region R_m.
- **Growing**: adding splits.

Regression Trees: Terminology

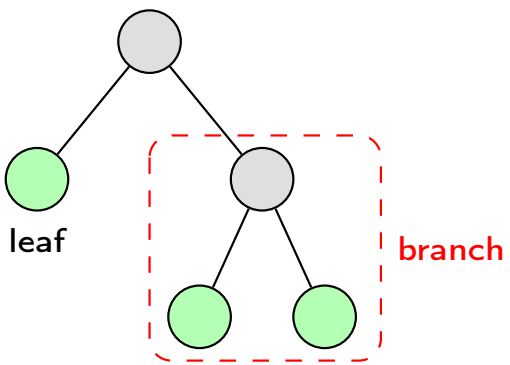
Language based on the metaphor of a living tree:

- **Branch**: a subsample, meaning a node and everything below it.
- **Leaves** (terminal nodes): branches that are not split further. Each leaf is one region R_m.
- **Growing**: adding splits.
- **Pruning**: removing splits.

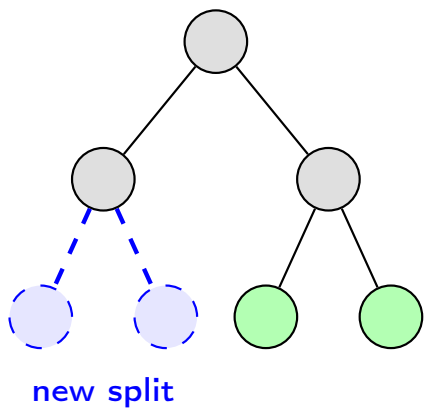
Regression Trees: Terminology

Language based on the metaphor of a living tree:

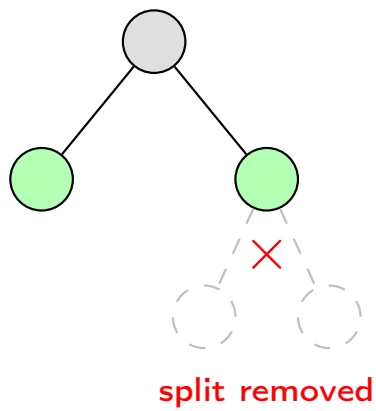
- **Branch**: a subsample, meaning a node and everything below it.
- **Leaves** (terminal nodes): branches that are not split further. Each leaf is one region R_m.
- **Growing**: adding splits.
- **Pruning**: removing splits.



A tree



Growing

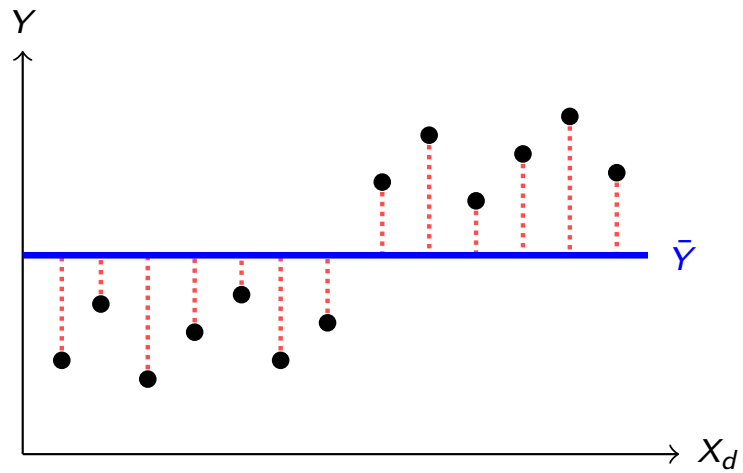


Pruning

Regression Trees: Step 0 — Start with One Branch

- No splits yet: the **entire sample is one branch**.
- With one group, the best constant prediction under squared loss is the **overall mean**:

$$\hat{m}(x) = \hat{\mu} = \bar{Y}$$



Dotted lines = errors. They are long because *one* number must fit everyone.

Regression Trees: Step 1 — Try One Split

Pick a regressor X_d and a cutoff γ. Then fit a **separate mean on each side**:

$$Y = \mu_1 1\{X_d \leq \gamma\} + \mu_2 1\{X_d > \gamma\} + e$$

- **For a fixed γ:** this is OLS on two dummies, so $\hat{\mu}_1 = \bar{Y}_L$ and $\hat{\mu}_2 = \bar{Y}_R$ (the group means).

Regression Trees: Step 1 — Try One Split

Pick a regressor X_d and a cutoff γ. Then fit a **separate mean on each side**:

$$Y = \mu_1 1\{X_d \leq \gamma\} + \mu_2 1\{X_d > \gamma\} + e$$

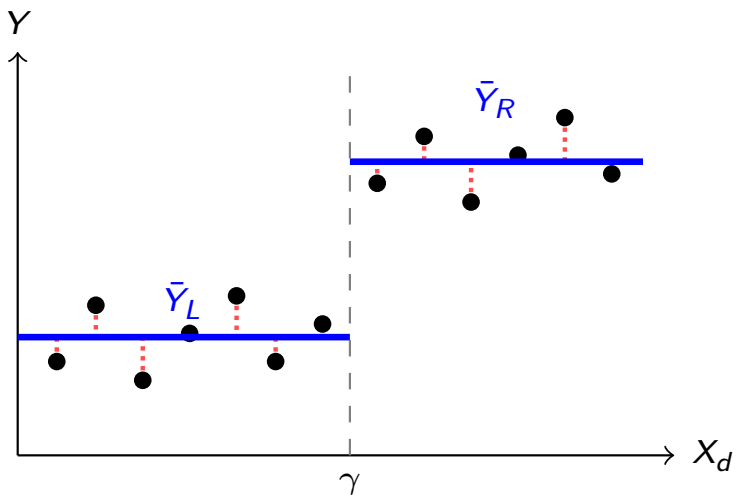
- **For a fixed γ:** this is OLS on two dummies, so $\hat{\mu}_1 = \bar{Y}_L$ and $\hat{\mu}_2 = \bar{Y}_R$ (the group means).
- γ **sits inside an indicator**, so the model is **nonlinear in** γ - this problem has no closed-form solution.

Regression Trees: Step 1 — Try One Split

Pick a regressor X_d and a cutoff γ. Then fit a **separate mean on each side**:

$$Y = \mu_1 1\{X_d \leq \gamma\} + \mu_2 1\{X_d > \gamma\} + e$$

- **For a fixed γ:** this is OLS on two dummies, so $\hat{\mu}_1 = \bar{Y}_L$ and $\hat{\mu}_2 = \bar{Y}_R$ (the group means).
- γ **sits inside an indicator**, so the model is **nonlinear in** γ - this problem has no closed-form solution.
  - ↪ We **search** over values of γ instead of solving first-order conditions.



Two means instead of one: the errors shrink.

Regression Trees: Step 2 — Choose the Best Cut

For each candidate cutoff, compute the SSR of the two-group fit:

$$SSR(\gamma) = \sum_{i:\, X_{di} \leq \gamma} (Y_i - \bar{Y}_L)^2 + \sum_{i:\, X_{di} > \gamma} (Y_i - \bar{Y}_R)^2$$

Regression Trees: Step 2 — Choose the Best Cut

For each candidate cutoff, compute the SSR of the two-group fit:

$$SSR(\gamma) = \sum_{i:\, X_{di} \leq \gamma} (Y_i - \bar{Y}_L)^2 + \sum_{i:\, X_{di} > \gamma} (Y_i - \bar{Y}_R)^2$$

- A good cut leaves each group **internally homogeneous**: everyone is close to their group's mean.

Regression Trees: Step 2 — Choose the Best Cut

For each candidate cutoff, compute the SSR of the two-group fit:

$$SSR(\gamma) = \sum_{i:\, X_{di} \le \gamma} (Y_i - \bar{Y}_L)^2 + \sum_{i:\, X_{di} > \gamma} (Y_i - \bar{Y}_R)^2$$

- A good cut leaves each group **internally homogeneous**: everyone is close to their group's mean.
- $SSR(\gamma)$ changes only when γ **crosses an observation**, so it is a step function.

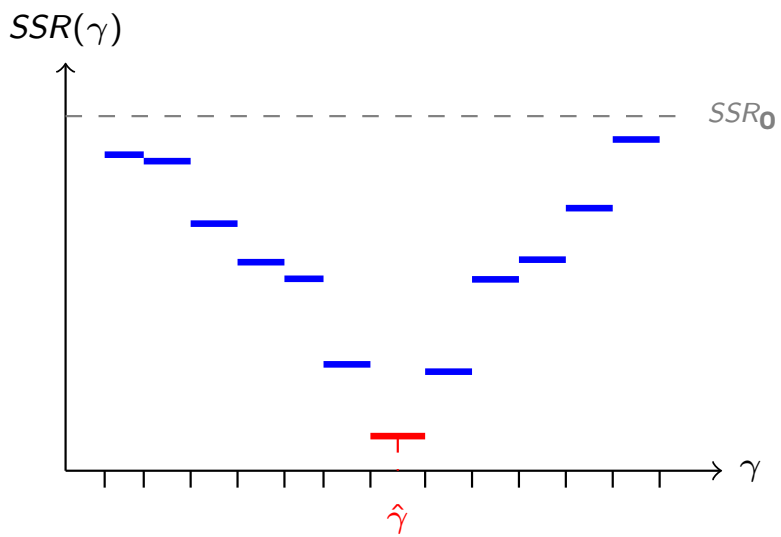
Regression Trees: Step 2 — Choose the Best Cut

For each candidate cutoff, compute the SSR of the two-group fit:

$$SSR(\gamma) = \sum_{i:\, X_{di} \le \gamma} (Y_i - \bar{Y}_L)^2 + \sum_{i:\, X_{di} > \gamma} (Y_i - \bar{Y}_R)^2$$

- A good cut leaves each group **internally homogeneous**: everyone is close to their group's mean.
- $SSR(\gamma)$ changes only when γ **crosses an observation**, so it is a step function.

↪ We only need to check the $n - 1$ **possibilities for** γ.



Regression Trees: The Splitting Algorithm

Minimizing $SSR(\gamma) \iff$ picks γ that makes the two groups' means **differ the most**

Regression Trees: The Splitting Algorithm

Minimizing $SSR(\gamma) \iff$ picks γ that makes the two groups' means **differ the most**

Full search for the first split:

Regression Trees: The Splitting Algorithm

Minimizing $SSR(\gamma) \iff$ picks γ that makes the two groups' means **differ the most**

Full search for the first split:

1. For each regressor $d = 1, \ldots, p$, and each candidate γ (midpoints of the sorted X_d), compute $SSR(\gamma)$.

Regression Trees: The Splitting Algorithm

Minimizing $SSR(\gamma) \iff$ picks γ that makes the two groups' means **differ the most**

Full search for the first split:

1. For each regressor $d = 1, \ldots, p$, and each candidate γ (midpoints of the sorted X_d), compute $SSR(\gamma)$.
2. Choose the pair $(\hat{d}, \hat{\gamma})$ with the **smallest SSR** over all variables and cuts.

Regression Trees: The Splitting Algorithm

Minimizing $SSR(\gamma) \iff$ picks γ that makes the two groups' means **differ the most**

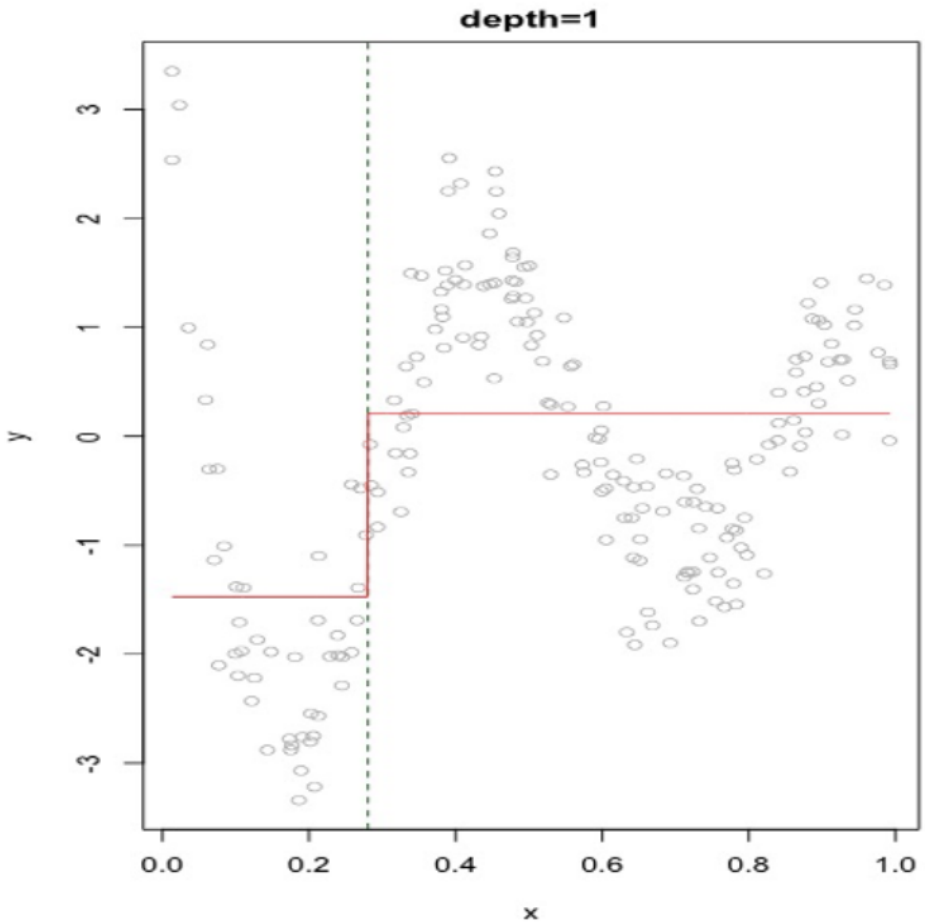
Full search for the first split:

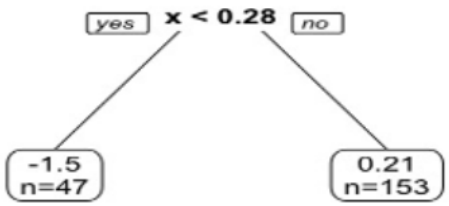
1. For each regressor $d = 1, \ldots, p$, and each candidate γ (midpoints of the sorted X_d), compute $SSR(\gamma)$.
2. Choose the pair $(\hat{d}, \hat{\gamma})$ with the **smallest SSR** over all variables and cuts.
3. Each new branch is now a smaller sample: **repeat inside it**. This is how the tree **grows**.

Regression Trees

ML finds the best split which minimizes the deviations to the mean in each leaf.

- This is a tree of depth 1:

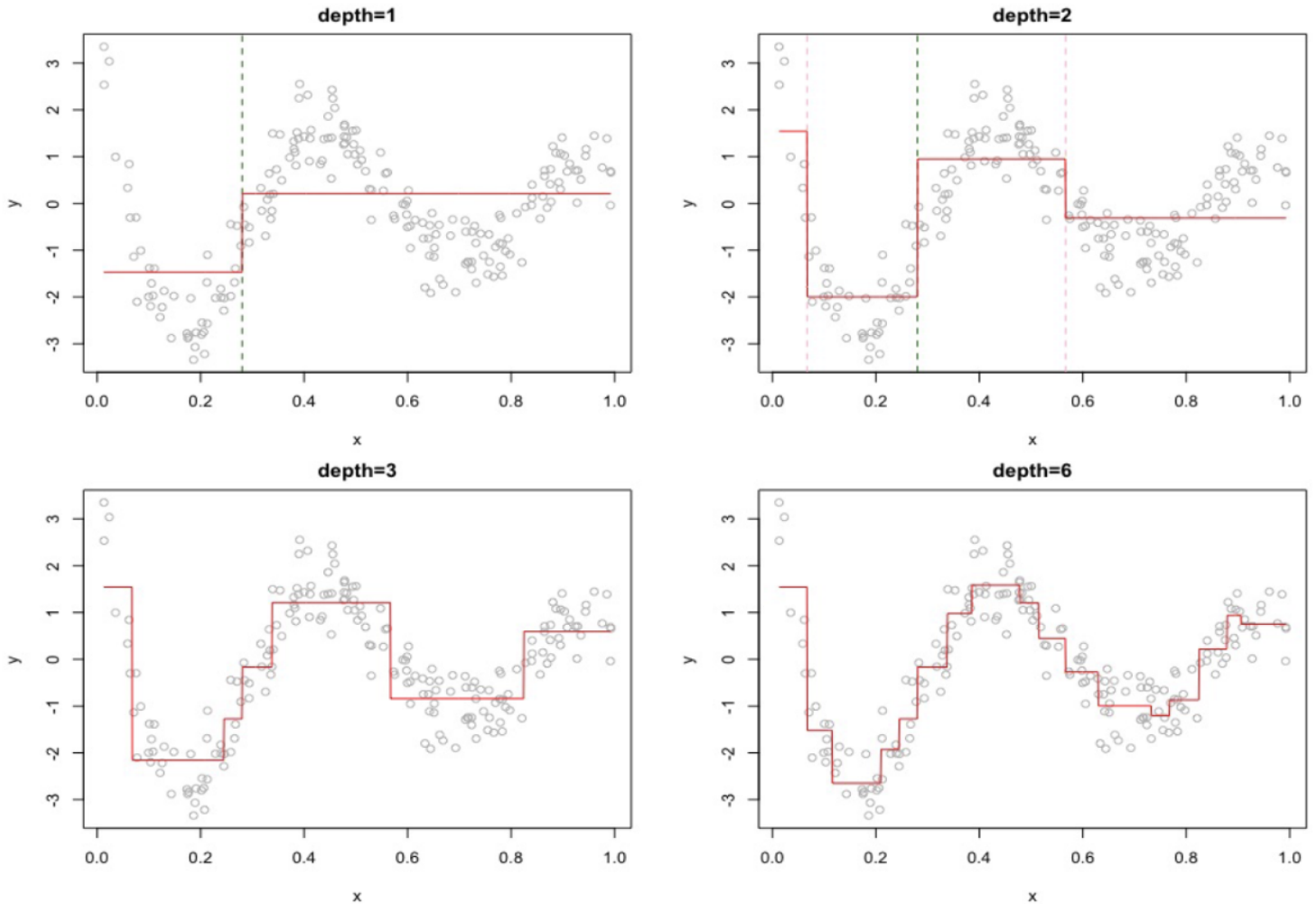




Regression Trees

ML finds the best split which minimizes the deviations to the mean in each leaf.

- These are trees with higher depths:



Pruning a Tree: Why and How Much

Problem: each split always lowers the SSR. A tree grown until it stops improving has tiny leaves that **fit noise** (overfitting).

Pruning a Tree: Why and How Much

Problem: each split always lowers the SSR. A tree grown until it stops improving has tiny leaves that **fit noise** (overfitting).

Fix: penalize complexity with a *Mallows-type* criterion:

$$C(\alpha) = \underbrace{\sum_{i=1}^{n} \hat{e}_i^2}_{\text{fit}} + \underbrace{\alpha N}_{\text{complexity}}$$

- N = number of leaves.
- α = price of one extra leaf, like a smoothing parameter λ.
- A split is kept only if it lowers the SSR by **more than** α.

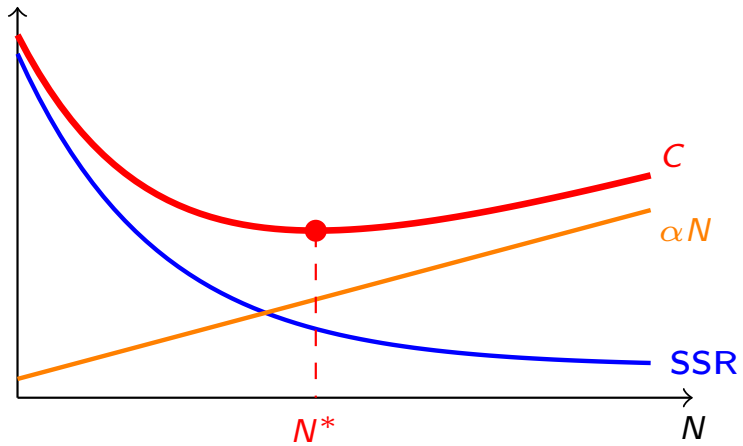
Pruning a Tree: Why and How Much

Problem: each split always lowers the SSR. A tree grown until it stops improving has tiny leaves that **fit noise** (overfitting).

Fix: penalize complexity with a *Mallows-type* criterion:

$$C(\alpha) = \underbrace{\sum_{i=1}^{n} \hat{e}_i^2}_{\text{fit}} + \underbrace{\alpha N}_{\text{complexity}}$$

- N = number of leaves.
- α = price of one extra leaf, like a smoothing parameter λ.
- A split is kept only if it lowers the SSR by **more than** α.



More leaves: better fit but a higher penalty. C is lowest in between.

Pruning a Tree: Backward Stepwise

Start from the **large tree** and work backward:

(a) Find the split whose removal (merging its two leaves) **lowers** C **the most**.

Pruning a Tree: Backward Stepwise

Start from the **large tree** and work backward:

(a) Find the split whose removal (merging its two leaves) **lowers** C **the most**.

(b) Prune it.

Pruning a Tree: Backward Stepwise

Start from the **large tree** and work backward:

(a) Find the split whose removal (merging its two leaves) **lowers** C **the most**.

(b) Prune it.

(c) Repeat until no removal lowers C.

Pruning a Tree: Backward Stepwise

Start from the **large tree** and work backward:

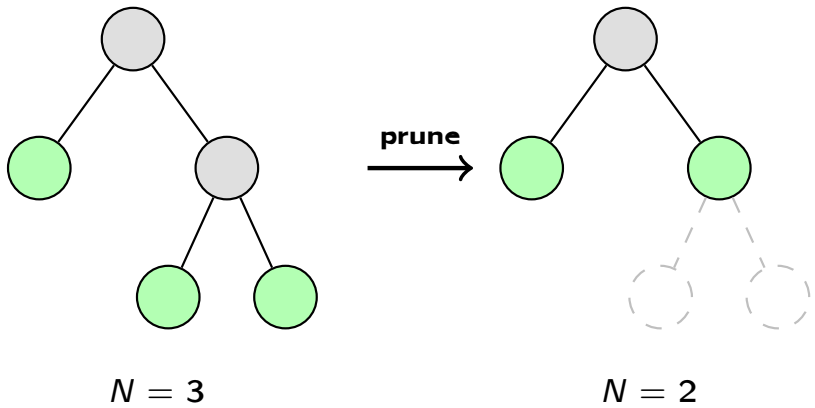
(a) Find the split whose removal (merging its two leaves) **lowers** C **the most**.

(b) Prune it.

(c) Repeat until no removal lowers C.

Choosing α**:** by K-fold cross-validation.

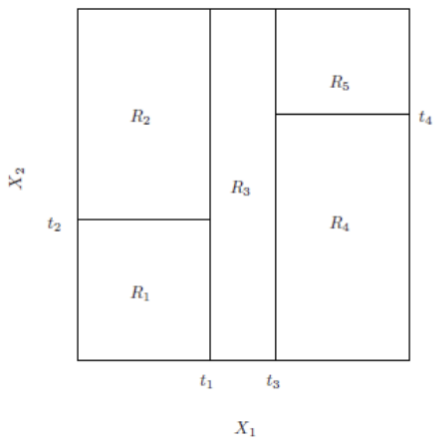
- α small: big tree, low bias, high variance.
- α large: small tree, more bias, low variance, and **easier to interpret**.

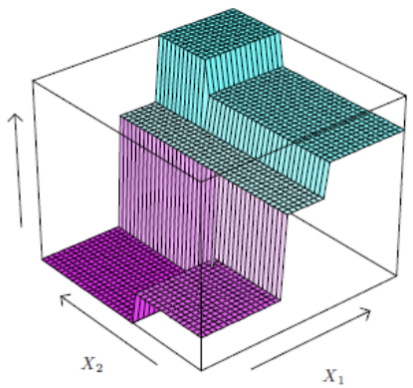


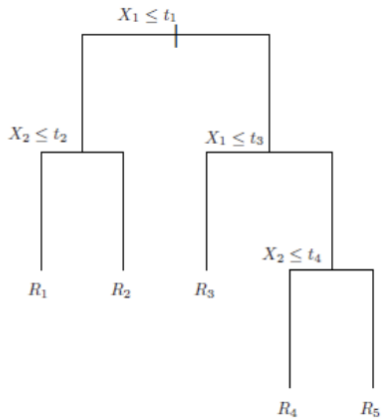
Keep the merge if the SSR rises by less than α.

Regression Trees

This is a tree with three covariates:







Disadvantages:

- The results are difficult to interpret as there are no regression coefficients.

Disadvantages:

- The results are difficult to interpret as there are no regression coefficients.
- Fitted regression $\hat{m}(x)$ is a discrete step function, a crude approximation when $m(x)$ is continuous and smooth.

Disadvantages:

- The results are difficult to interpret as there are no regression coefficients.
- Fitted regression $\hat{m}(x)$ is a discrete step function, a crude approximation when $m(x)$ is continuous and smooth.
- A regression tree may require a high number of leaves, which can result in a non-parsimonious model with high estimation variance.

Regression Tree Code

Tree vs. Linear Model

Depending on the problem at hand:

- **Linear regression:**

$$m(X) = \beta_0 + \sum_{j=1}^{K} X_j \beta_j$$

- **Regression tree:**

$$m(X) = \sum_{j=1}^{J} c_j 1(x \in R_j)$$

Tree vs. Linear Model

Depending on the problem at hand:

- **Linear regression:**

$$m(X) = \beta_0 + \sum_{j=1}^{K} X_j \beta_j$$

- **Regression tree:**

$$m(X) = \sum_{j=1}^{J} c_j 1(x \in R_j)$$

- If the relationship between y and $(x_1, \ldots, x_K)$ is **linear**, a linear model should perform better.

Tree vs. Linear Model

Depending on the problem at hand:

- **Linear regression:**

$$m(X) = \beta_0 + \sum_{j=1}^{K} X_j \beta_j$$

- **Regression tree:**

$$m(X) = \sum_{j=1}^{J} c_j 1(x \in R_j)$$

- If the relationship between y and $(x_1, \ldots, x_K)$ is **linear**, a linear model should perform better.
- If the relationship between y and $(x_1, \ldots, x_K)$ is **highly nonlinear and complex**, a tree model should perform better.

Outline

- Non-parametric Regression
- Series and Spline Regression
- Regression Trees
- Bagging and Random Forest

Bagging

Regression trees are flexible, but they are also **unstable**:

- A small change in the data can change the first split, and with it the whole tree.
- This means trees have **low bias** but **high variance**.

Bagging

Regression trees are flexible, but they are also **unstable**:

- A small change in the data can change the first split, and with it the whole tree.
- This means trees have **low bias** but **high variance**.

A natural way to reduce variance is to **average**: if we had many samples, we could fit one tree on each and average them.

Bagging

Regression trees are flexible, but they are also **unstable**:

- A small change in the data can change the first split, and with it the whole tree.
- This means trees have **low bias** but **high variance**.

A natural way to reduce variance is to **average**: if we had many samples, we could fit one tree on each and average them.

We only have one sample, but the **bootstrap** lets us create many "new" samples from it. This is the idea behind **bagging**.

Bagging

Bagging (**b**ootstrap **agg**regat**ing**) is a method to reduce the variance of a predictor.

Bagging

Bagging (**b**ootstrap **agg**regat**ing**) is a method to reduce the variance of a predictor.

(1) Generate a large number B of bootstrap samples (drawn with replacement from the original data).

Bagging

Bagging (**b**ootstrap **agg**regat**ing**) is a method to reduce the variance of a predictor.

(1) Generate a large number B of bootstrap samples (drawn with replacement from the original data).

(2) Estimate the regression tree on each bootstrap sample.

Bagging

Bagging (**b**ootstrap **agg**regat**ing**) is a method to reduce the variance of a predictor.

(1) Generate a large number B of bootstrap samples (drawn with replacement from the original data).

(2) Estimate the regression tree on each bootstrap sample.

(3) Take the average of all the predictions.

Bagging

Bagging (**b**ootstrap **agg**regat**ing**) is a method to reduce the variance of a predictor.

(1) Generate a large number B of bootstrap samples (drawn with replacement from the original data).

(2) Estimate the regression tree on each bootstrap sample.

(3) Take the average of all the predictions.

If $\hat{m}_b^*(x)$ is the tree estimated on bootstrap sample b, the **bagging estimator** of $m(x) = \mathbb{E}[Y \mid X = x]$ is:

Bagging

Bagging (**b**ootstrap **agg**regat**ing**) is a method to reduce the variance of a predictor.

(1) Generate a large number B of bootstrap samples (drawn with replacement from the original data).

(2) Estimate the regression tree on each bootstrap sample.

(3) Take the average of all the predictions.

If $\hat{m}_b^*(x)$ is the tree estimated on bootstrap sample b, the **bagging estimator** of $m(x) = \mathbb{E}[Y \mid X = x]$ is:

$$\hat{m}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{m}_b^*(x)$$

Bagging

Bagging (**b**ootstrap **agg**regat**ing**) is a method to reduce the variance of a predictor.

(1) Generate a large number B of bootstrap samples (drawn with replacement from the original data).

(2) Estimate the regression tree on each bootstrap sample.

(3) Take the average of all the predictions.

If $\hat{m}_b^*(x)$ is the tree estimated on bootstrap sample b, the **bagging estimator** of $m(x) = \mathbb{E}[Y \mid X = x]$ is:

$$\hat{m}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{m}_b^*(x)$$

- The more bootstrap samples we use, the closer we get to the **ideal bagging estimator** $\mathbb{E}^*[\hat{m}^*(x)]$.
- Using a large B does not cause overfitting. It only takes more time.

Bagging

When is bagging useful?

- When the estimator has low bias but high variance.
- With hard-threshold estimators such as regression trees, where small changes in the data move the splits.

Bagging

When is bagging useful?

- When the estimator has low bias but high variance.
- With hard-threshold estimators such as regression trees, where small changes in the data move the splits.

Out-of-bag (OOB) observations:

- Because we sample with replacement, some observations appear more than once and some never appear.
- Each tree uses around **2/3** of the observations.
- The remaining **1/3** are the out-of-bag observations: that tree never saw them.

Bagging

When is bagging useful?

- When the estimator has low bias but high variance.
- With hard-threshold estimators such as regression trees, where small changes in the data move the splits.

Out-of-bag (OOB) observations:

- Because we sample with replacement, some observations appear more than once and some never appear.
- Each tree uses around **2/3** of the observations.
- The remaining **1/3** are the out-of-bag observations: that tree never saw them.
- We can use them to predict and to compute the mean squared error (MSE). This works like a test sample, without needing extra data.

Bagging

Why does it work?

- Each tree splits the data differently, depending on which points were resampled.
- Some trees split earlier, others later. Each one makes its own random mistakes.
- When we average them, these mistakes **cancel out**, leaving a smoother and more stable prediction.

Bagging

Why does it work?

- Each tree splits the data differently, depending on which points were resampled.
- Some trees split earlier, others later. Each one makes its own random mistakes.
- When we average them, these mistakes **cancel out**, leaving a smoother and more stable prediction.

But bagging may exaggerate bias:

- Averaging smooths out extremes. What if those extremes are real and important?
- Predictions get pulled toward the center, which can mean *more* bias.
- The trees are similar to each other, since they share most of the data, so averaging cannot remove all the variance. **Random forests** address this.

Code

Random Forests

- Bootstrap samples are similar to one another, hence the estimated bootstrap regression trees will also be similar.
  - e.g. Bagging will create many trees where the first splits will be given by the main predictor.
- This causes collinearity across trees.
- Variance of the bootstrap average remains high even when the number of bootstrap replications B is large.

Random Forests

- Bootstrap samples are similar to one another, hence the estimated bootstrap regression trees will also be similar.
  - e.g. Bagging will create many trees where the first splits will be given by the main predictor.
- This causes collinearity across trees.
- Variance of the bootstrap average remains high even when the number of bootstrap replications B is large.
- Random forests decorrelate the bootstrap regression trees by adding randomness, reducing MSE.
- Random forests have effectively displaced simple regression trees.

Algorithm

Step 1 — Choose tuning parameters

- B: number of trees
- $m < p$: number of regressors tried at each split (default $m = p/3$)
- N_{min}: minimum leaf size (default 5)
- $\alpha \in [0, 1)$: minimum share of observations each branch must keep

Algorithm

Step 1 — Choose tuning parameters

- B: number of trees
- $m < p$: number of regressors tried at each split (default $m = p/3$)
- N_{min}: minimum leaf size (default 5)
- $\alpha \in [0, 1)$: minimum share of observations each branch must keep

Step 2 — Grow B trees. For $b = 1, \ldots, B$:

- Draw a bootstrap sample of size n.
- At each node, pick m of the p regressors **at random**.
- Split on the best of these m, provided each branch keeps at least N_{min} observations and a fraction α of the node.
- Keep splitting until each leaf has between N_{min} and $2N_{\text{min}} - 1$ observations.

Algorithm

Step 1 — Choose tuning parameters

- B: number of trees
- $m < p$: number of regressors tried at each split (default $m = p/3$)
- N_{min}: minimum leaf size (default 5)
- $\alpha \in [0, 1)$: minimum share of observations each branch must keep

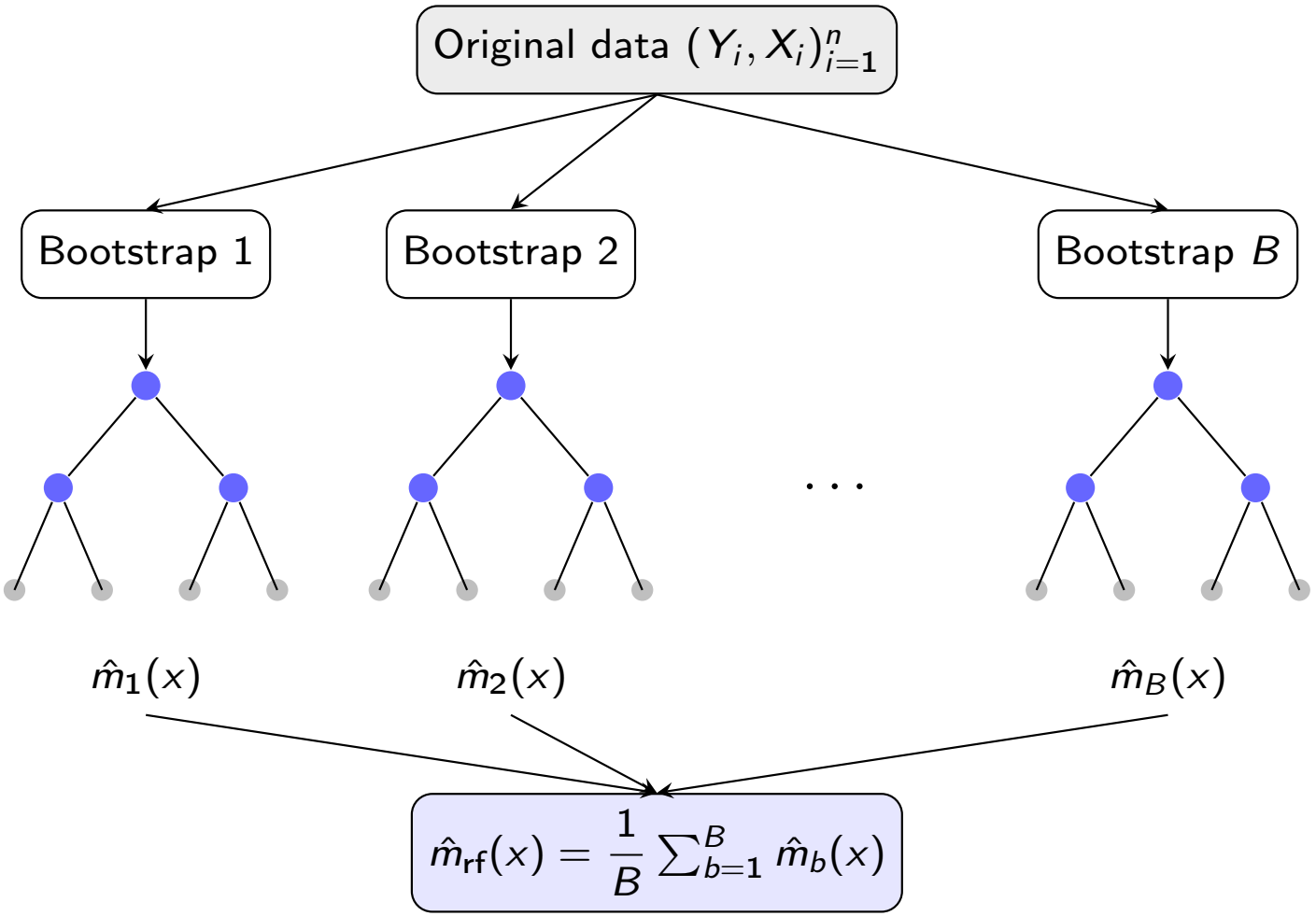
Step 2 — Grow B trees. For $b = 1, \ldots, B$:

- Draw a bootstrap sample of size n.
- At each node, pick m of the p regressors **at random**.
- Split on the best of these m, provided each branch keeps at least N_{min} observations and a fraction α of the node.
- Keep splitting until each leaf has between N_{min} and $2N_{\text{min}} - 1$ observations.

Step 3 — Average. Let $\hat{m}_b(x)$ be the mean of Y in the leaf of tree b that contains x. Then

$$\hat{m}_{\text{rf}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{m}_b(x).$$

Random Forests: the picture



split on best of m random regressors

leaf: mean of Y

Random Forests

Using randomization to reduce the number of variables from p to m at each step alters the tree structure.

- Reduces the correlation between the bootstrapped regression trees.
- While random forests are popular in applications, **a distributional theory has been slow to develop**.

Note: ***Honest trees****, developed by Wager and Athey (2019) — beyond the scope of this course — suggest viable properties of asymptotic normality.*

Random Forest Code

"Real life" application

Random Forests in Practice: Examples

Prediction problems (we care about $\hat{Y}$, not $\hat{\beta}$):

- **House prices.** Mullainathan & Spiess (2017, *JEP*): predicting home values from the American Housing Survey; tree ensembles beat OLS **out of sample**.
- **Poverty targeting.** McBride & Nichols (2018, *WBER*): (quantile) random forests improve proxy-means tests that decide who receives social transfers.
- **Credit and default risk.** Banks predict loan default from many mixed continuous/discrete borrower traits — exactly where trees shine.
- **Program evaluation diagnostics.** PETC (previous slide): variable importance points to mechanisms.

Causal questions (forests as a tool, not the identification):

- **Heterogeneous effects.** Davis & Heller (2017, *AER P&P*): *causal forests* find which youths benefit most from a Chicago summer-jobs program (randomized).

Common thread: many regressors, unknown nonlinearities and interactions, and an out-of-sample criterion to judge the fit.

Random Forest in Stata

Command `rforest` (Schonlau & Zou, 2020, *Stata Journal*). Mapping to our algorithm: `iterations` $= B$, `numvars` $= m$, `lsize` $= N_{\text{min}}$.

```
ssc install rforest
set seed 2026
gen train = runiform() < 0.5                    // 50/50 train-test split

* Benchmark: OLS with quadratic experience
reg lwage c.exper##c.exper educ female informal kids if train
predict yhat_ols if !train

* Random forest: B = 500 trees, m = 2 of p = 5 regressors, N_min = 5
rforest lwage exper educ female informal kids if train, ///
    type(reg) iterations(500) numvars(2) lsize(5)
display e(OOB_Error)                            // out-of-bag error
predict yhat_rf if !train
display e(RMSE)                                 // test-sample RMSE

* Compare test MSE
gen se_ols = (lwage - yhat_ols)^2
gen se_rf  = (lwage - yhat_rf)^2
summarize se_ols se_rf if !train
```