Bootstrap and Splines

EN5422/EV4238 | Fall 2023

w03_bootstrap_splines.pdf

(Week 3 – 1/2)

Contents

1 Introduction to the Bootstrap

1.1 Uncertainty in a test statistic

There is often interested to understand the uncertainty in the estimated value of a test statistic.

- For example, let *p* be the actual/true proportion of customers who will use your company's coupon.
- To estimate *p*, you decide to take a sample of $n = 200$ customers and find that $x = 10$ or $\hat{p} = \frac{10}{200} = 0.05 = 5\%$ redeemed the coupon.

1.1.1 Uncertainty in a test statistic

- It is common to calculate the 95% *confidence interval* (*CI*)

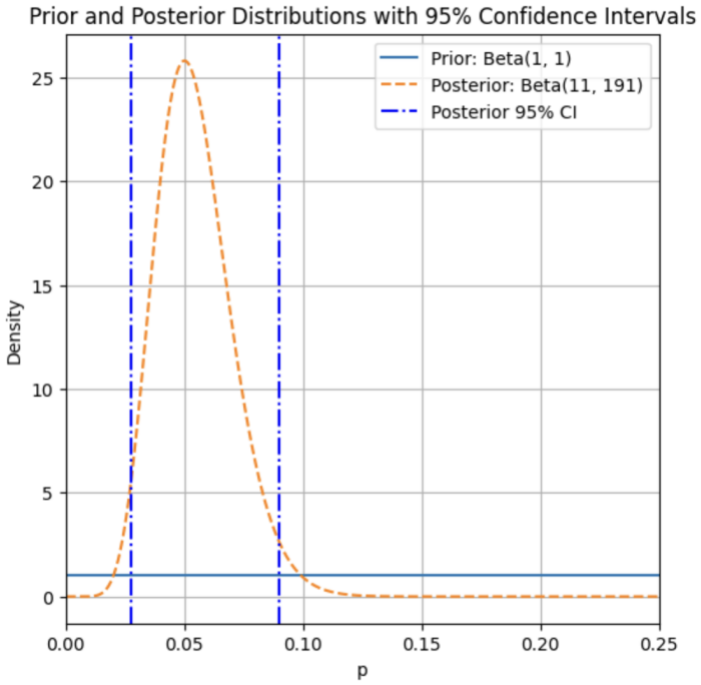
$$\begin{aligned} \mathrm{CI}(p) &= \hat{p} \pm 2 \cdot \mathrm{SE}(\hat{p}) \\ &= \hat{p} \pm 2\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \\ &= 0.05 \pm 0.03 \end{aligned}$$

  where SE is *standard error*.
- This calculation is based on the assumption that $\hat{p}$ is approximately normally distributed with the mean equal to the *unknown* true *p*, i.e., $\hat{p} \sim N(p, \sqrt{\frac{p(1-p)}{n}})$.

  Note: Binomial distribution's variance is $\frac{p(1-p)}{n}$.

1.1.2 Bayesian Posterior Distribution

In the Bayesian world, you'd probably specify a Beta *prior* for *p*, i.e., $p \sim \mathrm{Beta}(a,b)$ and calculate the *posterior* distribution $p \mid x = 10 \sim \mathrm{Beta}(a+x, b+n-x)$ which would fully characterize the uncertainty.



Note

The beta distribution is a continuous probability distribution that is defined on the interval [0, 1]. It's often used to model random variables that represent proportions or probabilities. The distribution has two shape parameters, usually denoted as α (alpha) and β (beta), which determine its shape. One common use case of the beta distribution is in Bayesian statistics, where it's used as a conjugate prior for modeling the distribution of a probability parameter in binomial or Bernoulli trials. In simpler terms, it's used to update beliefs about the probability of success or failure as more data is collected.

Example: Coin Toss Probability

Let's say you have a biased coin, and you want to estimate the probability (p) of getting heads in a single toss. You believe that the coin might be biased towards heads, but you're not sure how much. You decide to model your beliefs using a beta distribution.

The beta distribution's shape parameters α and β can be interpreted as follows:

- α: Represents the number of times you've observed heads.
- β: Represents the number of times you've observed tails.

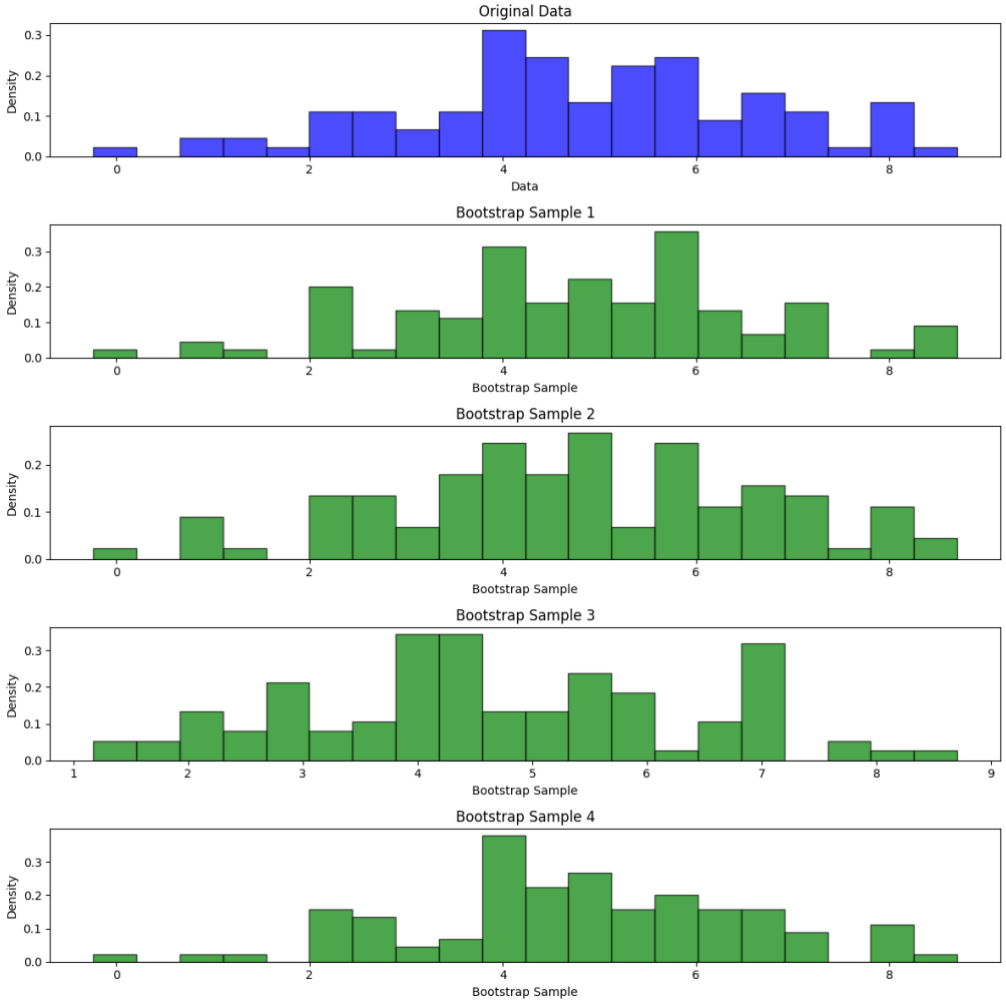
Initially, you might have no information, so you set $\alpha = \beta = 1$, which corresponds to a uniform distribution on [0, 1] (a flat curve).

As you start flipping the coin and recording the outcomes, you update α and β accordingly. For example, after 10 tosses with 7 heads and 3 tails, you would update $\alpha = 1 + 7 = 8$ and $\beta = 1 + 3 = 4$. Now, the beta distribution's shape reflects your updated beliefs about the coin's bias. The mode (peak) of the distribution would be around 0.7, indicating that you believe there's a higher probability of getting heads.

As you gather more data, the beta distribution becomes more peaked around the "true" underlying probability of heads based on your observations. This example demonstrates how the beta distribution can serve as a flexible way to model uncertainty in probabilities, allowing you to update your beliefs as you collect more data. Remember, the beta distribution is not limited to coin tosses; it's a versatile distribution that can be used in various scenarios where you're modeling probabilities or proportions.

1.1.3 The Bootstrap

- The Bootstrap is a way to assess the uncertainty in a test statistic using *resampling*.
- The idea is to simulate the data from the *empirical distribution*, which puts a point mass of 1/*n* at each observed data point (i.e., sample the original data **with replacement**).
    - It is important to simulate *n* observations (same size as original data) because the uncertainty in the test statistic is a function of *n*.



- Then, calculate the test statistic for each bootstrap sample. The variability in the collection of bootstrap test statistics should be similar to the variability in the test statistic.

Algorithm: Nonparametric/Empirical Bootstrap

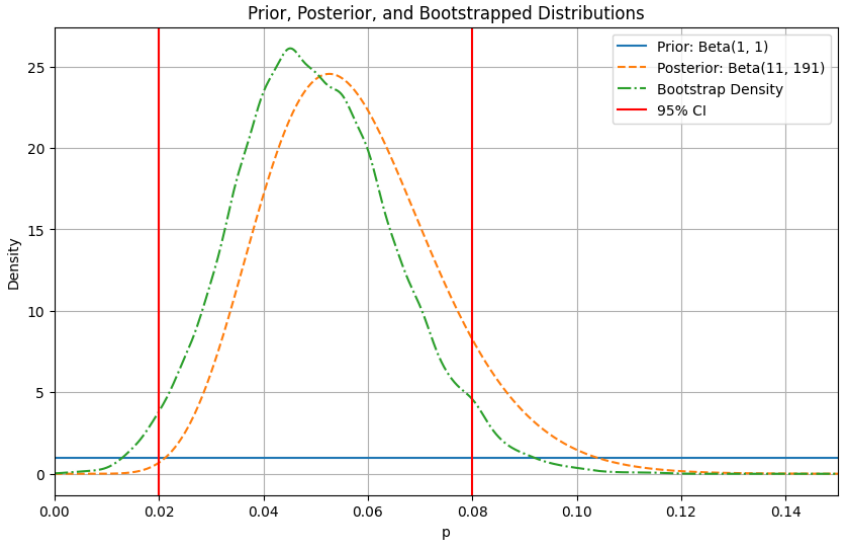
Observe data $D = [X_1, X_2, \ldots, X_n]$ (n observations).
Calculate a test statistic $\hat{\theta} = \hat{\theta}(D)$, which is a function of D.
Repeat steps 1 and 2 M times:

1. Simulate D^*, a new data set of n observations by sampling from D with replacement.
2. Calculate the bootstrap test statistic $\hat{\theta}^* = \hat{\theta}(D^*)$

The bootstrapped samples $\hat{\theta}_1^*, \hat{\theta}_2^*, \hat{\theta}_3^*, \ldots, \hat{\theta}_M^*$ can be used to estimate the distribution of $\hat{\theta}$.

- Or properties of the distribution, like standard deviation (standard error), percentile, etc.

```
# Original Data
x = np.concatenate([np.ones(10), np.zeros(190)])  # 10 successes, 190
failures
n = len(x)

# Bootstrap Distribution
M = 5000  # number of bootstrap samples
p = np.zeros(M)  # initialize vector for test statistic
np.random.seed(202309)  # set random seed

for m in range(M):
    xboot = np.random.choice(x, size=n, replace=True)  # bootstrap sample
    p[m] = np.mean(xboot)  # calculate test statistic (proportion of
successes)

# Bootstrap Percentile based confidence Intervals
lower, upper = np.percentile(p, [2.5, 97.5])
print(f"95% bootstrap interval: ({lower:.3f}, {upper:.3f})")

# Plotting
p_values = np.linspace(0, 0.25, 1000)
prior = beta.pdf(p_values, 1, 1)
posterior = beta.pdf(p_values, 1 + 10, 1 + 190 - 10)

# Bootstrapped density
density = gaussian_kde(p)
bootstrap_density = density(p_values)
```

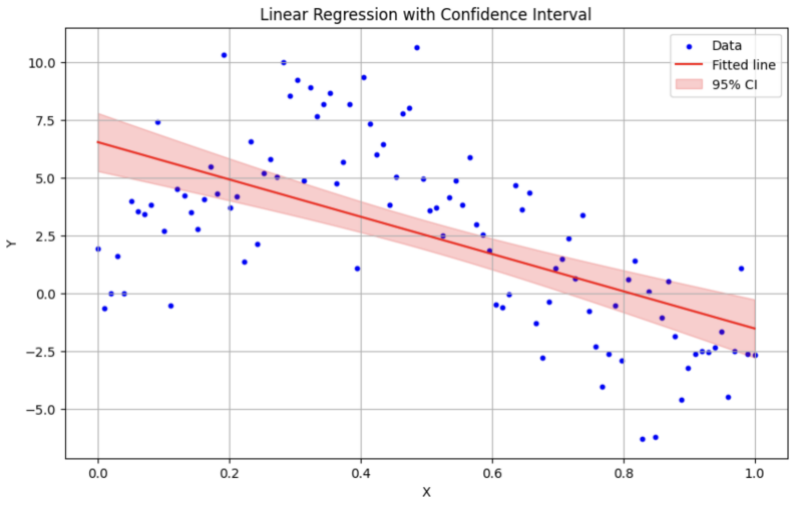


Note

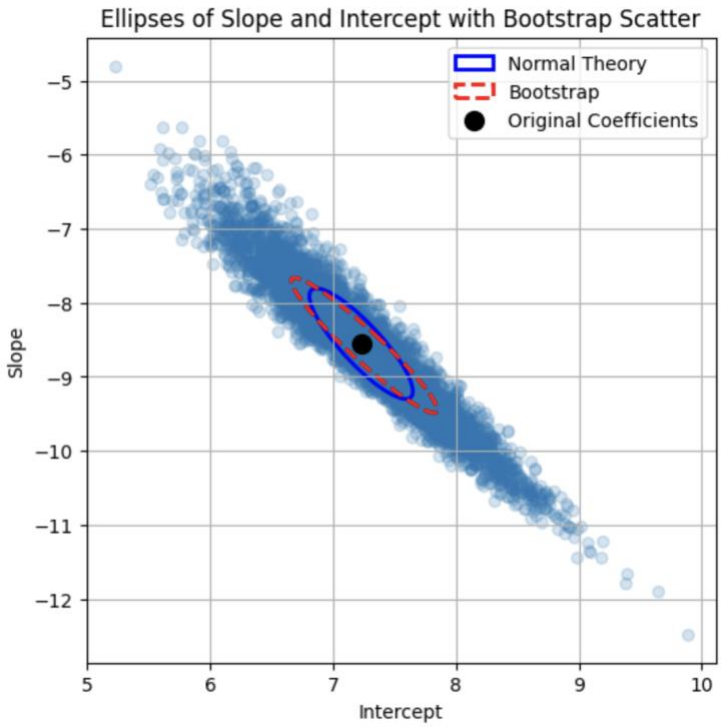
- Notice that in the above example the bootstrap distribution is close to the Bayesian posterior distribution (using the uninformative Uniform prior).
- This is no accident, it turns out there is a close correspondence between the bootstrap derived distribution and the Bayesian posterior distribution under *uninformative priors*.
    - See ESL 8.4 for more details.

2 Bootstrapping Regression Parameters

The bootstrap is not limited to univariate test statistics. It can be used on multivariate test statistics. Consider the uncertainty in estimates of the parameters (i.e., β coefficients) of a regression model.



2.1 Bootstrapping Regression Parameters



3 Non-linear Modeling via Basis Expansion

For a univariate *x*, a linear basis expansion is:

$$\hat{f}(x) = \sum_j \hat{\theta}_j b_j(x)$$

where $b_j(x)$ is the value of the j^{th} basis function at *x* and θ_j is the coefficient to be estimated.

- The $b_j(x)$ are sometimes specified in advanced (i.e., not estimated). But other approaches use sample data to estimate (e.g., using quantiles for knot placement).
    - Just be sure to estimate everything from the training data so there is no data leakage!

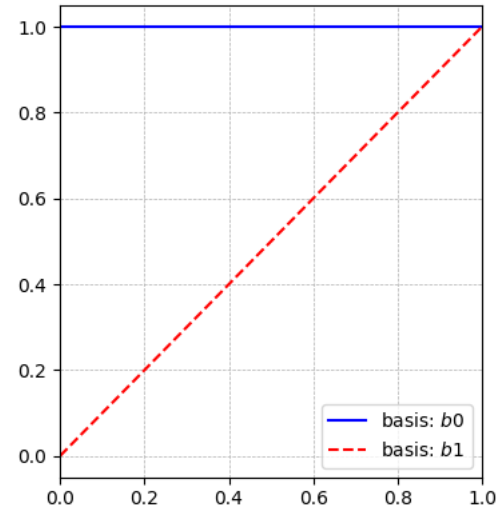
Note: Basis expansion allows linear models to capture non-linear relationships between predictors and the response variable. By transforming features, a linear model can fit to a non-linear curve in the data.

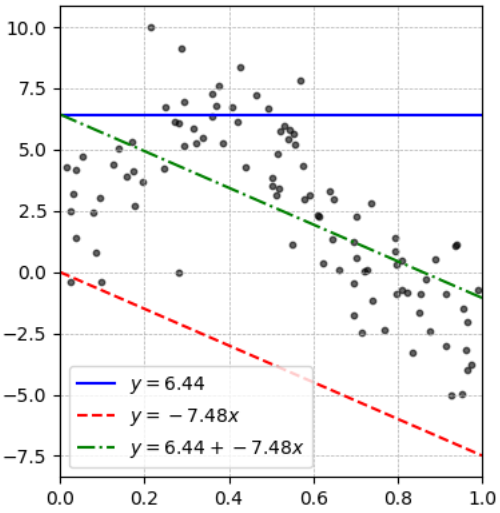
Example:

- **Linear Regression**

$$\hat{f}(x) = \hat{\beta}_0 + \hat{\beta}_1 x$$

$$b_0(x) = 1$$
$$b_1(x) = x$$

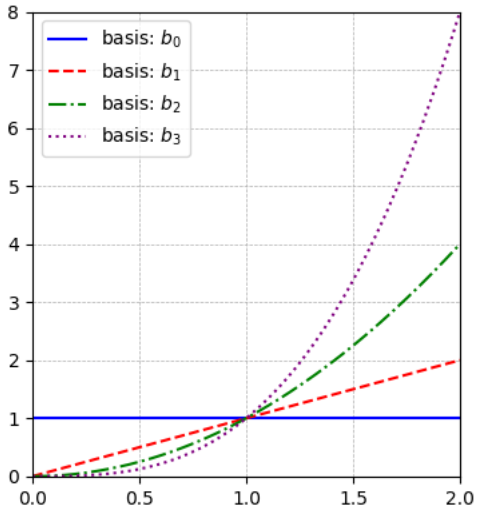


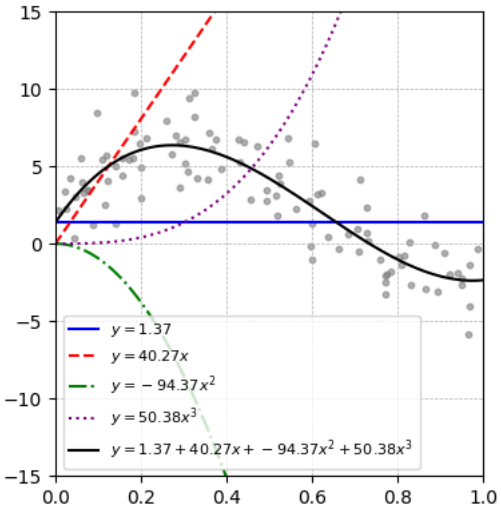


- **Polynomial Regression**

$$\hat{f}(x) = \sum_{j=1}^{d} \hat{\beta}_j x^j$$

$$b_j(x) = x^j$$

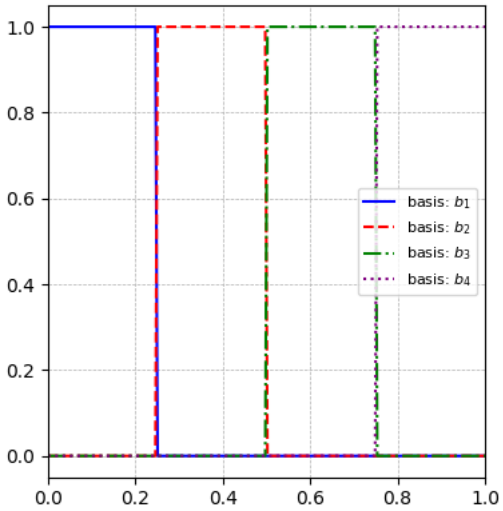


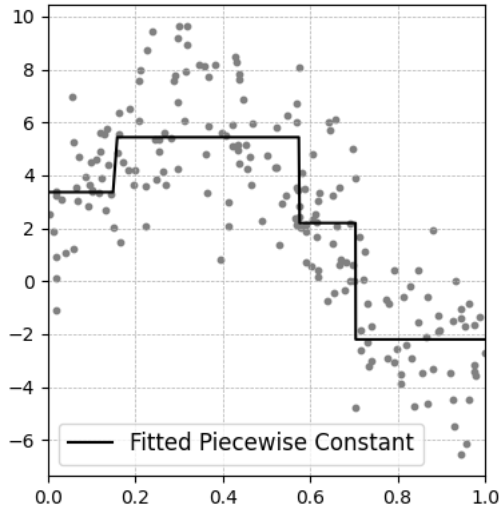


• Piecewise Constant Regression (Regressogram)

$$\hat{f}(x) = \sum_{j=1}^{p} \hat{\beta}_j \mathbb{1}(x \in R_j)$$

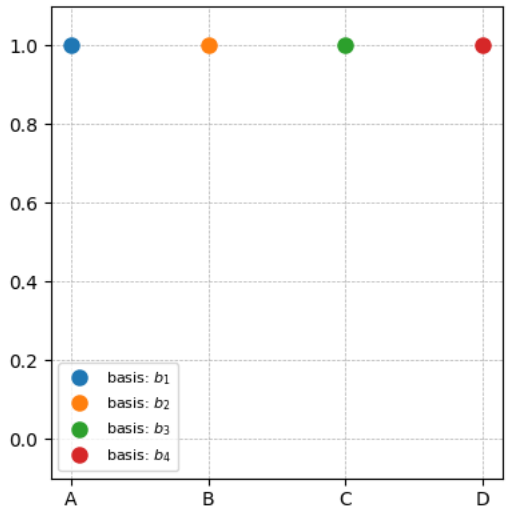
$$b_1(x) = \mathbb{1}(x \in R_1)$$
$$b_2(x) = \mathbb{1}(x \in R_2)$$
$$\vdots$$
$$b_p(x) = \mathbb{1}(x \in R_p)$$

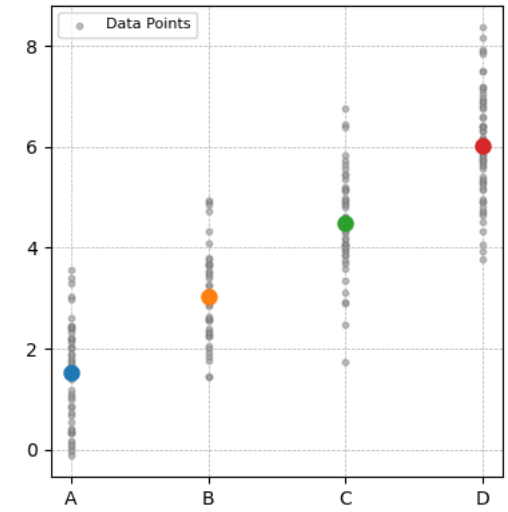




• Categorical Encoding (dummy, one-hot)

$$x \in \{c_1, c_2, \ldots, c_p\}$$
$$\hat{f}(x) = \sum_{j=1}^{p} \hat{\beta}_j \mathbb{1}(x = c_j) \text{ one-hot}$$
$$= \hat{\beta}_0 + \sum_{j=2}^{p} \hat{\beta}_j \mathbb{1}(x = c_j) \text{ dummy}$$
$$b_1(x) = \mathbb{1}(x = c_1)$$
$$b_2(x) = \mathbb{1}(x = c_2)$$
$$\vdots$$
$$b_p(x) = \mathbb{1}(x = c_p)$$





3.1 Piecewise Polynomials

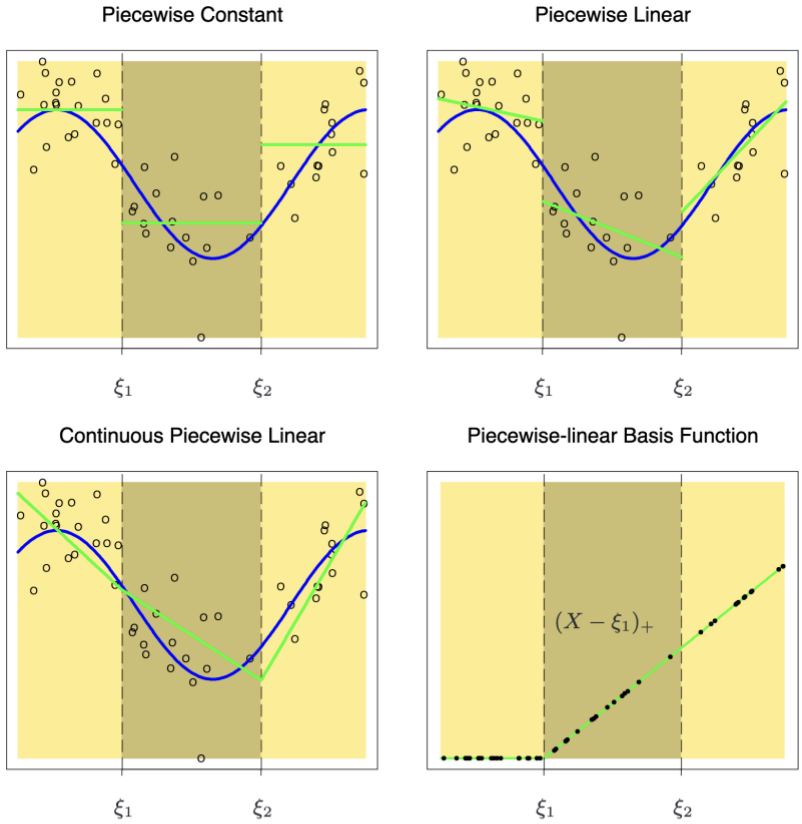


FIGURE 5.1. *The top left panel shows a piecewise constant function fit to some artificial data. The broken vertical lines indicate the positions of the two* knots *ξ_1 and ξ_2. The blue curve represents the true function, from which the data were generated with Gaussian noise. The remaining two panels show piecewise linear functions fit to the same data—the top right unrestricted, and the lower left restricted to be continuous at the knots. The lower right panel shows a piecewise–linear basis function, $h_3(X) = (X - \xi_1)_+$, continuous at ξ_1. The black points indicate the sample evaluations $h_3(x_i)$, $i = 1, \ldots, N$.*

3.2 B-Splines

- A degree = 0 B-spline is a *regressogram* basis. Will lead to a piecewise constant fit.
- A degree = 3 B-spline (called *cubic* spline) is similar in shape to a Gaussian pdf. But the B-spline has finite support and facilitates quick computation (due to the induced sparseness).

3.2.1 Parameter Estimation

$$\hat{f}(x) = \sum_j \hat{\theta}_j b_j(x)$$

In matrix notation,

$$\hat{f}(x) = B\hat{\theta}$$

where B is the *basis matrix* and X is the model matrix.

- For example, a polynomial matrix is:

$$B = \begin{bmatrix} 1 & X_1 & X_1^2 & \dots & X_1^J \\ 1 & X_2 & X_2^2 & \dots & X_2^J \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & X_n & X_n^2 & \dots & X_n^J \end{bmatrix}$$

- More generally,

$$B = \begin{bmatrix} b_1(x_1) & b_2(x_1) & b_3(x_1) & \dots & b_J(x_1) \\ b_1(x_2) & b_2(x_2) & b_3(x_2) & \dots & b_J(x_2) \\ \vdots & \vdots & \vdots & \vdots & \\ b_1(x_n) & b_2(x_n) & b_3(x_n) & \dots & b_J(x_n) \end{bmatrix}$$

- Now, it is in a form just like linear regression! Estimate with OLS:

$$\hat{\theta} = (B^T B)^{-1} B^T Y$$

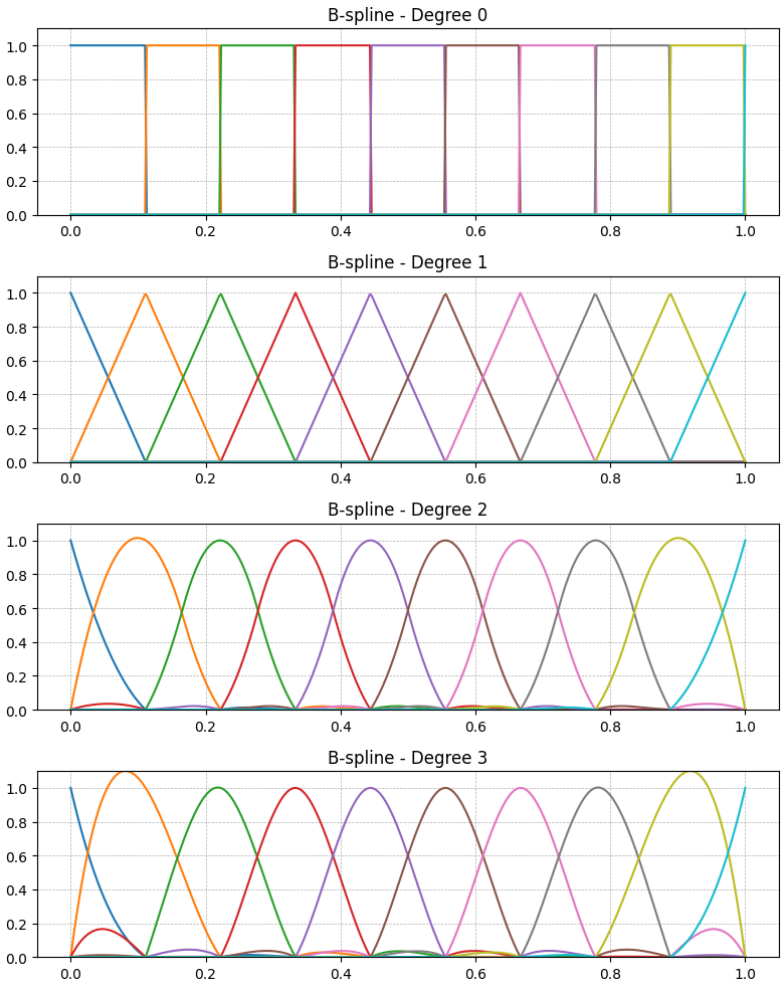
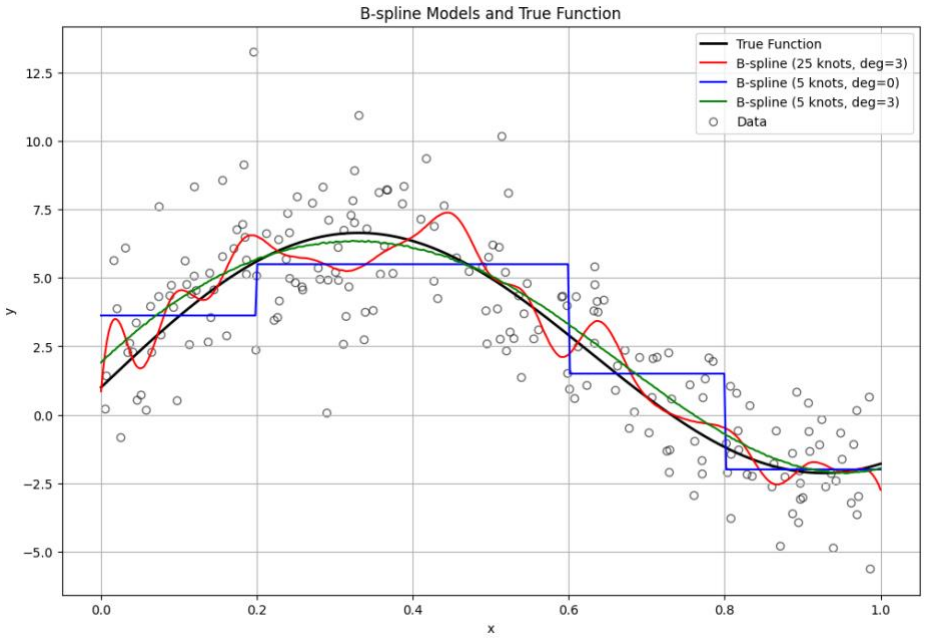


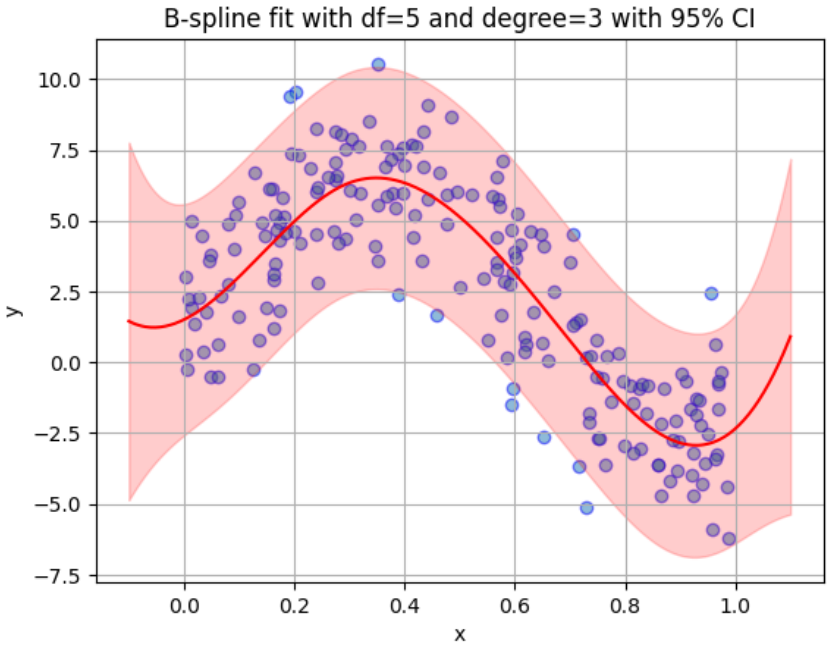
Figure 1: Like ESL Fig 5.20 (p188), B-splines (knots shown by vertical dashed lines)



- It may be helpful to think of a basis expansion as similar to a dummy coding for categorical variables.
    - This expands the single variable *x* into *df* variables.
- In Python, `statsmodel` can put directly in formula to make a B-spline.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
from patsy import dmatrix

# Assuming sim_x and sim_y functions have been defined and data is generated:
n = 200
x = sim_x(n)
y = sim_y(x, 2)
data_train = pd.DataFrame({'x': x.flatten(), 'y': y.flatten()})

# Create B-spline basis matrix
X_bs = dmatrix("bs(data_train.x, df=5, degree=3, include_intercept=False, "
               "lower_bound=-0.2, upper_bound=1.2)", {"data_train.x":
data_train.x})

# Fit the model
model_bs = sm.OLS(data_train.y, X_bs).fit()

# Plot the data
plt.scatter(data_train.x, data_train.y, edgecolor='blue', alpha=0.5)

# Plot the B-spline fit and its 95% confidence interval
x_range = np.linspace(-0.1, 1.1, 400)
X_range_bs = dmatrix("bs(x_range, df=5, degree=3, include_intercept=False, "
                     "lower_bound=-0.2, upper_bound=1.2)", {"x_range":
x_range})
```

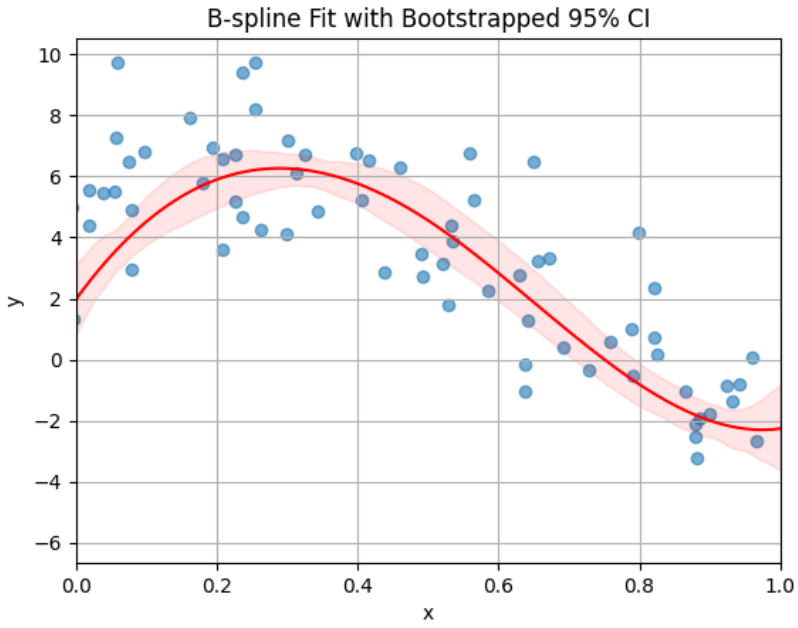
```
# Prediction with confidence interval
predictions = model_bs.get_prediction(X_range_bs)
pred_df = predictions.summary_frame(alpha=0.05)

# Extract upper and lower confidence limits
lower_bounds = pred_df['obs_ci_lower']
upper_bounds = pred_df['obs_ci_upper']

plt.plot(x_range, pred_df['mean'], color='red')
plt.fill_between(x_range, lower_bounds, upper_bounds, color='red', alpha=0.2)

plt.xlabel('x')
plt.ylabel('y')
plt.grid()
plt.title('B-spline fit with df=5 and degree=3 with 95% CI')
plt.show()
```



- Setting `df=5` will create a B-spline design matrix with 5 columns.
  - So, there are 5 basis functions

3.3 Bootstrap Confidence Interval for $f(x)$

Bootstrapping can be used to understand the uncertainty in the fitted values:



4 More Bagging

4.1 Out-of-Bag Samples

Your Turn #1: Observations not in bootstrap sample

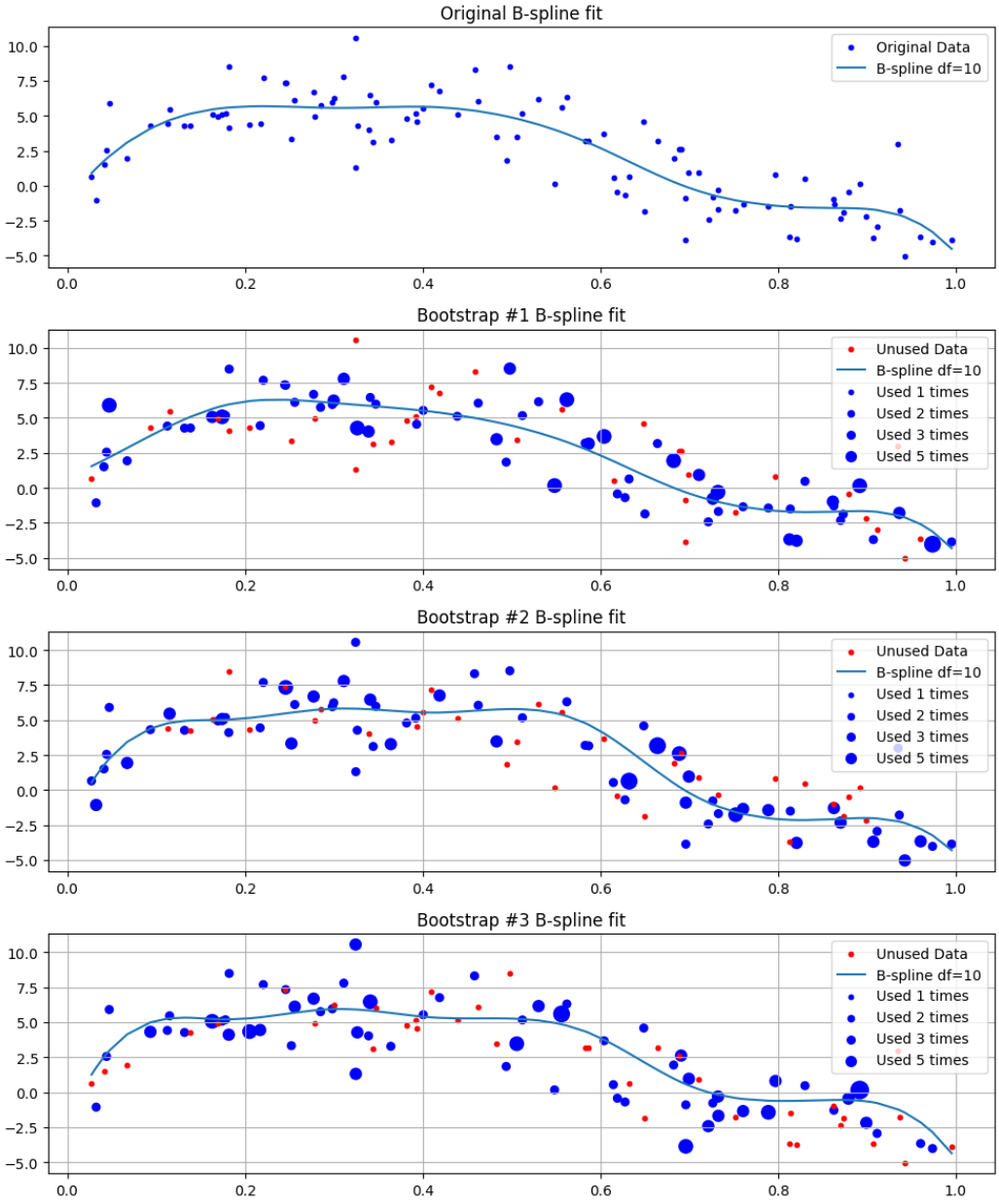
What is the expected number of observations that will *not* be in a bootstrap sample? Suppose n observations. When using bootstrap resampling, each observation has an equal probability $1/n$ of being selected for the sample on any given draw. The probability that an observation is not selected in a single draw is $(1-1/n)^n$. If we perform n draws (i.e., we sample with replacement n times to generate a bootstrap sample of size n), the probability that a specific observation is not selected in any of the draws is $(1-1/n)^n$. Thus, the expected number of distinct observations that are not included in the bootstrap sample is: $n\times(1-1/n)^n$. As n grows larger, $(1-1/n)^n$ approaches 1/e (where 1/ is the base of the natural logarithm, approximately equal to 2.71828). So, for large n:

Expected number of observations not in the bootstrap sample$\approx n\times 1e\approx 0.368\times n$

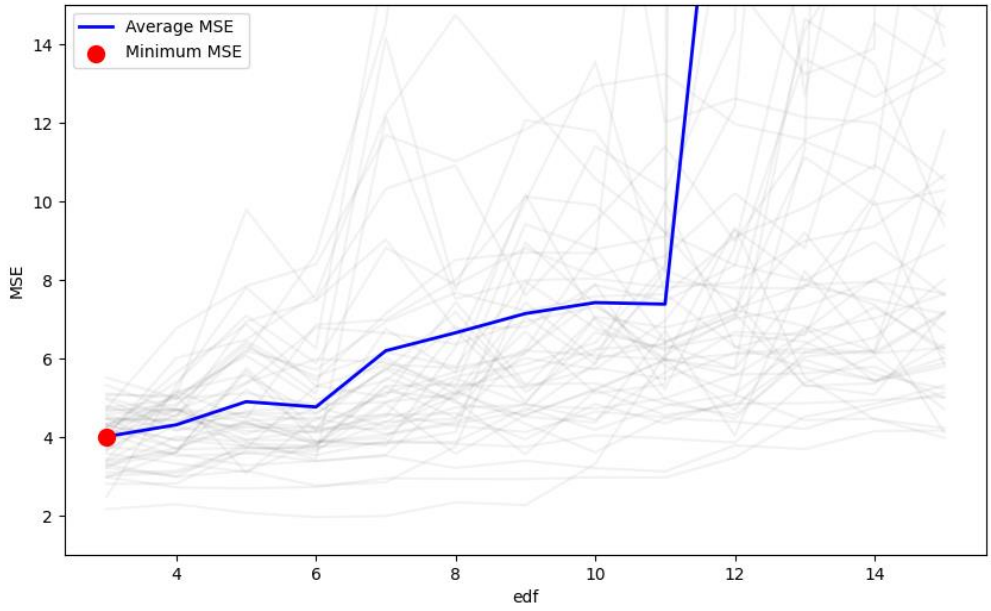
This implies that about 36.8% of the original observations are expected to be missing from a bootstrap sample, regardless of how large n is.

Let's look at a few bootstrap fits:

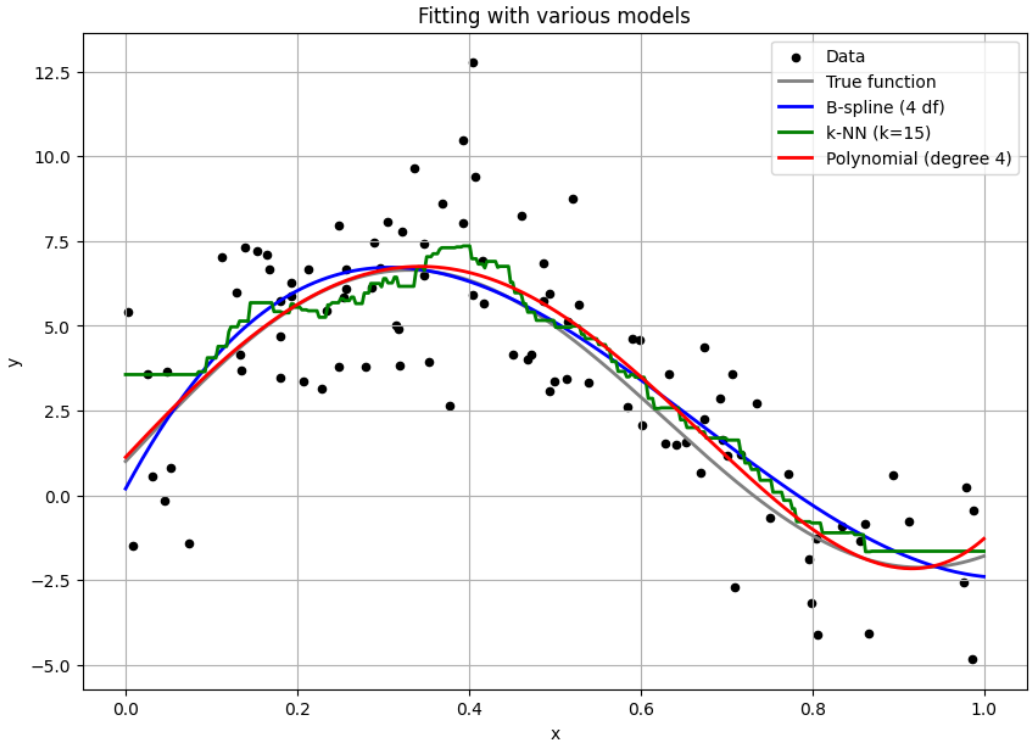
B-Spline with *df*=10
Red dots are out-of-bag



- Notice that each bootstrap sample does not include about 37% of the original observations.
- These are called *out-of-bag* (OOB) samples and can be used to assess model fit.
    - The OOB observations were not used to estimate the model parameters, so will be sensitive to over/under fitting.
- Below, we evaluate the OOB error over the spline complex (*df* = number of estimated coefficients).



- The minimum OOB error occurs at *df* =4. This matches the optimal complexity in a polynomial fit from the previous lecture nots.



4.2 Number of Bootstrap Simulations

Hesterberg recommends using $M \geq 15{,}000$ (number of bootstrap samples) for real applications to remove most of the Monte Carlo variability. For the examples in class, I used much less to demonstrate the principles.

5 More Resources

- Bootstrap
  - ISL 5.2
  - ESL 7.11
- Splines
  - ISL 7.2-7.5
  - ESL 5.1-5.4
- What Teachers Should Know About the Bootstrap: Resampling in the Undergraduate Statistics Curriculum, by Tim C. Hesterberg

5.1 Variations of the Bootstrap

- We have discussed only one type of bootstrap, *nonparametric/empirical/ordinary* where the observations are resampled.
- Another option is to simulate from the *fitted model*. This is called the *parametric* bootstrap.
  - For example, in the regression setting, estimate $\hat{\theta}$ and $\hat{\sigma}$.
  - Then given the original X's simulate new $y_i^* \mid x_i \sim f\left(x_i, \hat{\theta}\right) + \epsilon(\hat{\sigma})$.