大数据时代下的管理决策

Lec5: Machine Learning and Prediction

Zhaopeng Qu

Nanjing University Business School

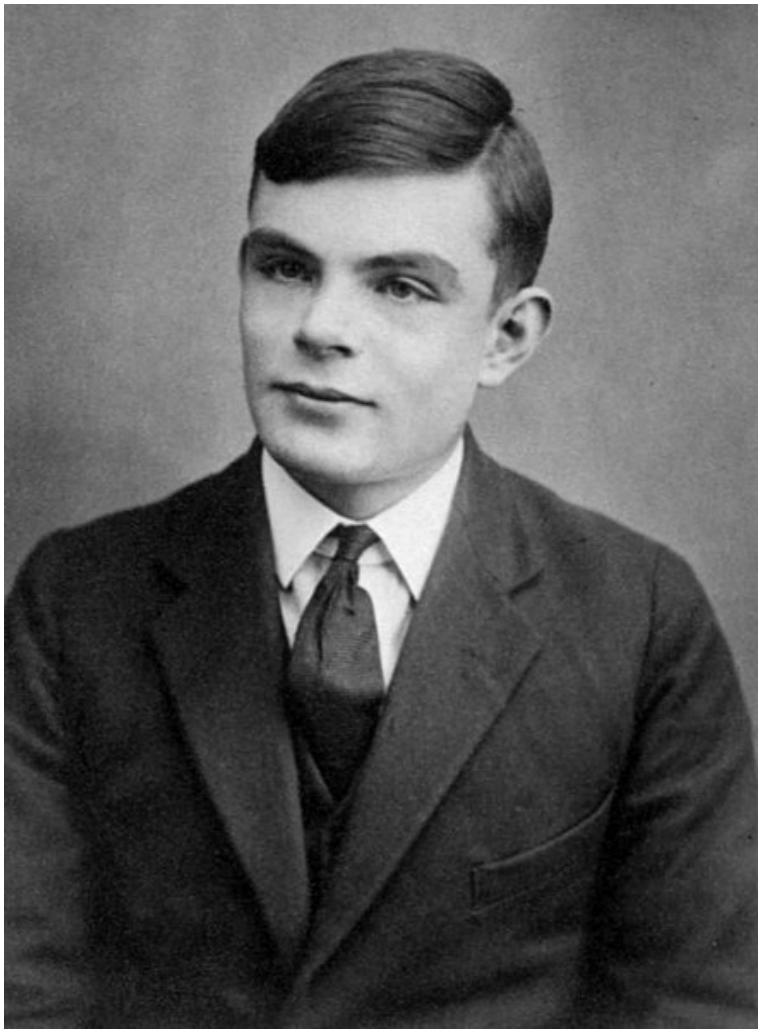
Mar 19 2022

Machine Learning and Prediction

Introduction

- **Machine learning** is originally a branch of computer science and statistics, tasked with developing algorithms to **predict outcomes y from observable variables x**.
- The *learning* part comes from the fact that we do not specify how exactly the computer should predict y from x. This is left as an empirical problem that the computer can *"learn"*.
- In general, this means that we abstract from the underlying models (biologic, economic, etc.) that creates the outcome that we want to predict.

The Origins of ML and AI



- **Alan Truing**(1912-1954) was an English mathematician, computer scientist and logician.
- He is widely considered to be the father of theoretical computer science and artificial intelligence.

The Origins of ML and AI

- **Authur Samuel**(1901-1990) was an American pioneer in the field of computer gaming and artificial intelligence.
- He pioneerly popularized the term **"machine learning"** in 1959.

Terminology: Econometrics V.S ML

	Econometrics	Machine learning
Object	estimate β	Fitted value of $\hat{y}$
Criterion	Unbiasedness and Consistency	Optimal fit
Evaluation	Conceptual Key assumptions	Empirical Cross-validate fit
Key question	*causal or not*	*accurate or not*

- Machine learning is a set of data-driven algorithms that use data to **predict** or **classify** some variable Y as a function of other variables X.

Terminology: Econometrics V.S ML

Machine learning	**Econometrics**
a learning problem	an empirical problem
to train an algorithm	to run an estimation procedure
training data	estimation sample
regression	predicting continuous dependent variables
classification	predicting limited dependent variables
a feature	an independent variable (x)
a label	a (categorical) dependent variable (y)
a response	a (continuous) dependent variable (y)

Machine Learning Algorithm

- **Any algorithm** that maps *features*(independent variables) into a *prediction*(dependent variable) can be thought of as within the realm of machine learning.
- There are many machine learning algorithm. The best methods vary with the particular data application.
  - **Regression**: OLS,LASSO,Ridge
  - **Classification**: logit,probit
  - **Regression trees** and **random forests**
  - **Neural networks** and **support vector machines**
  - ...
- In many cases, the theoretical properties (e.g. convergence and limit distribution) of these algorithms are even unknown – but *that is not the point*.

Machine Learning: A broad classification

- Main methodological classes:
  1. **Supervised learning**: We have data on both an *outcome* y and *explanatory variables* x.
     - **Regression**: if y is continuous,
     - **Classification**: if y is discrete
  2. **Unsupervised learning**: we have no data on y, only on x.
     - **Cluster Analysis** (without specifying what to group together)
     - **Principle Component Analysis**(PCA)
  3. **Reinforcement learning**

Our focuses

- Main Content
  - Basic ideas of ML
  - Some new econometric models(algorithms)
  - How and when to apply ML methods in economics in some cases,especially for prediction
- Not about
  - Cutting-edge ML techniques
  - Computational aspects
  - Data wrangling
  - Distributed computation systems

Supervised learning

A Statistical model

Suppose the the relationship between x and y can be written as an additive error model:

$$Y = f(X) + \epsilon$$

- where $f()$ is some fixed but unknown function of X, which it represents the systematic relationship between X and Y.
- And ϵ represents *idiosyncratic deviations* from this systematic relationship, so it satisfies

$$E(\epsilon \mid X) = 0 \; and \; E(\epsilon) = 0$$

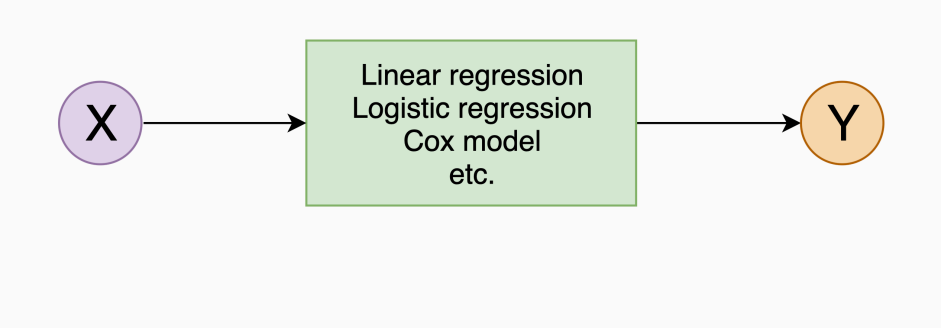
Causal inference v.s Prediction

1. Causal inference: How do changes in X affect Y?
2. Prediction: Predict Y using our estimated $f(X)$, i.e.,

$$\hat{Y} = \hat{f}(X)$$

The Objective of Superivsed Learning

- In supervised learning, we want to make a prediction about the response Y based on features X.
- Because it helps us to make a prediction, it is useful to estimate $f(\cdot)$, which represents the systematic relationship between features(X) and the response(Y).



- However, for prediction we **do not care about** $f(\cdot)$ itself. We can treat it as a *black box*, and any approximation $\hat{f}(\cdot)$ that yields a good prediction is *good enough*.
  - Whatever works, works

Example: predicting electricity demand

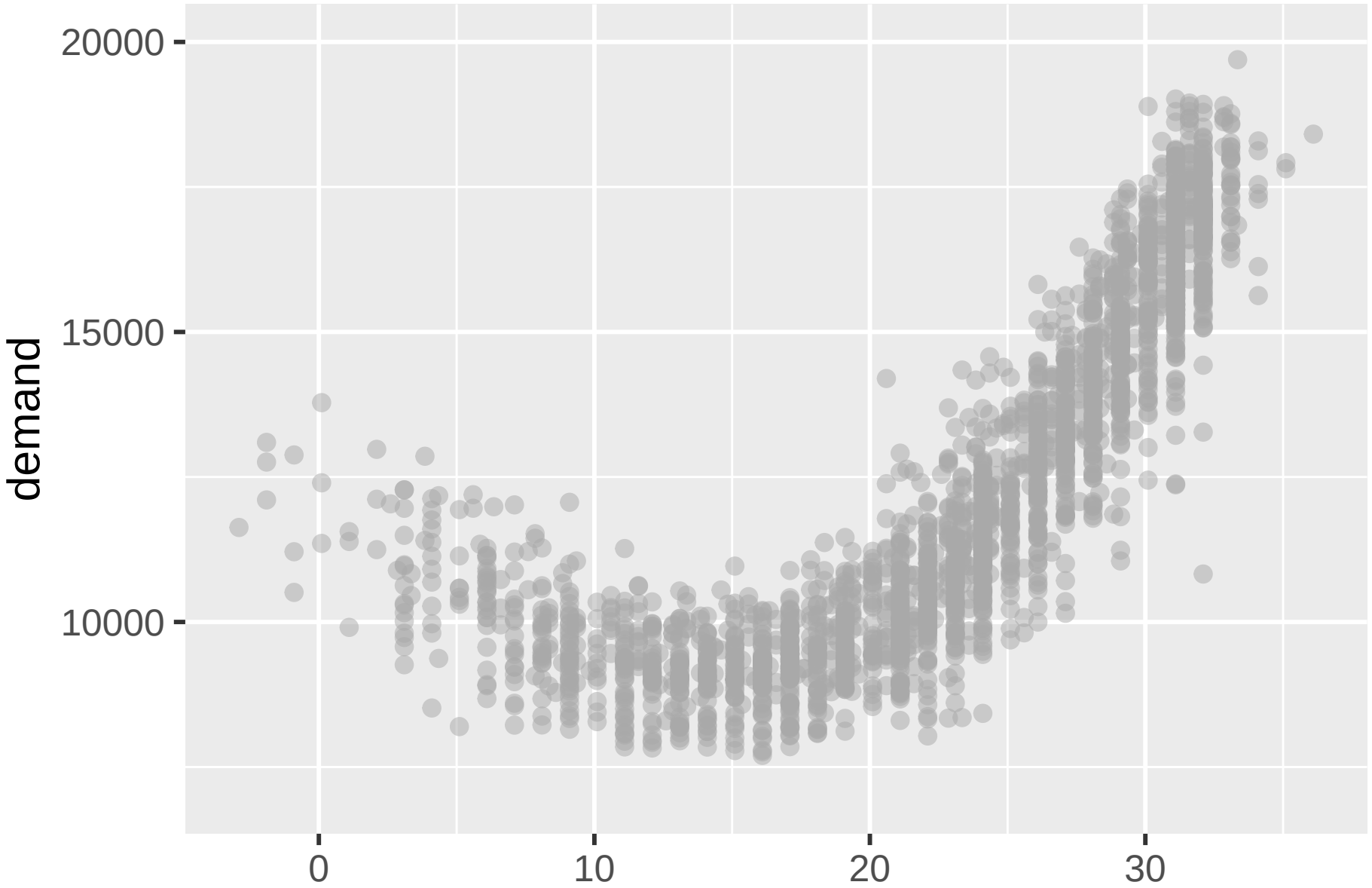
- ERCOT operates the electricity grid for 75% of Texas by area.

Example: predicting electricity demand

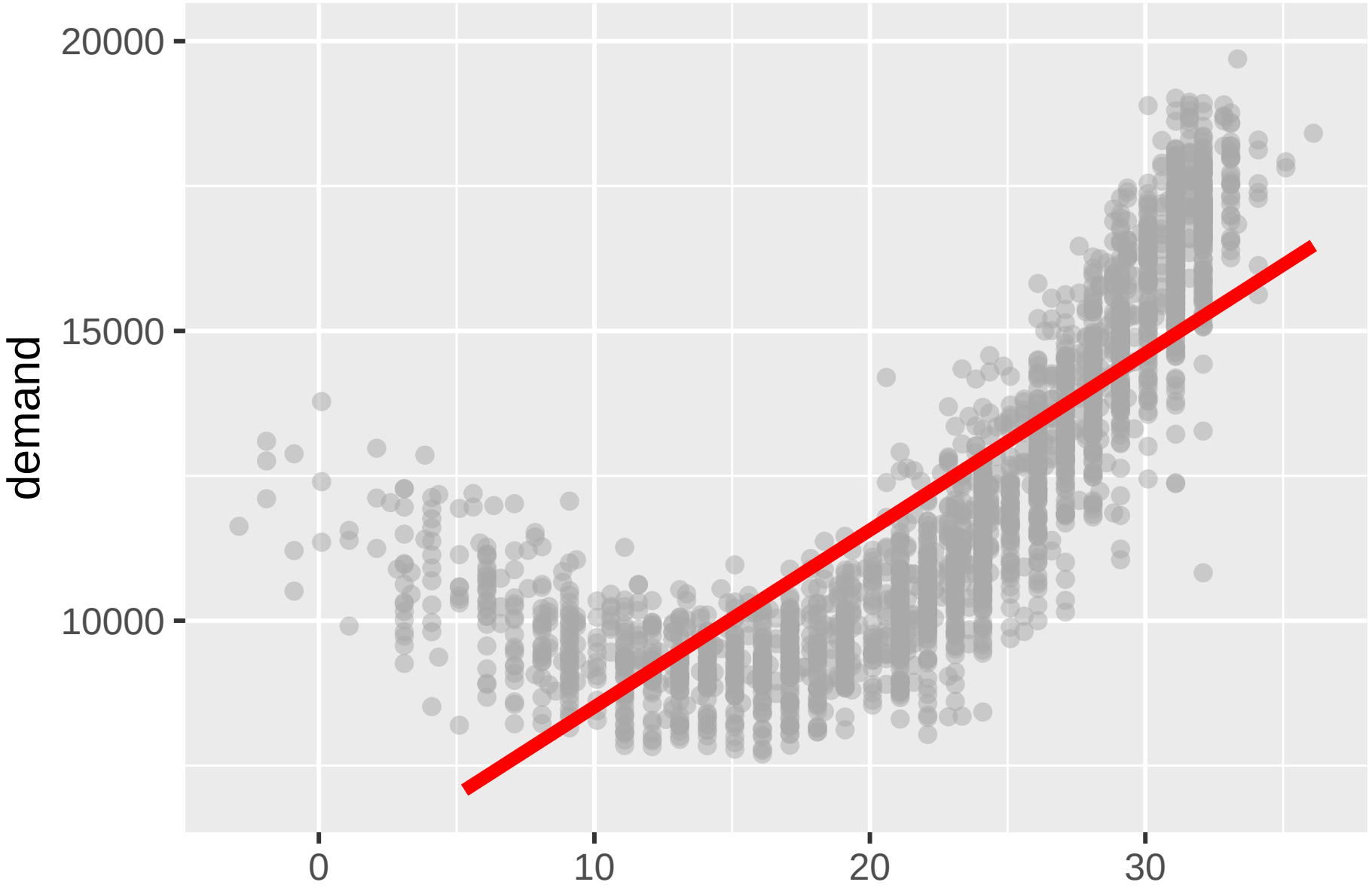
The 8 ERCOT regions are shown at left. We'll focus on a basic prediction task:

- $Y =$ demand (megawatts) in the Coast region at 3 PM, every day from 2010-2016.
- $X =$ average daily temperature at Houston's Hobby Airport (degrees Celsius)

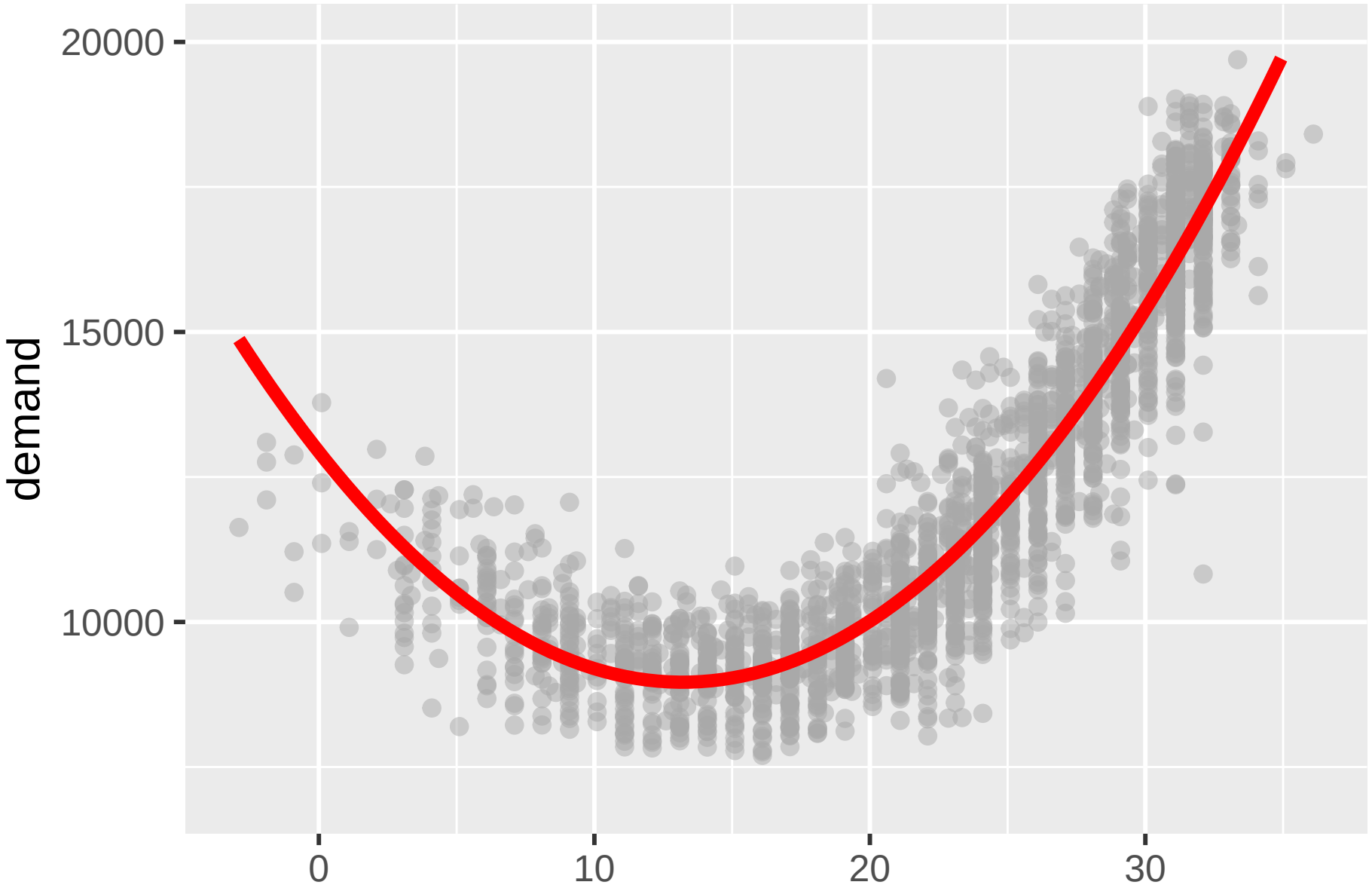
Demand v.s Temperature



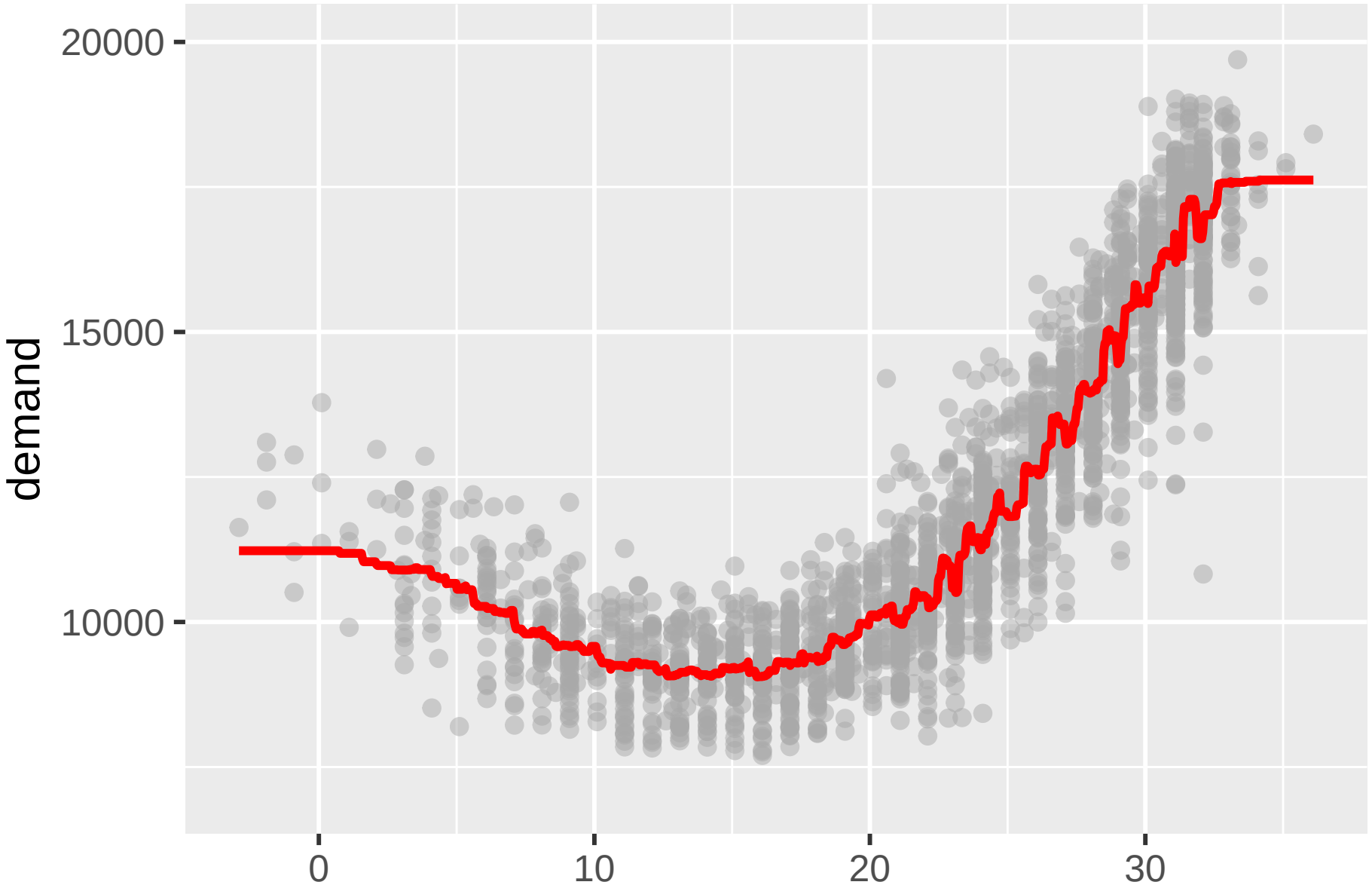
A linear model?



A quadratic model?



How about this model?



Formally: Optimal Objective

- Formal statistics can help us though to figure out **what is a good prediction**.
- Formally, a supervised learning algorithm takes as an input a **loss function** and searches for *a function within a function class* that has a low expected prediction loss on a new data point from the same distribution.
- A very common loss function in a regression setting is the **mean squared error** (MSE), thus

$$MSE = \frac{1}{N}\Sigma_{i=1}^{N}(Y_i - \hat{Y}_i)^2$$

Error Decomposition

- The MSE is a sample concept. The population analogue is called the **expected mean-squared error**(EMSE). Because $E(\epsilon \mid x) = 0$ and $E(\epsilon) = 0$, then

$$\begin{aligned} EMSE &= E[f(X) + \epsilon - \hat{f}(X)]^2 \\ &= E[(f(X) - \hat{f}(X))^2] + E[\epsilon^2] - E[2(f(X) - \hat{f}(X))\epsilon] \\ &= \underbrace{E[f(X) - \hat{f}(X)]^2}_{\textbf{Reducible error}} + \underbrace{Var(\epsilon)}_{\textbf{Irreducible error}} \end{aligned}$$

- We could prove that

$$E(Y \mid X) = arg\ min_{f(X)} EMSE$$

(Ref: MHE-Theorem 3.1.2,pp33)

  - Thus the **Conditional Expectation Function(CEF)** is the best predictor of Y given X.

Unknown function form of $f(X)$

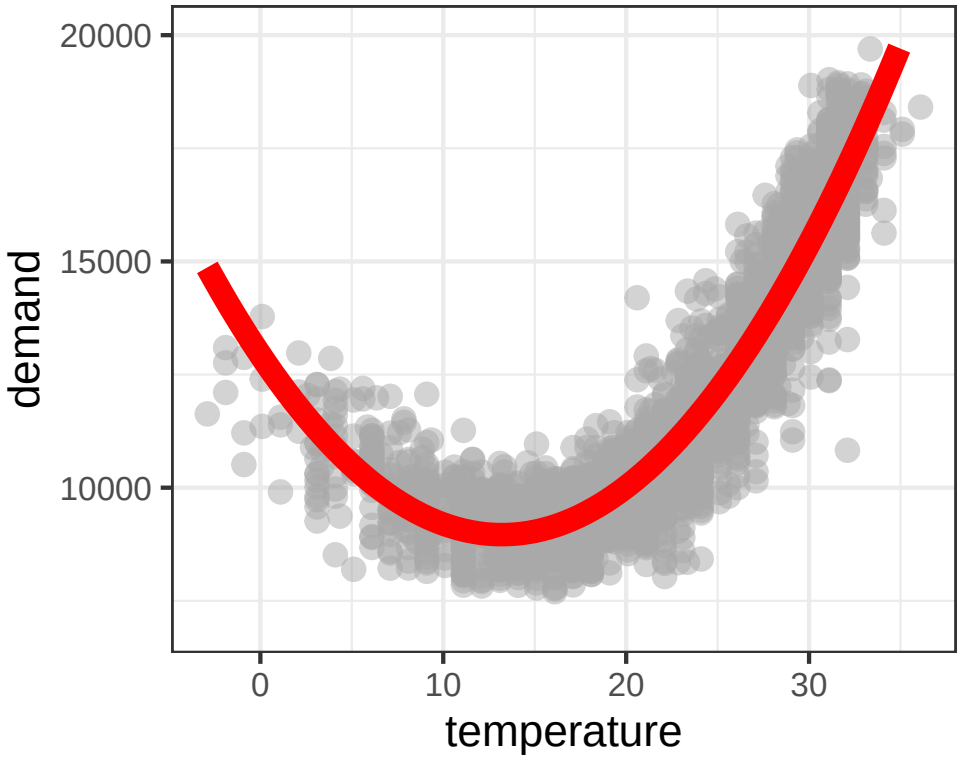
How to obtain the forms of CEF or f(X)

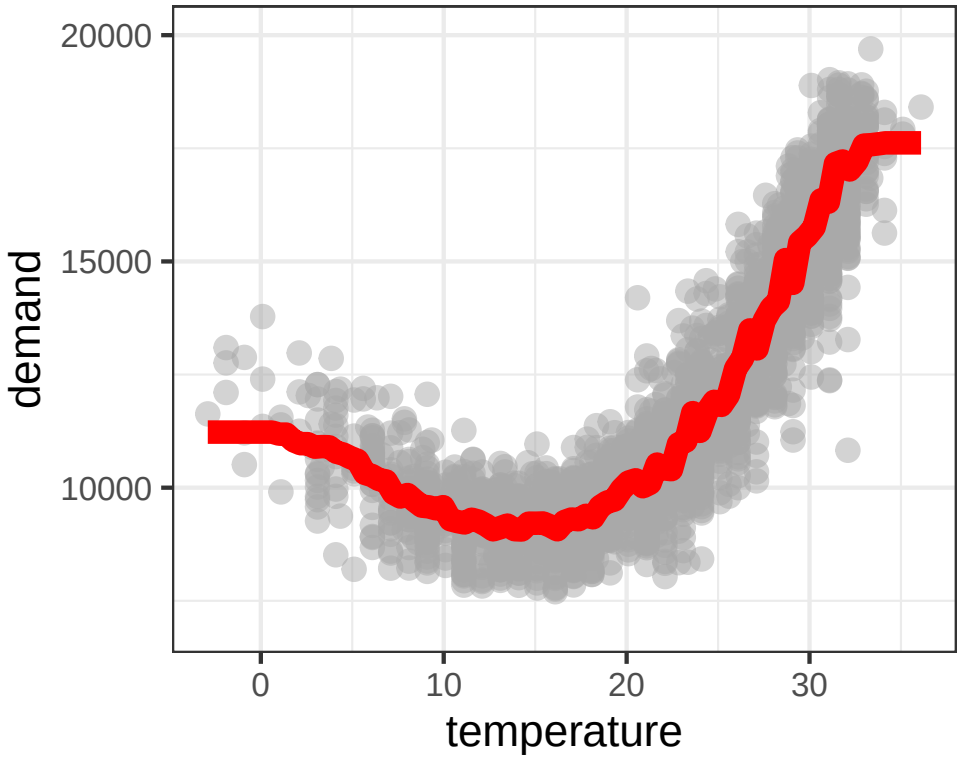
- Parametric: assume a particular, restricted functional form (e.g. linear, quadratic, logs, exp)

$$f(X) = g(\beta X)$$

  - The simplest one is OLS regression: $f(X) = X'\beta$
- Nonparametric: flexible forms not easily described by simple math functions.
  - Matching(Nearest Neighbors)

A quick comparison





Parametric: polynomial model

- $f(X) = \beta_0 + \beta_1 X + \beta_2 X^2$

Nonparametric: k-nearest neighbors

- $f(X) =$ average Y value of the 50 points closest to X

Estimating a parametric model: three steps

Suppose we have data in the form of (x_i, y_i) pairs. Now we want to predict y at some **new** point $y^\star$.

1. Choose a functional form of the model, e.g.

$$f(X) = \beta_0 + \beta_1 X$$

2. Choose a *loss function* that measures the difference between the model predictions $f(X)$ and the actual outcomes y. E.g. least squares:

$$\begin{aligned} L(\beta_0, \beta_1) &= \sum_{i=1}^{N} (y_i - f(X_i))^2 \\ &= \sum_{i=1}^{N} (y_i - \{\beta_0 + \beta_1 x_i\})^2 \end{aligned}$$

3. Find the parameters that minimize the loss function.

Estimating k-nearest neighbors

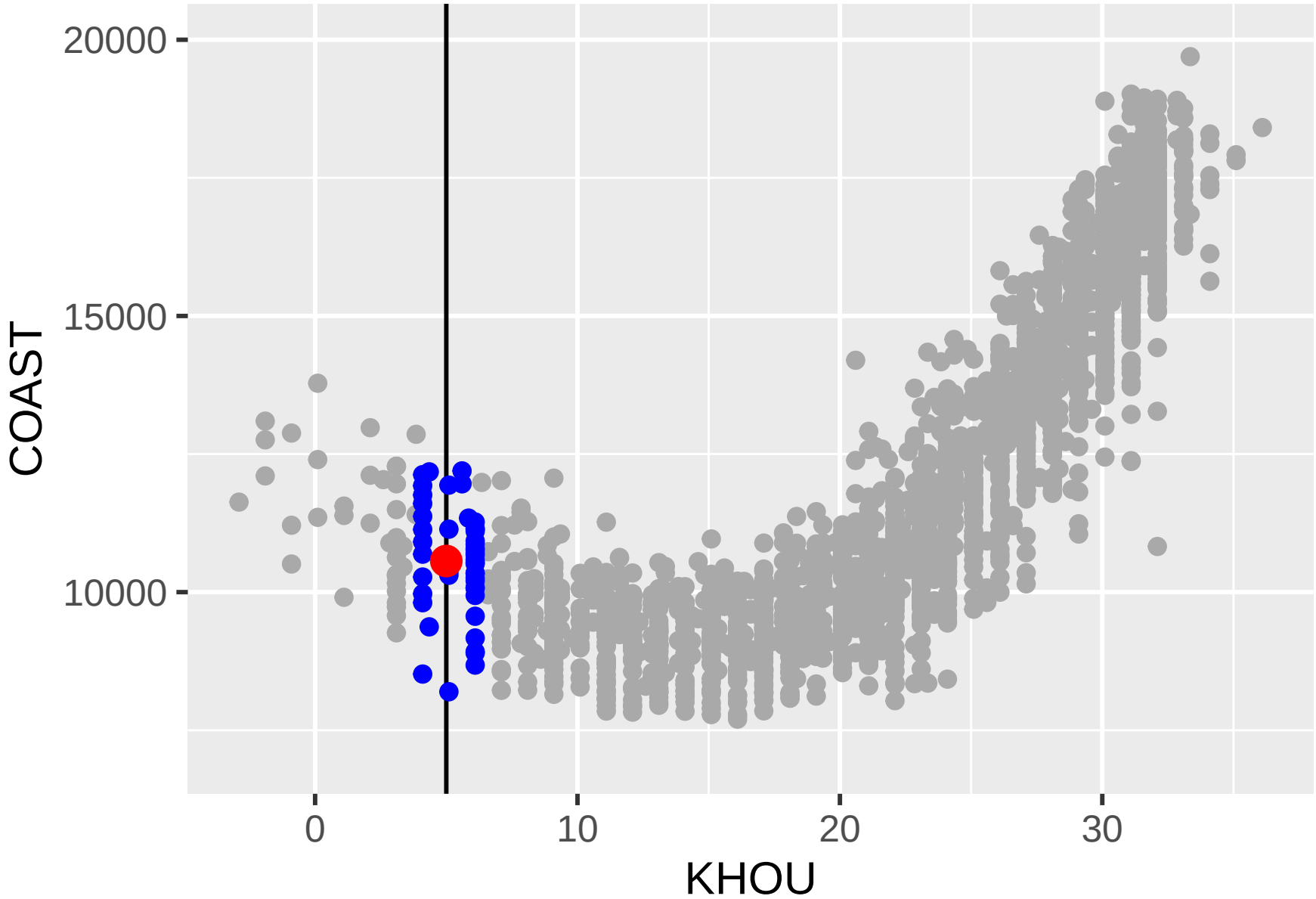
1. Pick the K points in the data whose x_i values are closest to $x^\star$. Call this neighborhood $\mathcal{N}_K(x^\star)$.
2. Average the y_i values for those points and use this average to estimate $f(x^\star)$:

$$\hat{f}(x^\star) = \frac{1}{K} \sum_{i:x_i \in \mathcal{N}_K(x^\star)} y_i$$

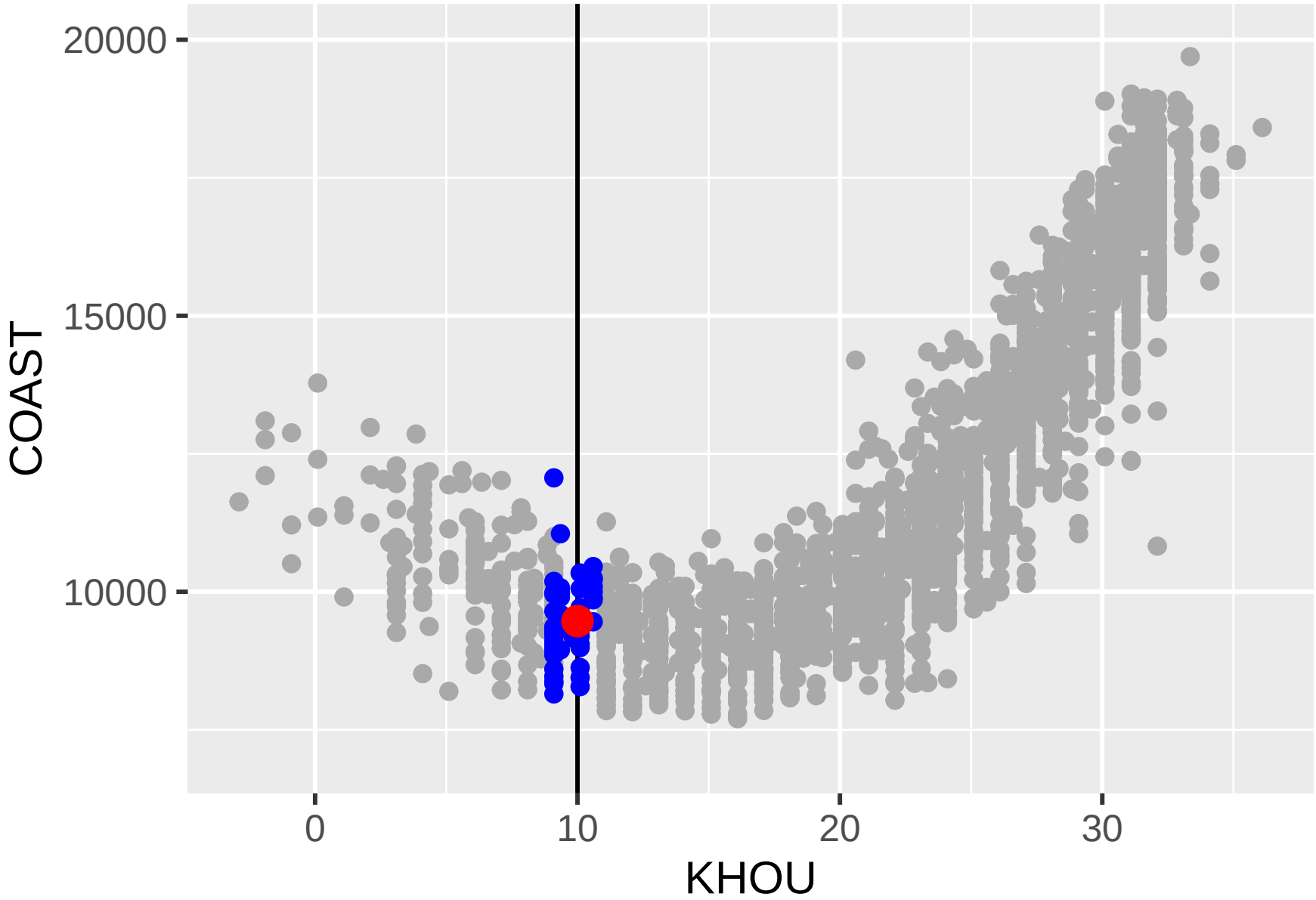
There are **no explicit parameters** (i.e. β's) to estimate.

- Rather, the estimate for $f(x)$ is defined by a particular *algorithm* applied to the data set.

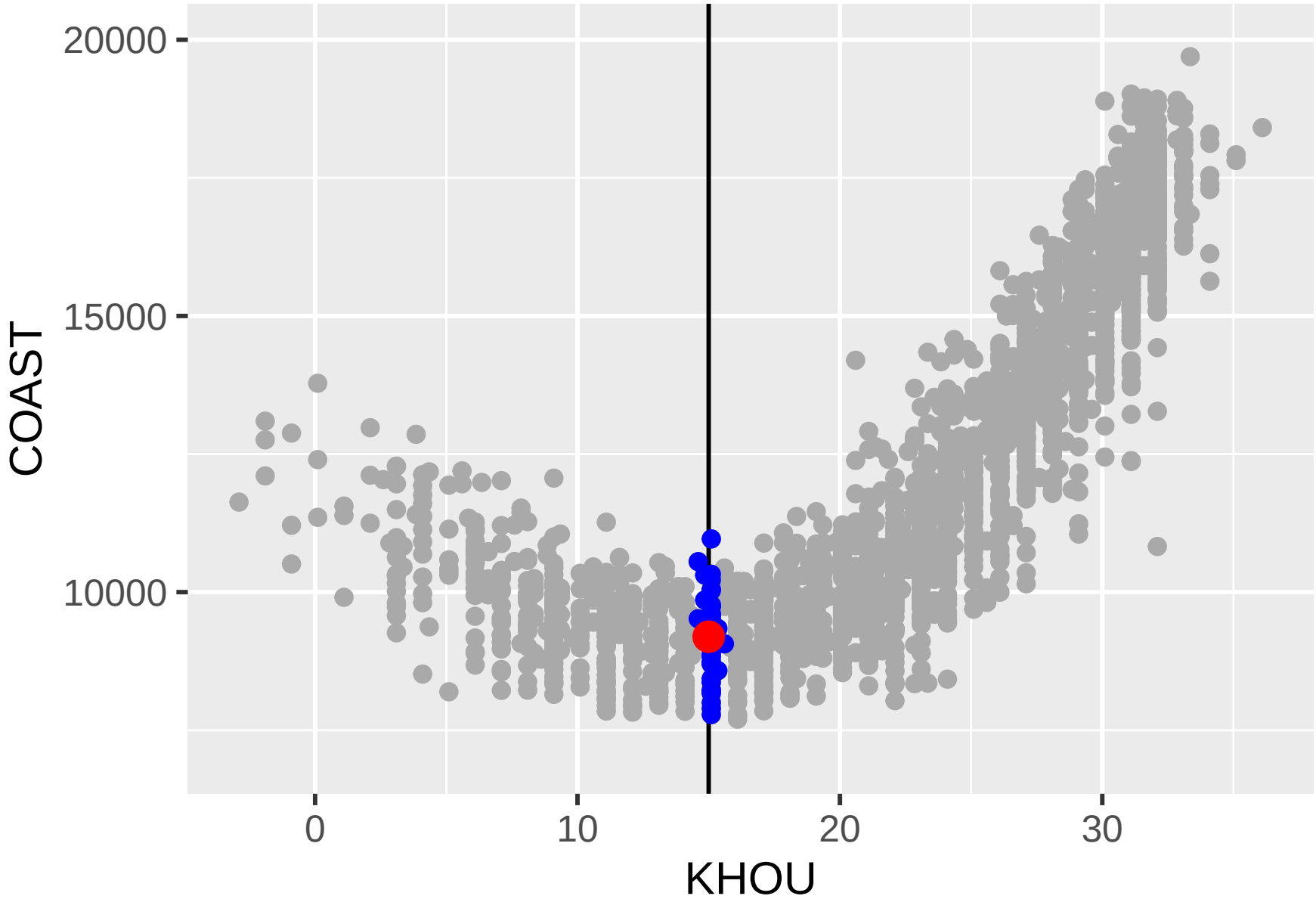
At x=5 and K=50



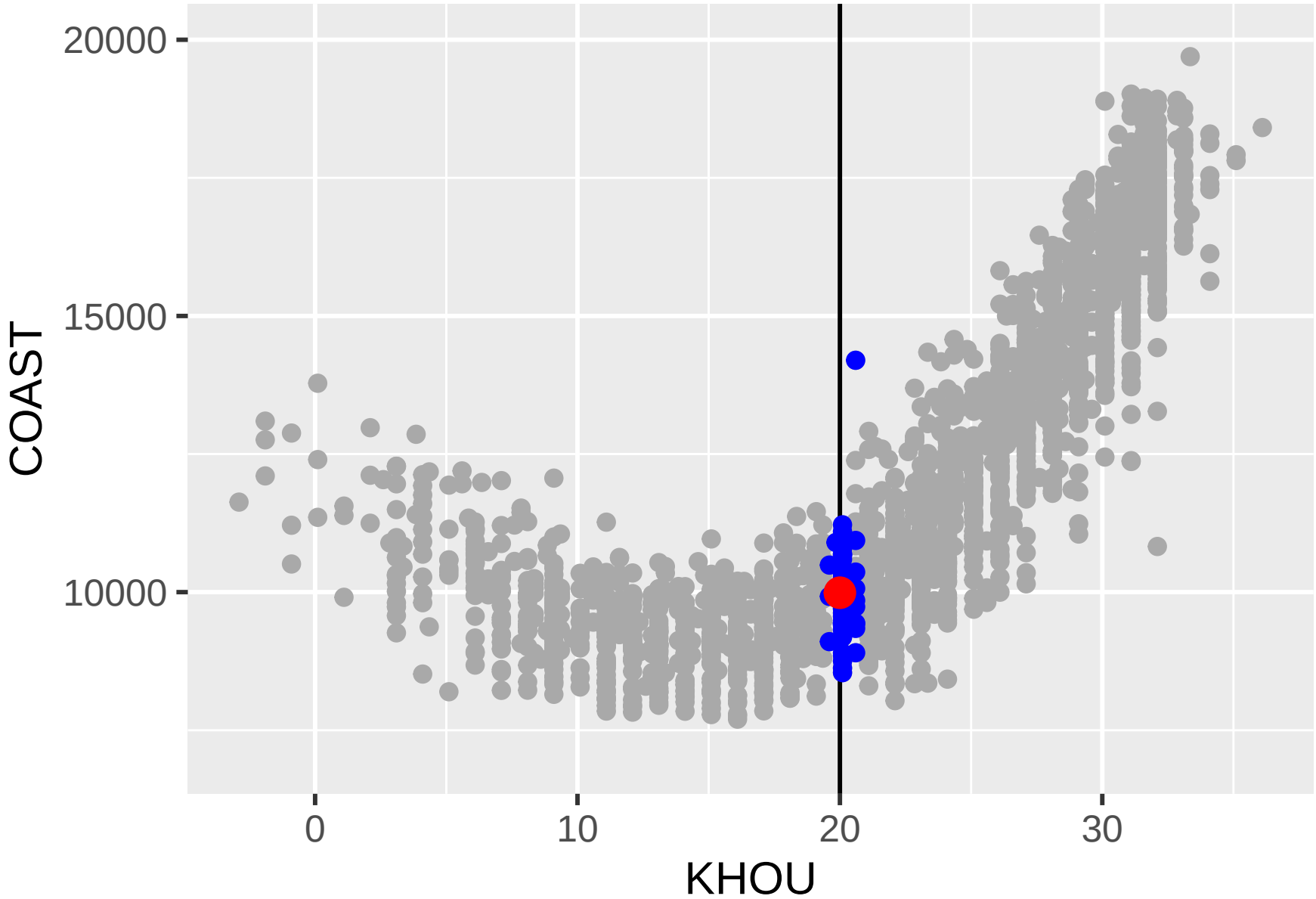
At x=10 and K=50



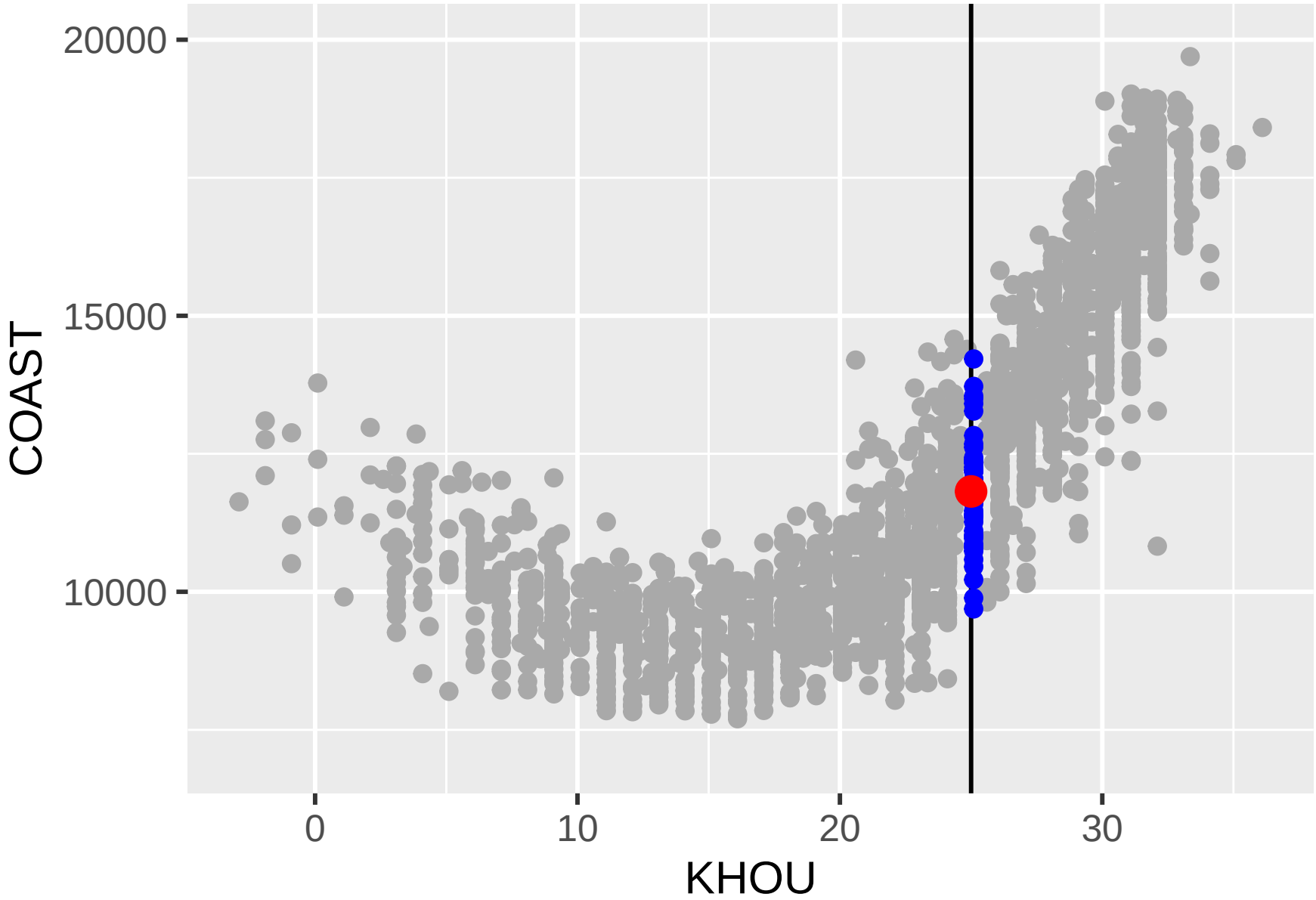
At x=15 and K=50



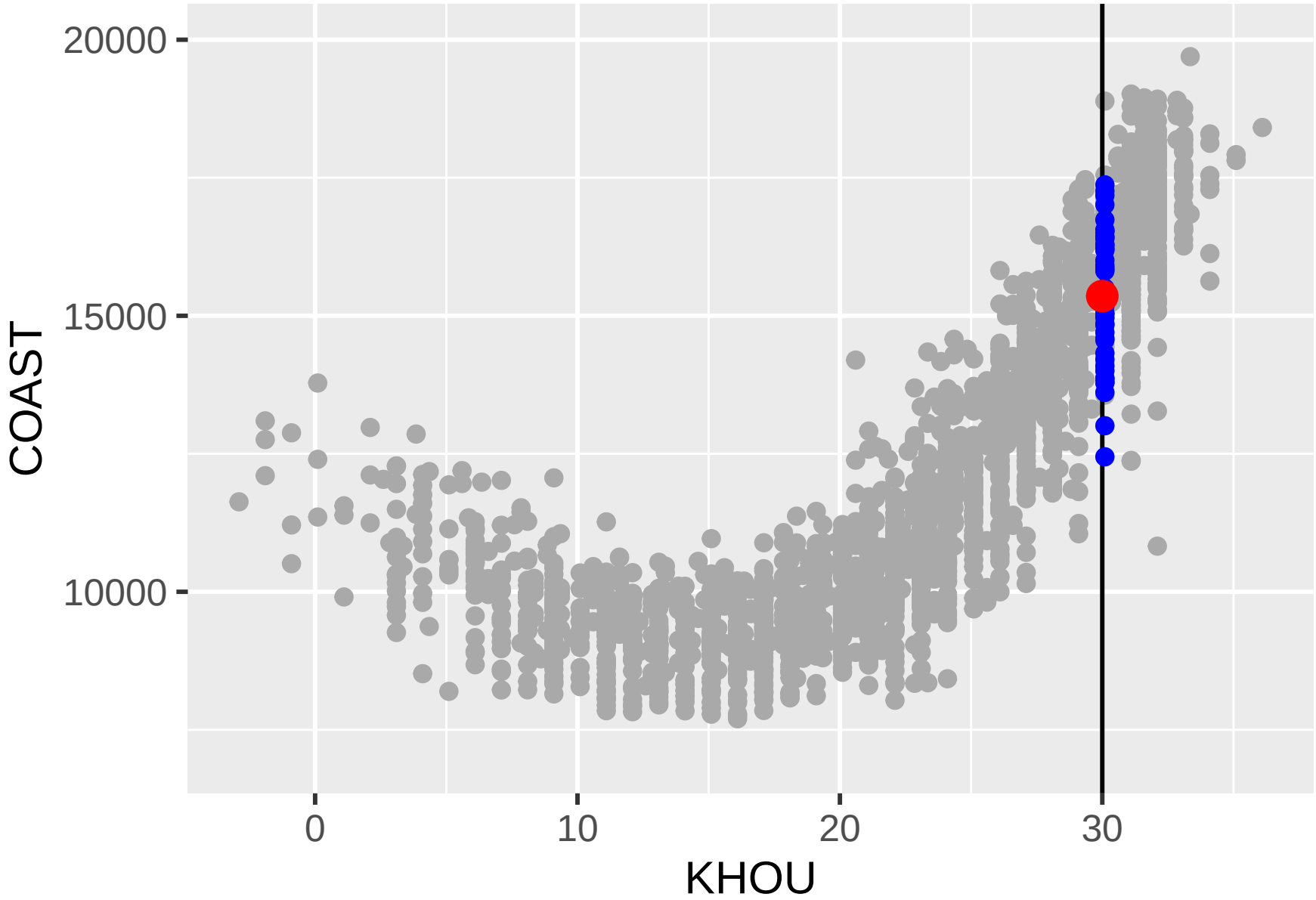
At x=20 and K=50



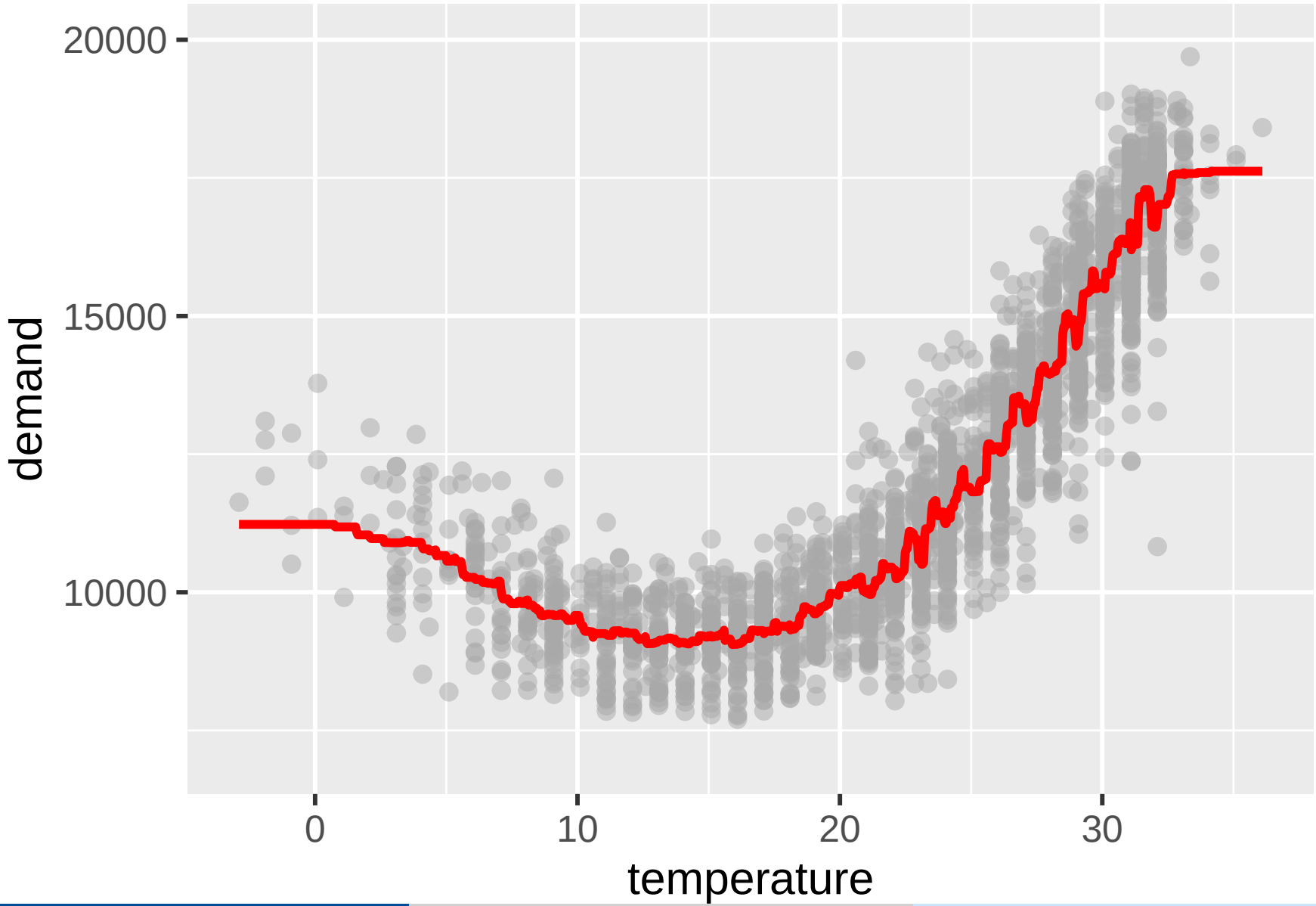
At x=25 and K=50



At x=30 and K=50



The predictions across all x values

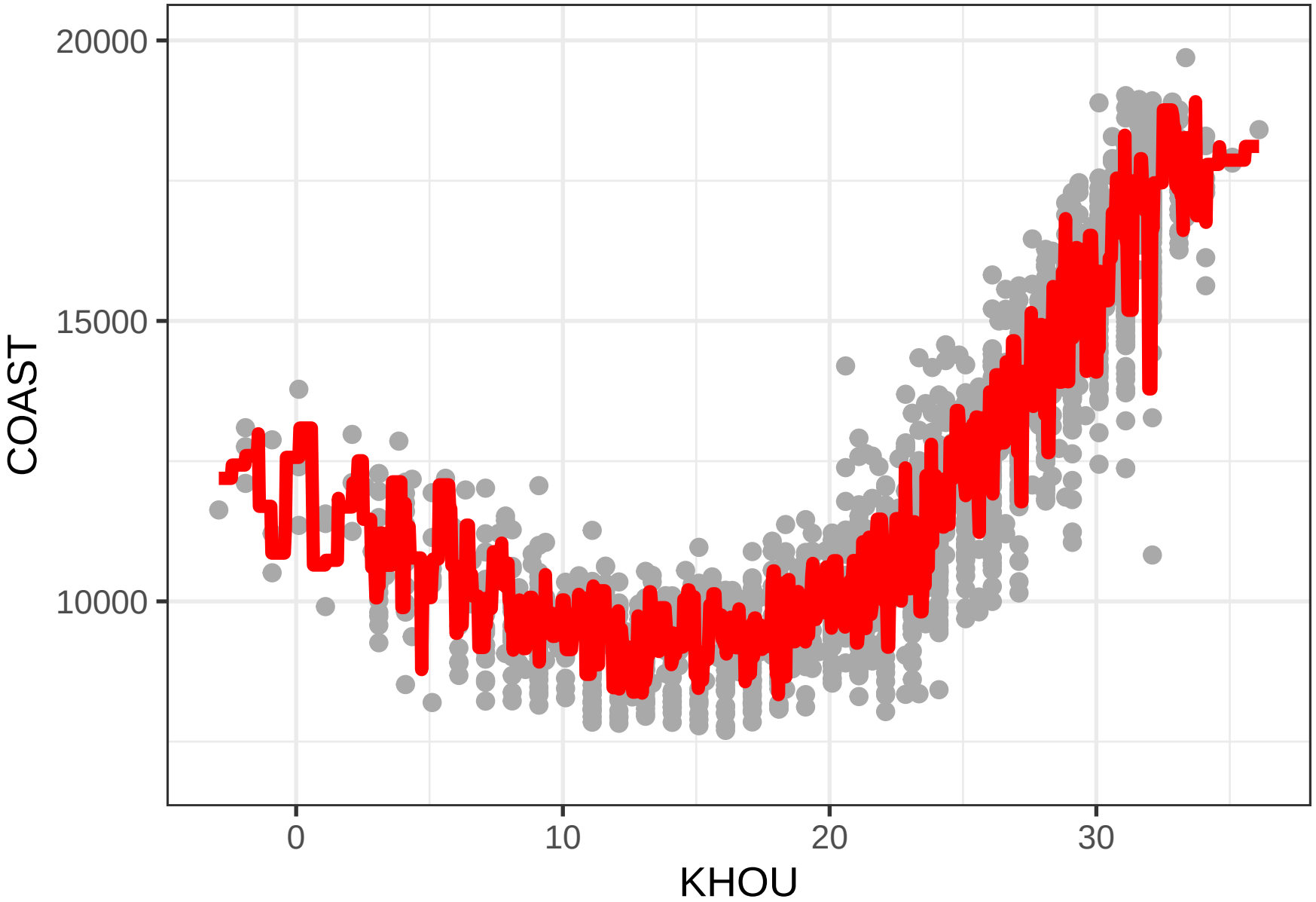


Two questions

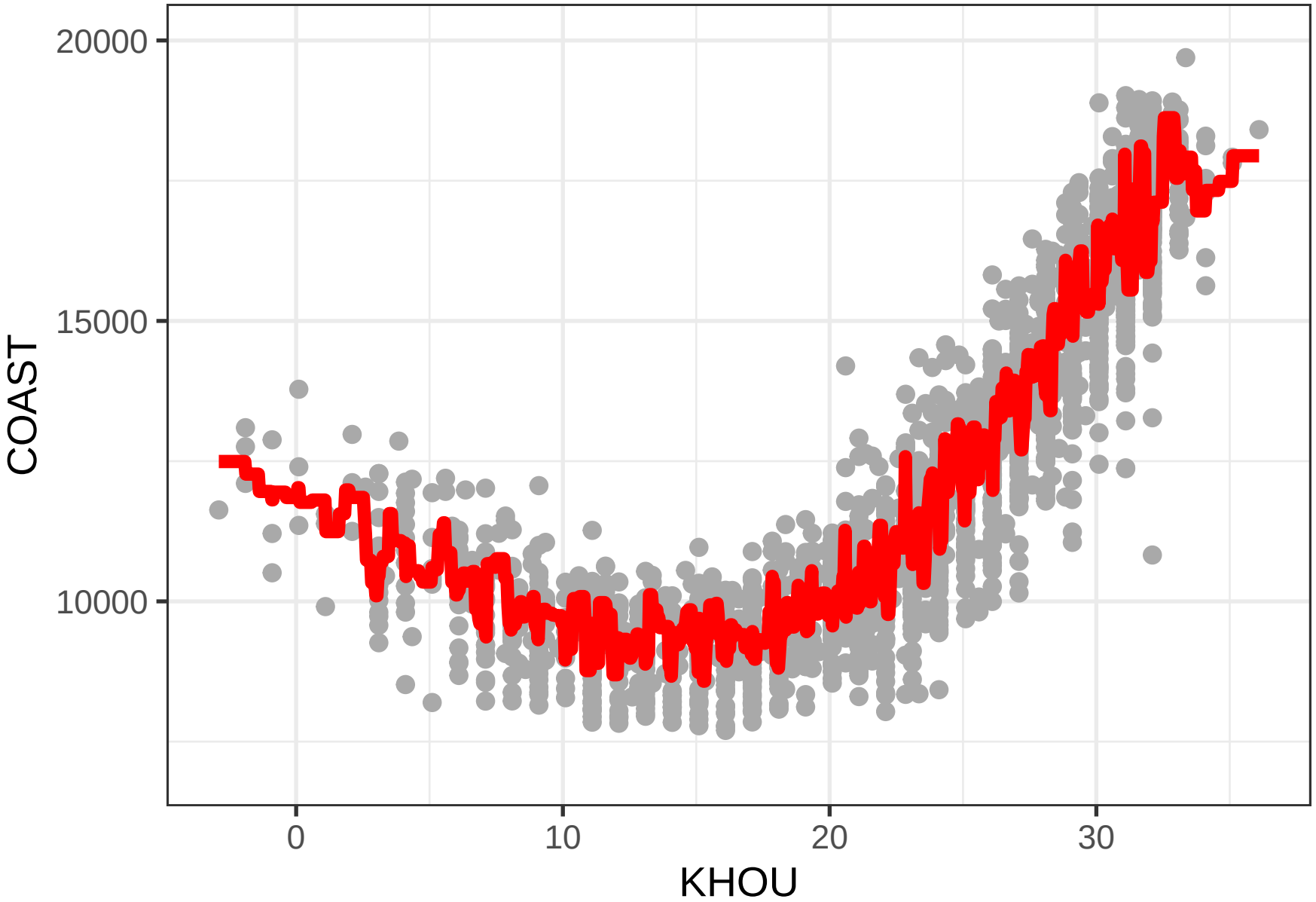
This procedure raises two obvious questions:

1. So why average the nearest $K = 50$ neighbors? Why not $K = 2$, or $K = 200$?
2. And if we're free to pick any value of K we like, how should we choose?

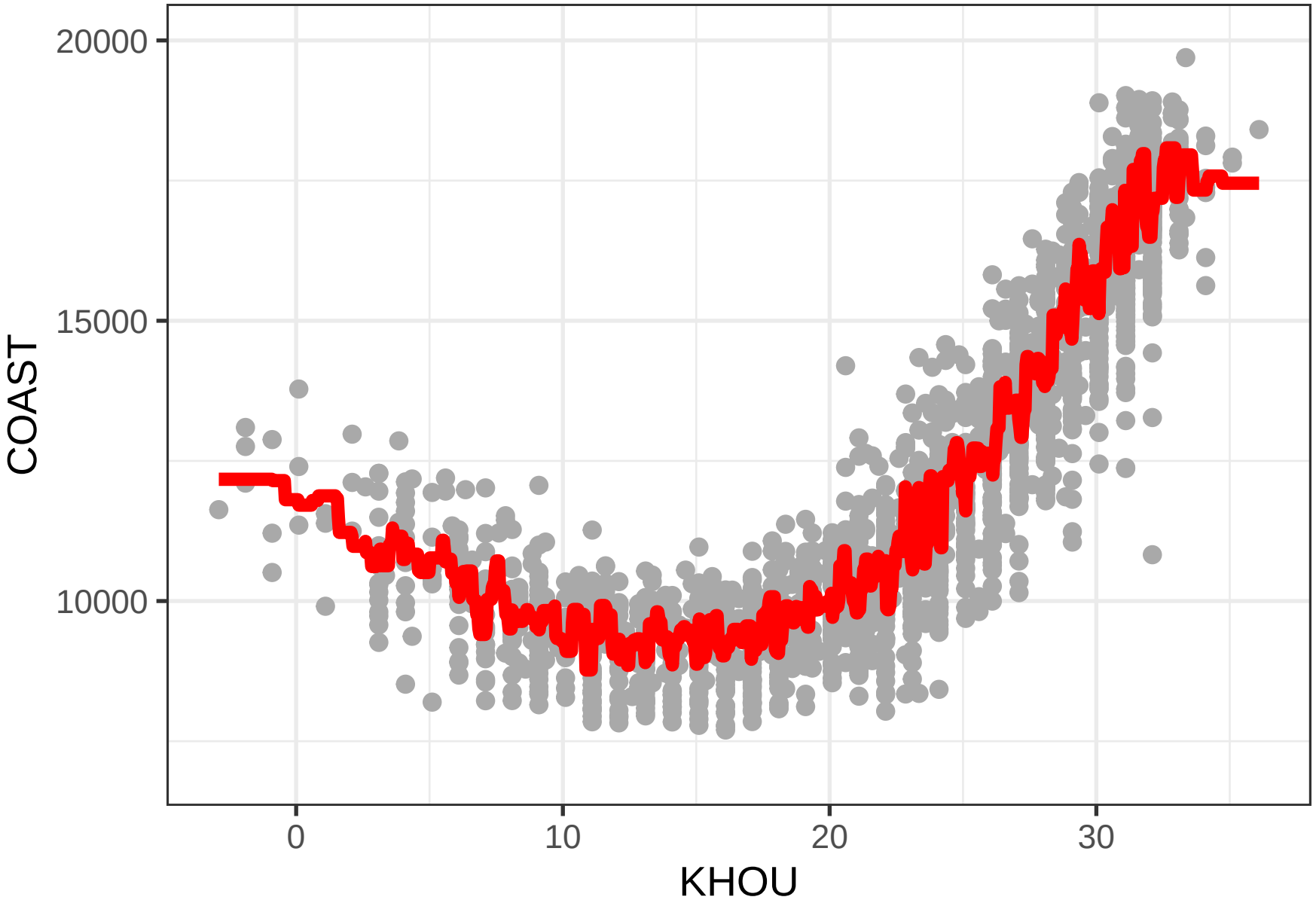
K=2



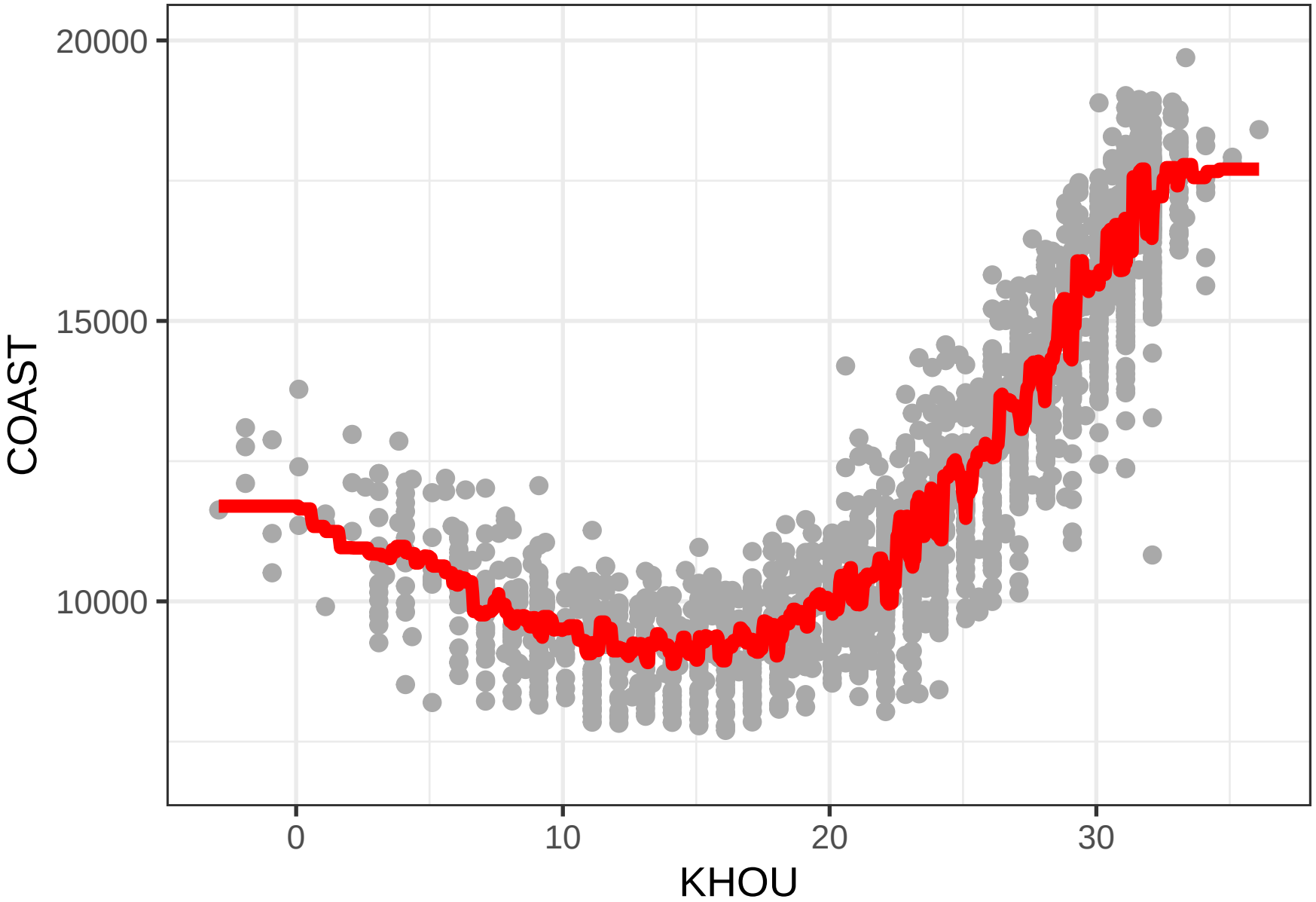
K=5



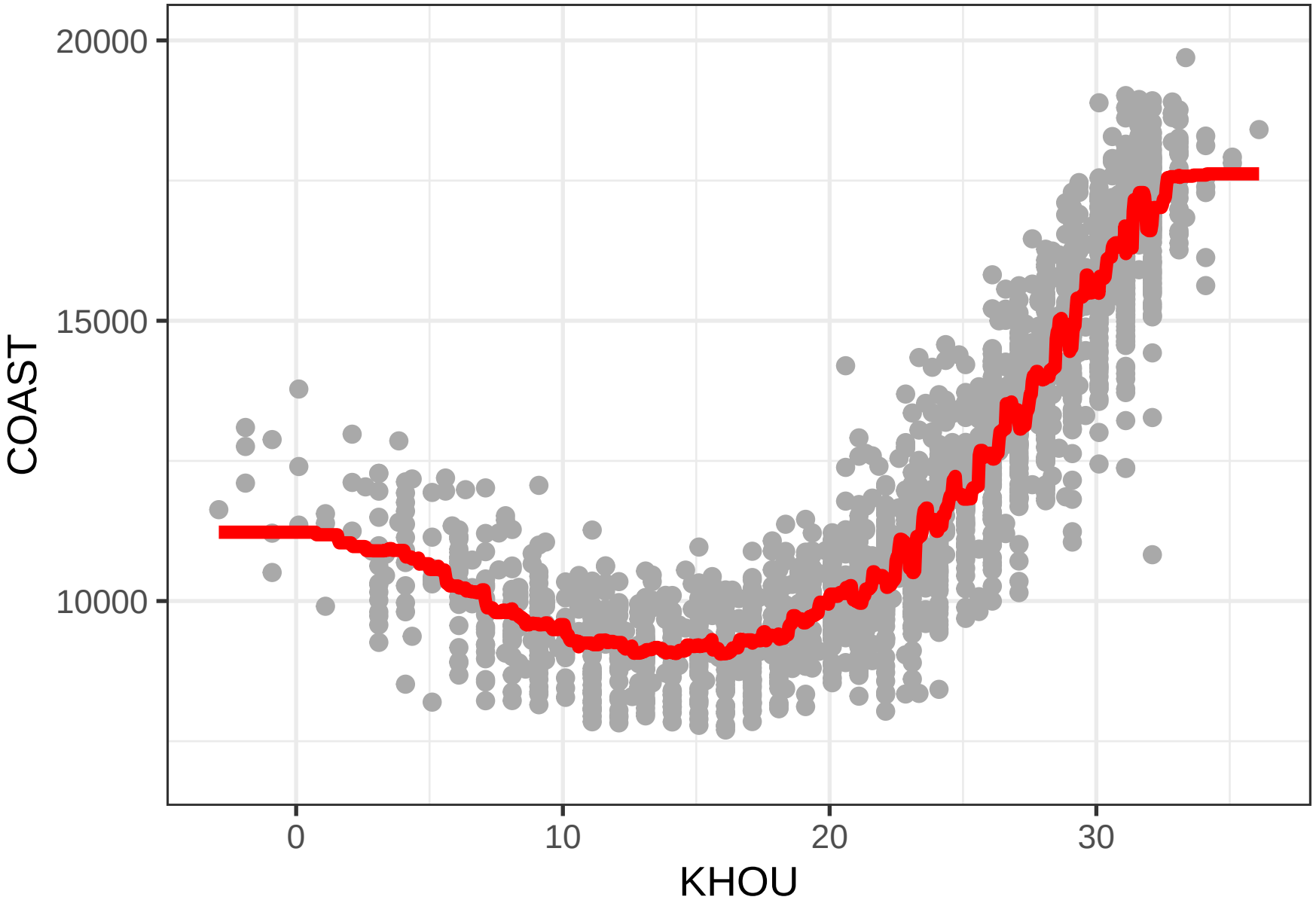
K=10



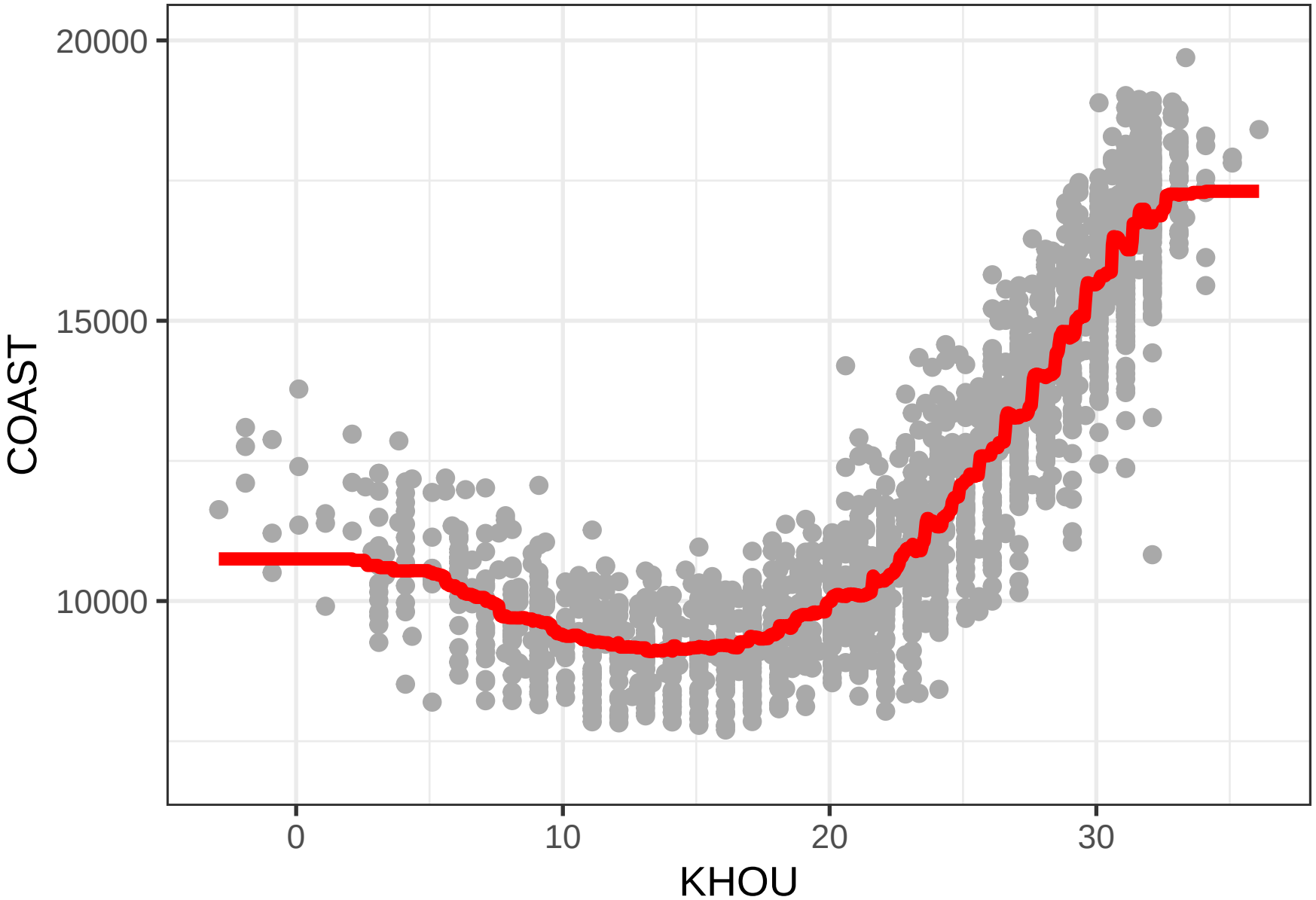
K=20



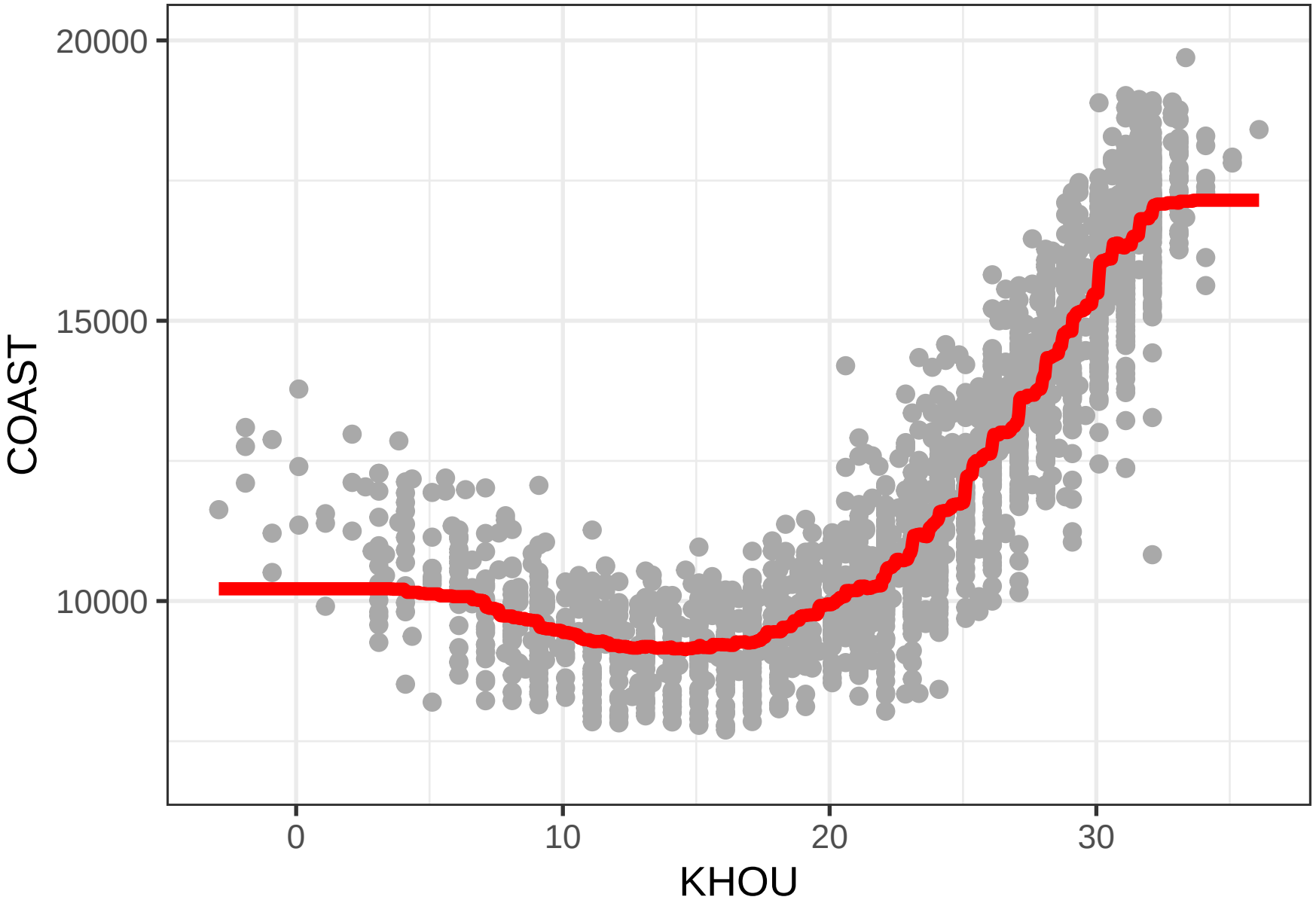
K=50



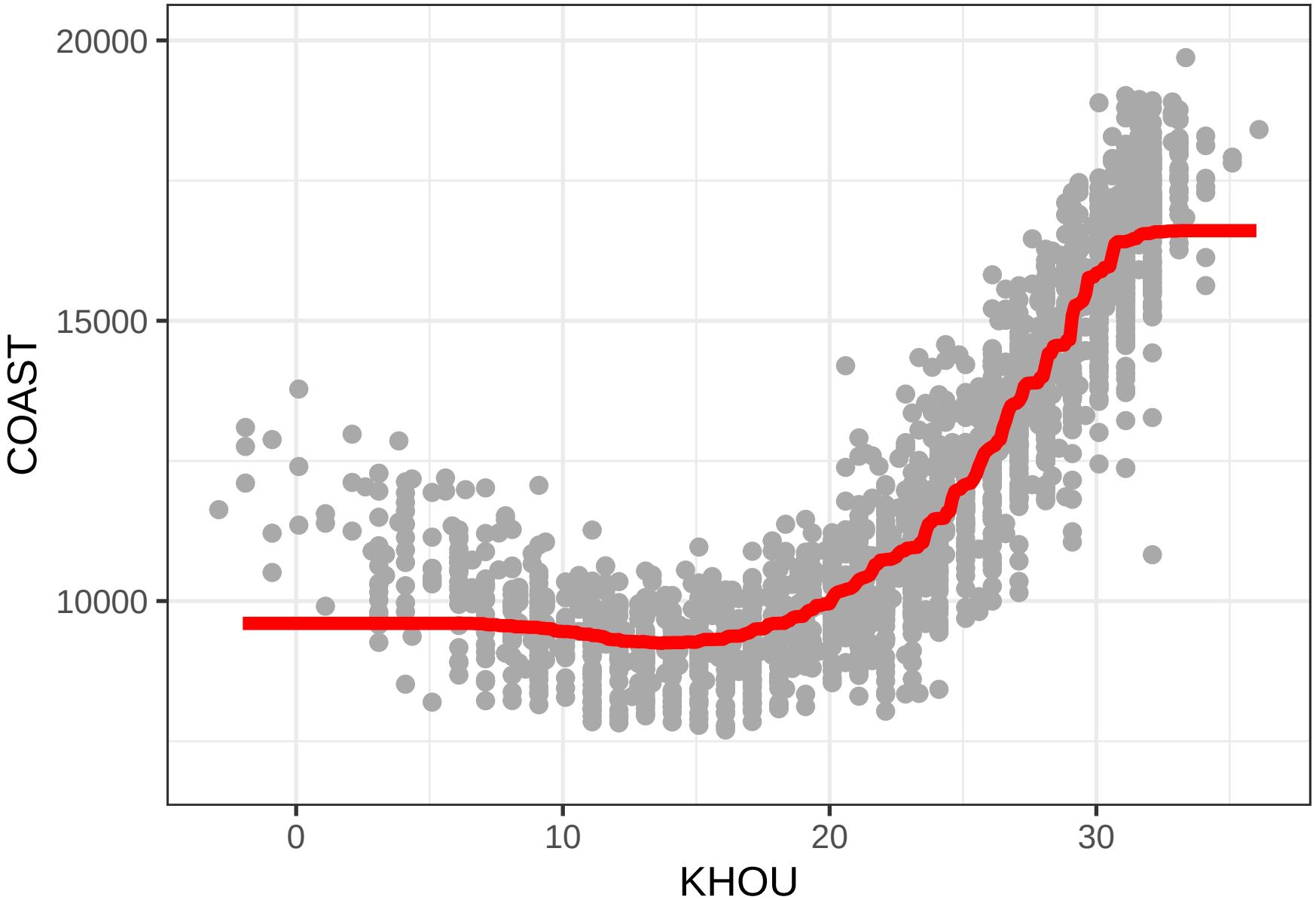
K=100



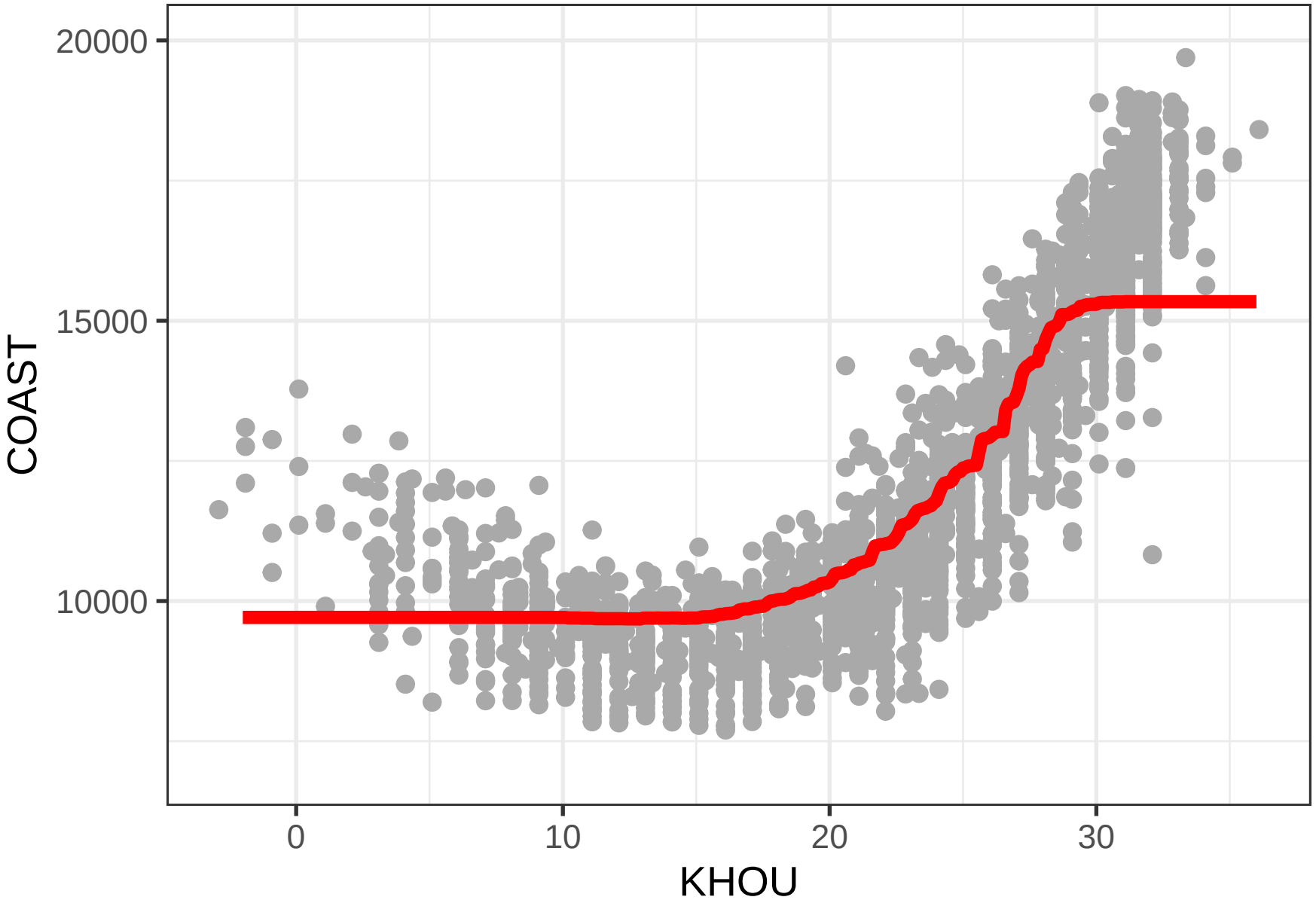
K=200



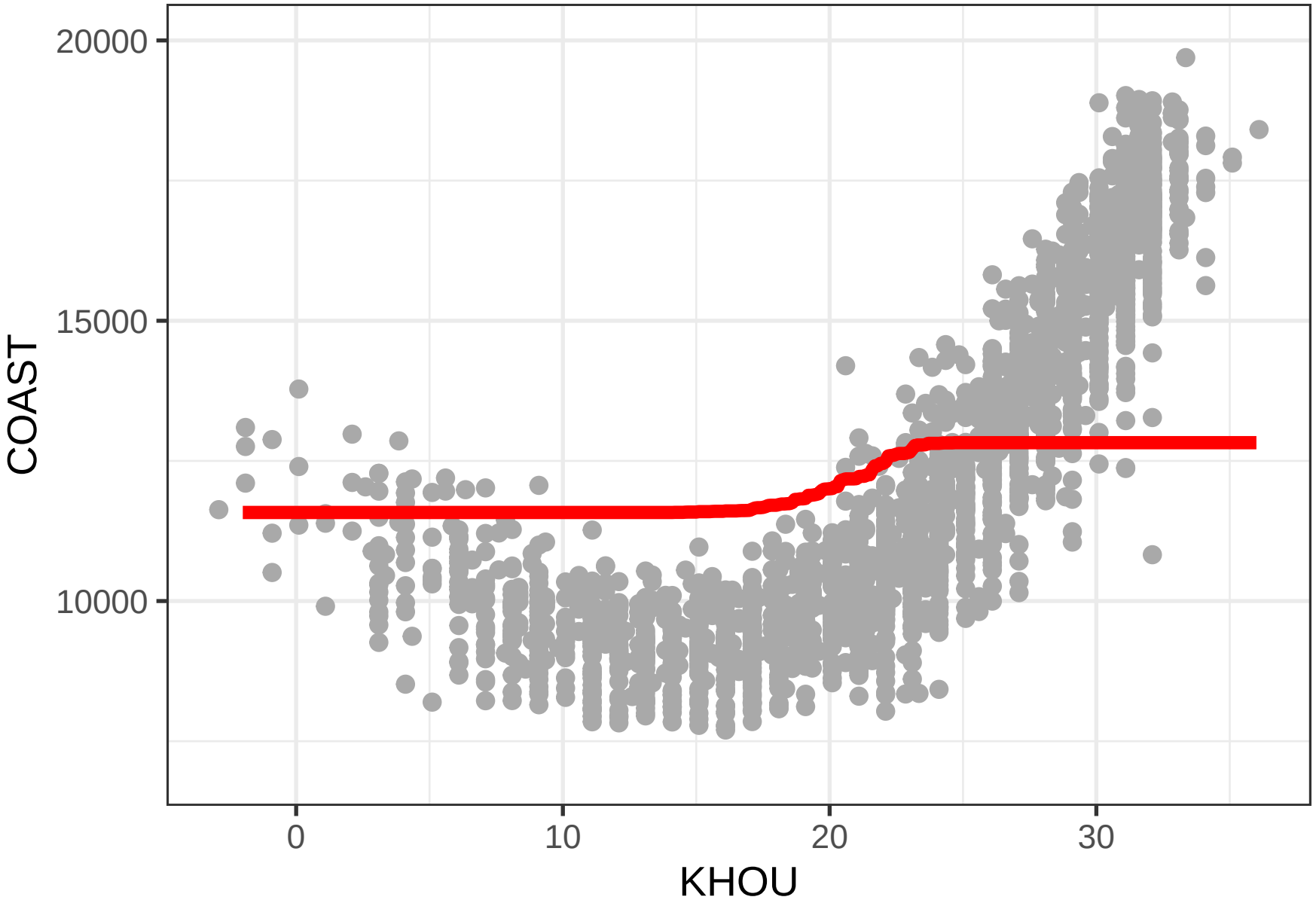
K=500



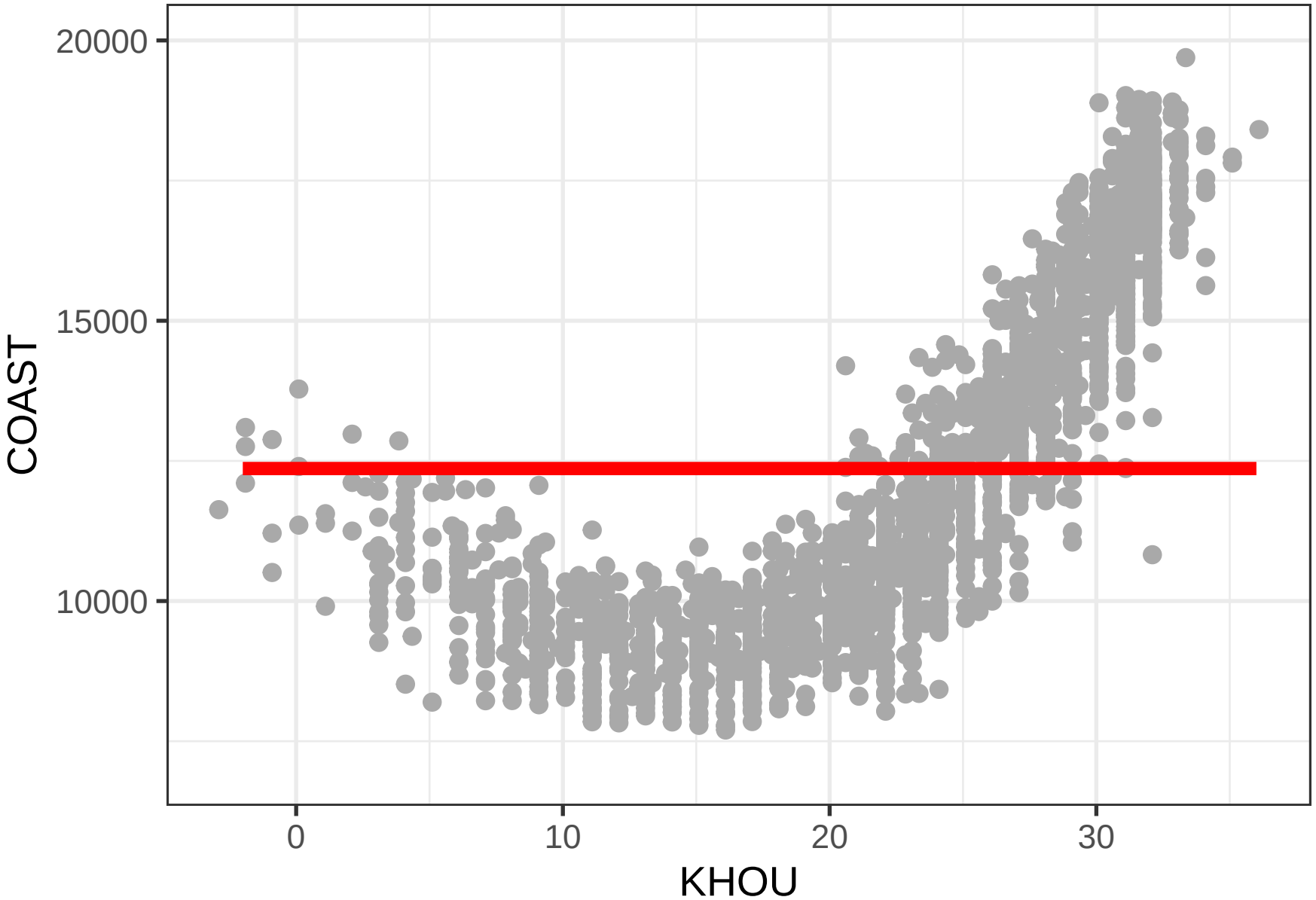
K=1000



K=2000



K=2357



Complexity, generalization, and interpretion

Smaller values of K give more flexible, but less stable function estimates:

- they can capture very fine-scale structure in $f(x)$, because they're only averaging points from a small neighborhood...
- but they can also confuse noise for signal!

Larger values of K give less flexible, but more stable function estimates:

- they can't adapt as much to wiggles in $f(x)$, because they're averaging points over a larger neighborhood.
- but this makes them less prone to confusing noise for signal.

You probably got the sense from the pictures that there's a "happy medium" somewhere. **How should we find it?**

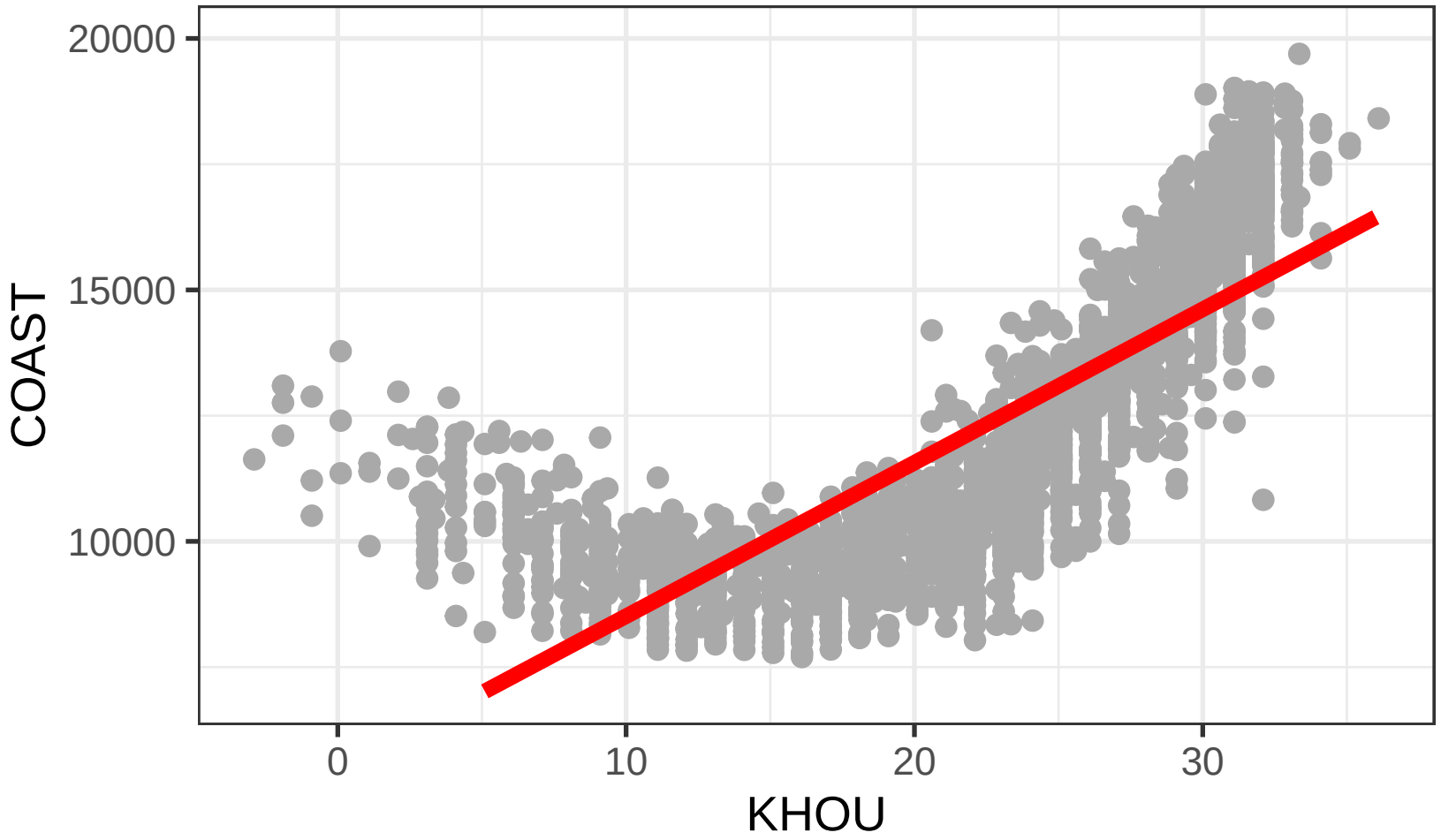
Measuring accuracy

- A simple principle: choose the model that makes the most "accurate" predictions, on average.
- A standard measure of (in)accuracy is the root mean-squared error:

$$\text{RMSE}_{\text{in}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2}$$

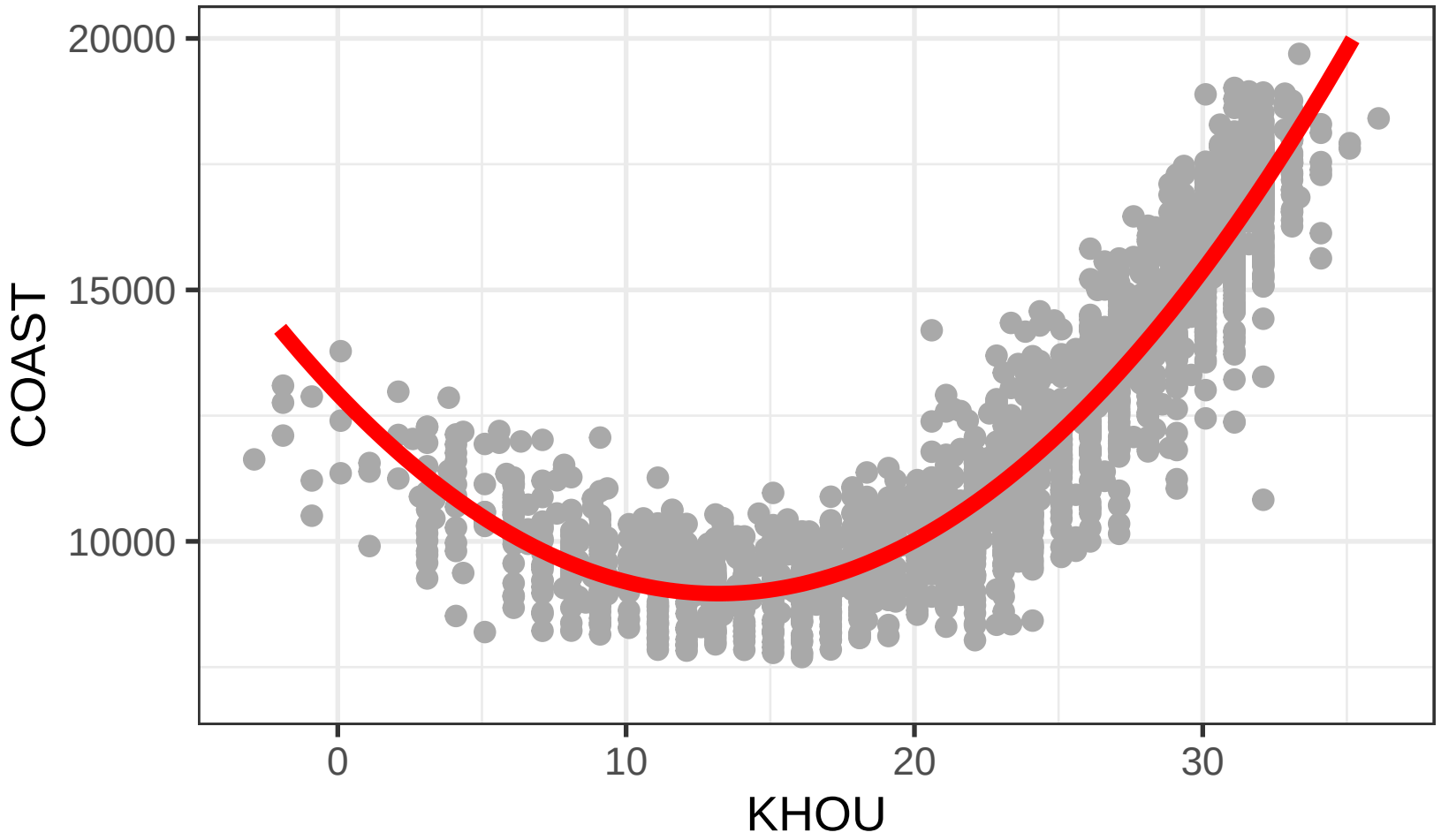
- This measures, on average, how large are the errors made by the model on the training data. (OLS minimizes this quantity over the set of linear functions.)
- RMSE is just one possible error metric, but it is pretty popular.

Measuring accuracy: linear vs. quadratric



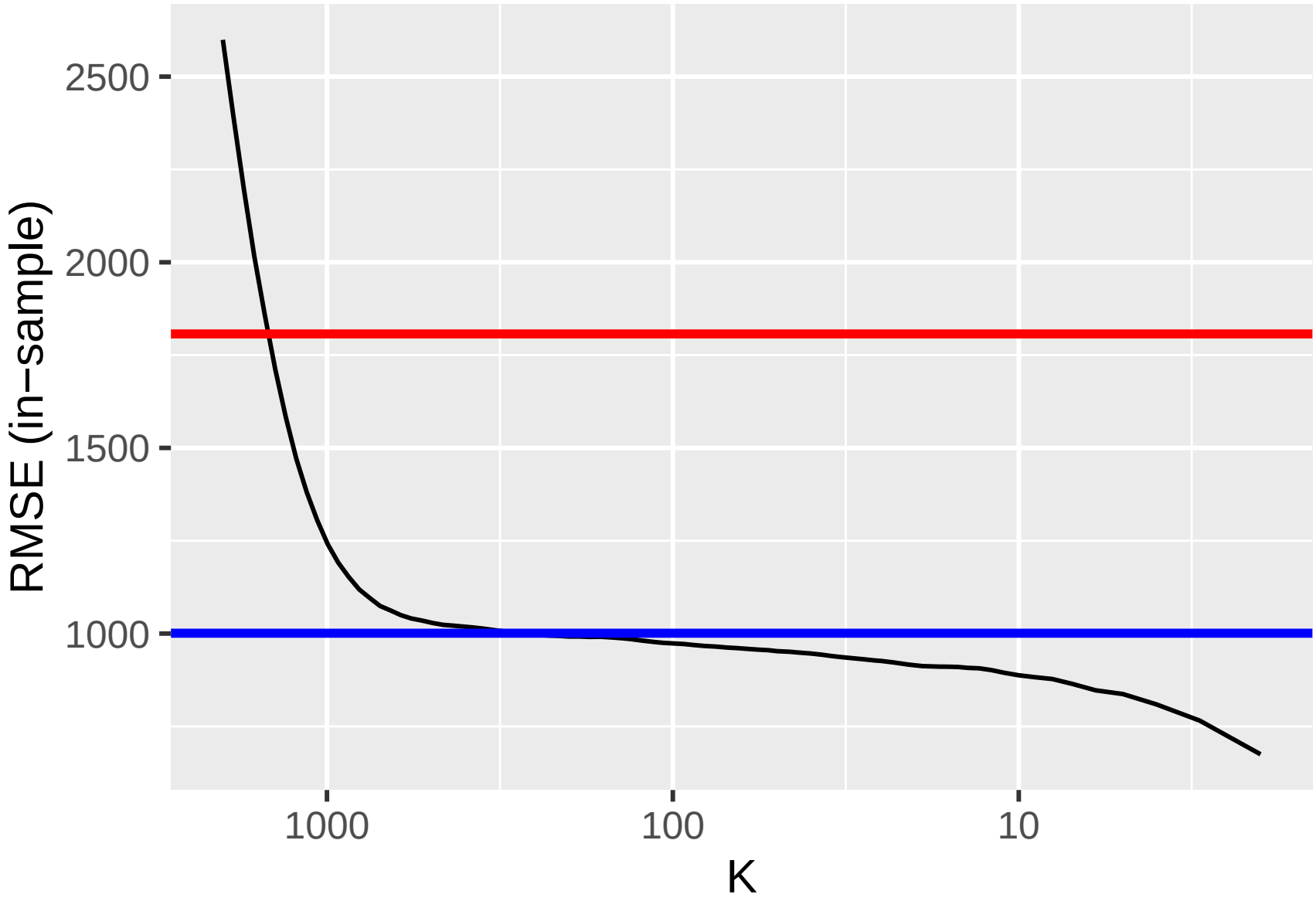
- $RMSE = 1807$

Measuring accuracy: linear vs. quadratric

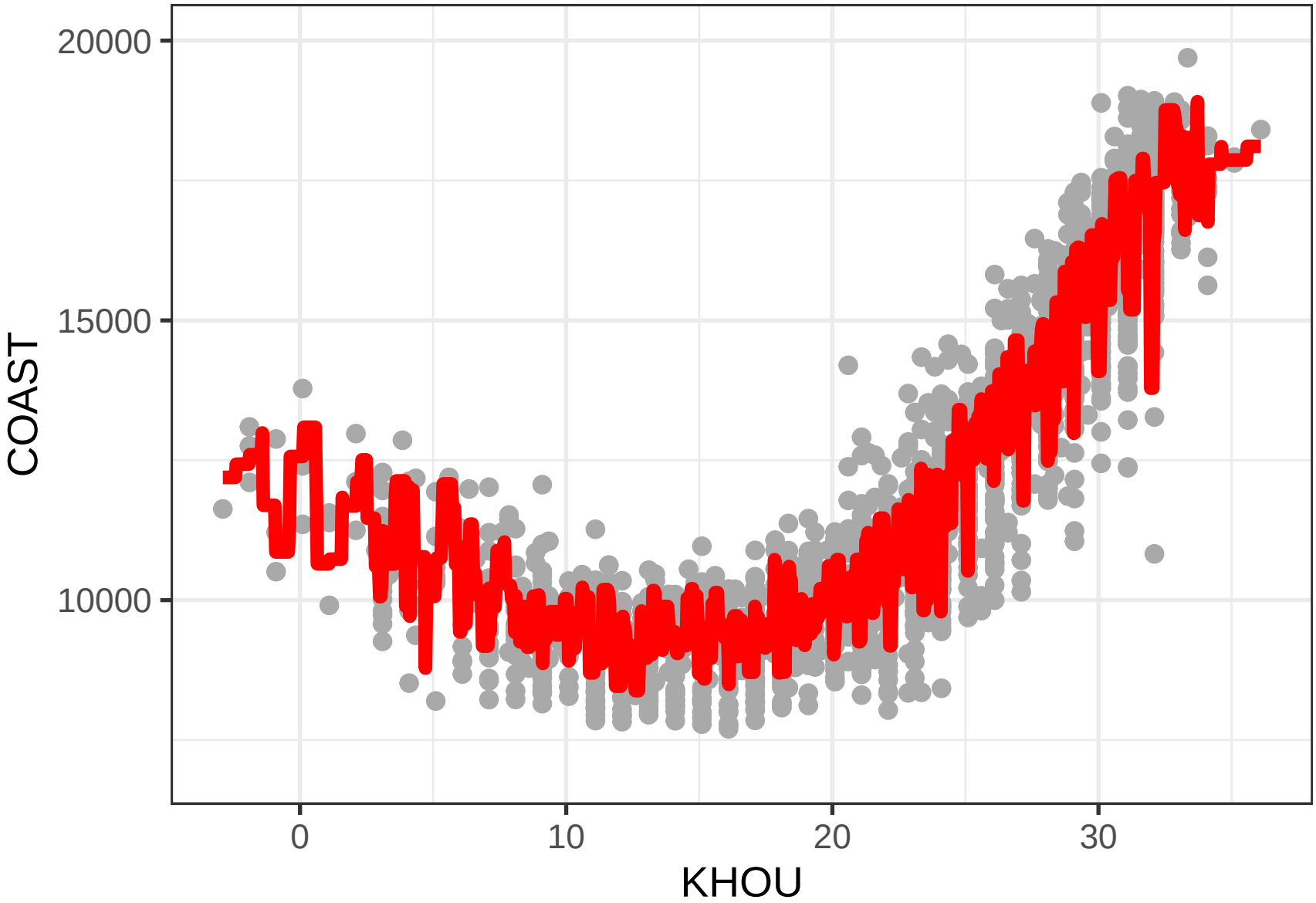


- $RMSE = 1001$

Measuring accuracy: RMSE of K for KNN



So we should pick K=2?



Overfitting

- Maximizing flexibility does not work. Empirically, it leads to a **terrible** prediction.
- Absorb all the idiosyncratic noise(ϵ) in the prediction model.
- A new observation with the same X will have a different idiosyncratic noise, and so the prediction is off.
- Remember: our aim is not to fit the model but to predict future.
- To avoid overfitting, we need to find **the optimal degree of flexibility**

Bias-variance trade-off

- Overfitting is an illustration of the **bias-variance trade-off** common to non-parametric statistics and econometrics.
- **Variance** refers to the degree by which $\hat{f}$ would change if we estimated on a different data set.
- **Bias** refers to the error of approximating a complex general function by a restricted functional form – this means that $f(\hat{x})$ systematically under-/overpredicts for regions of x.
- Flexibility reduces bias but increases variance. There is an optimal degree of flexibility.

Bias-variance trade-off

High K = high bias, low variance:

- We estimate $f(x)$ using many points, some of which might be far away from x. These far-away points bias the prediction; their values of $f(x)$ are slightly off on average.
- But more data points means lower variance—less chance of memorizing random noise.

Low K = low bias, high variance:

- We estimate $f(x)$ using only points that are *very close* to x. Far-away x points don't bias the prediction with their "slightly off" y values.
- But fewer data points means higher variance—more chance of memorizing random noise.

Bias-variance trade-off and the MSE

- For any r.v. X, we have $E(X^2) = Var(X) + (EX)^2$.Then we have

$$\begin{aligned} EMSE &= E[f(X) + \epsilon - \hat{f}(X)]^2 \\ &= Var[f(X) + \epsilon - \hat{f}(X)] + (E[f(X) + \epsilon - \hat{f}(X)])^2 \\ &= \underbrace{Var[\hat{f}(X)]}_{\textbf{var}(\hat{f}(x))} + \underbrace{Var(\epsilon)}_{\textbf{Irreducible error}} + \underbrace{(E[f(X) - \hat{f}(X)])^2}_{\textbf{bias}(\hat{f}(x))} \\ &= \textbf{Irreducible error} + \textbf{bias}(\hat{f}(x)) + \textbf{var}(\hat{f}(x)) \end{aligned}$$

- **Note**: $f(x)$ is not a random variable(function) but a predetermined function to represent the real relationship between X and Y. While $\hat{f}(x)$ is a random variable(function) because this is what we obtain by an estimate.

Bias-variance trade-off

That's why it's a trade-off!

- Smaller estimation variance generally requires a *less complex* model
  - intuitively, one that "wiggles less" from sample to sample.
- Smaller bias generally requires a *more complex* model
  - one that can "wiggle more," to adapt to the true function.
- Models that "wiggle more" can adapt to more kinds of functions, but they're also more prone to memorizing random noise.

Out-of-sample vs in sample

- Making good predictions about the past isn't very impressive. Our very complex (K=2) model earned a low RMSE by simply memorizing the random pattern of noise in the data.
- So we divide the whole sample into two subsets
  - **In sample** or **training data**: to fit the model
  - **Out-of-sample** or **testing data**: Additional data used to evaluate how good is the regression model fit(assume "future")
- Thus suppose we have data
  $(x_1, y_1), (x_2, y_2)...(x_n, y_n), (x_{n+1}, y_{n+1})...(x_{n+m}, y_{n+m})$
  - n in sample: $(x_1, y_1), (x_2, y_2)...(x_n, y_n)$
  - m out-of-sample: $(x_{n+1}, y_{n+1})...(x_{n+m}, y_{n+m})$

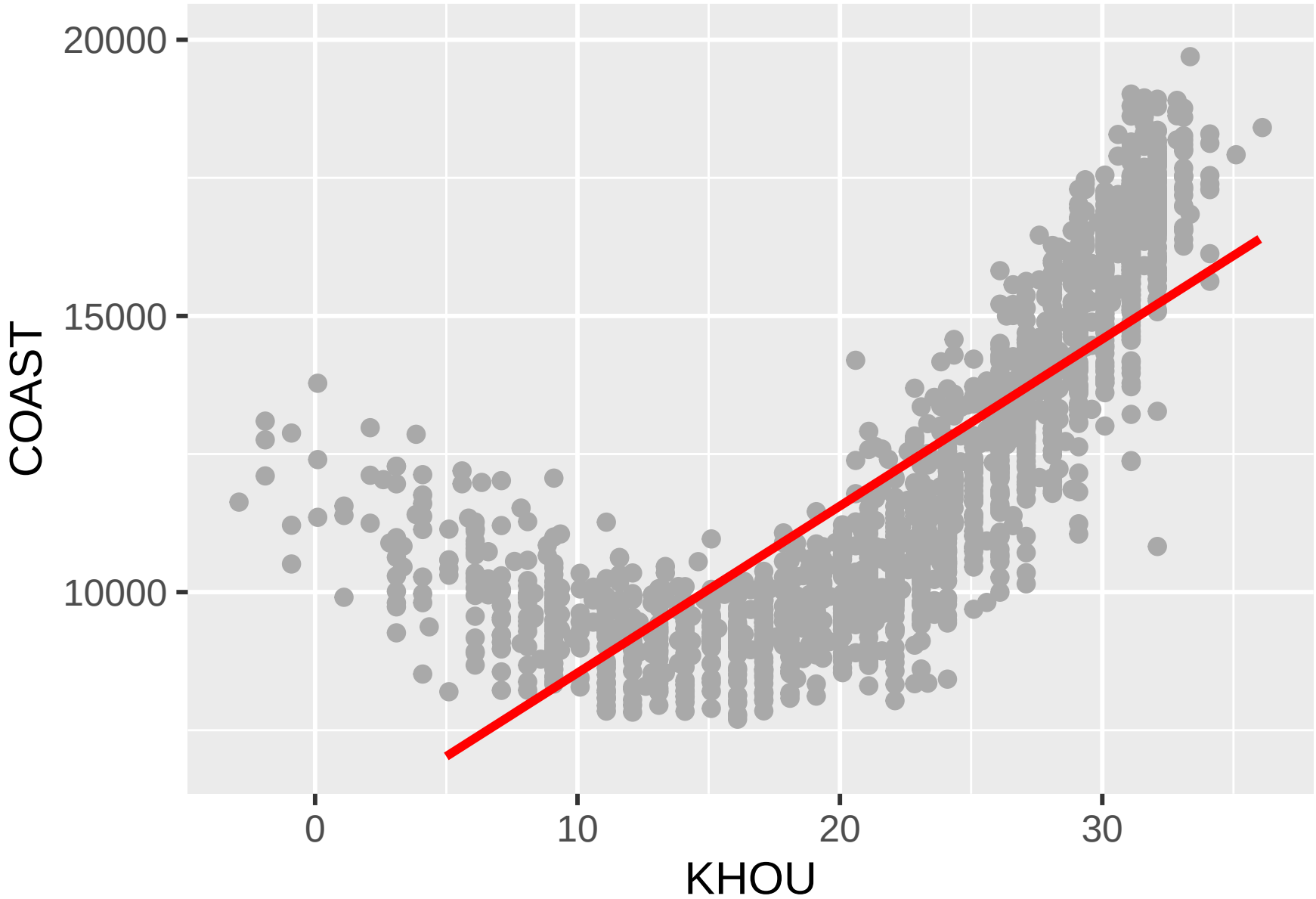
Out-of-sample accuracy

- Key idea: what really matters is our prediction accuracy out-of-sample!
- The only $RMSE$ we ever really care about is **out-of-sample**.
- The **out-of-sample** root mean-squared error is then:

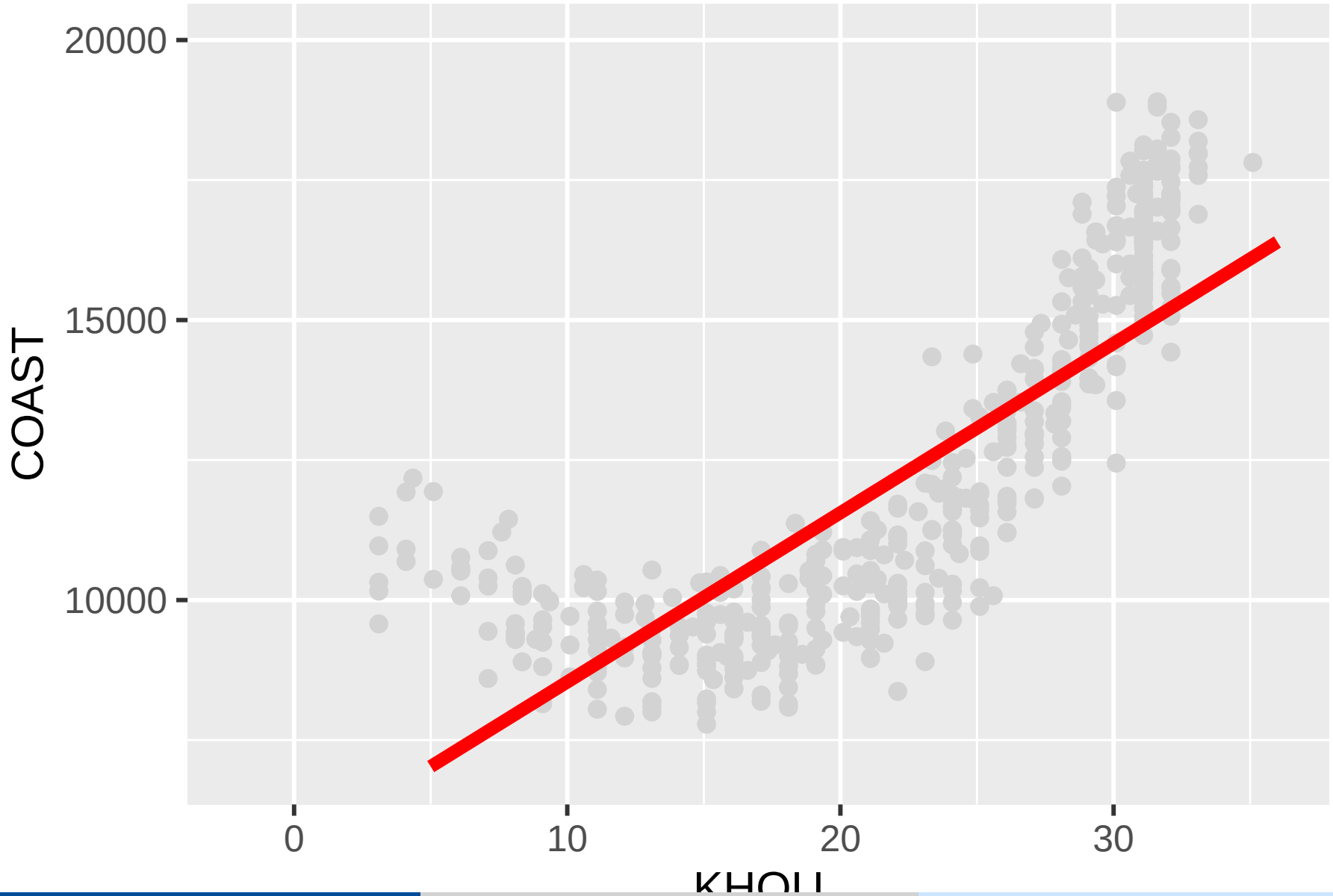
$$\text{RMSE}_{\text{out}} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - f(x_i))^2}$$

- Let's check performance across a variety of choices for K, versus the linear and quadratic models.

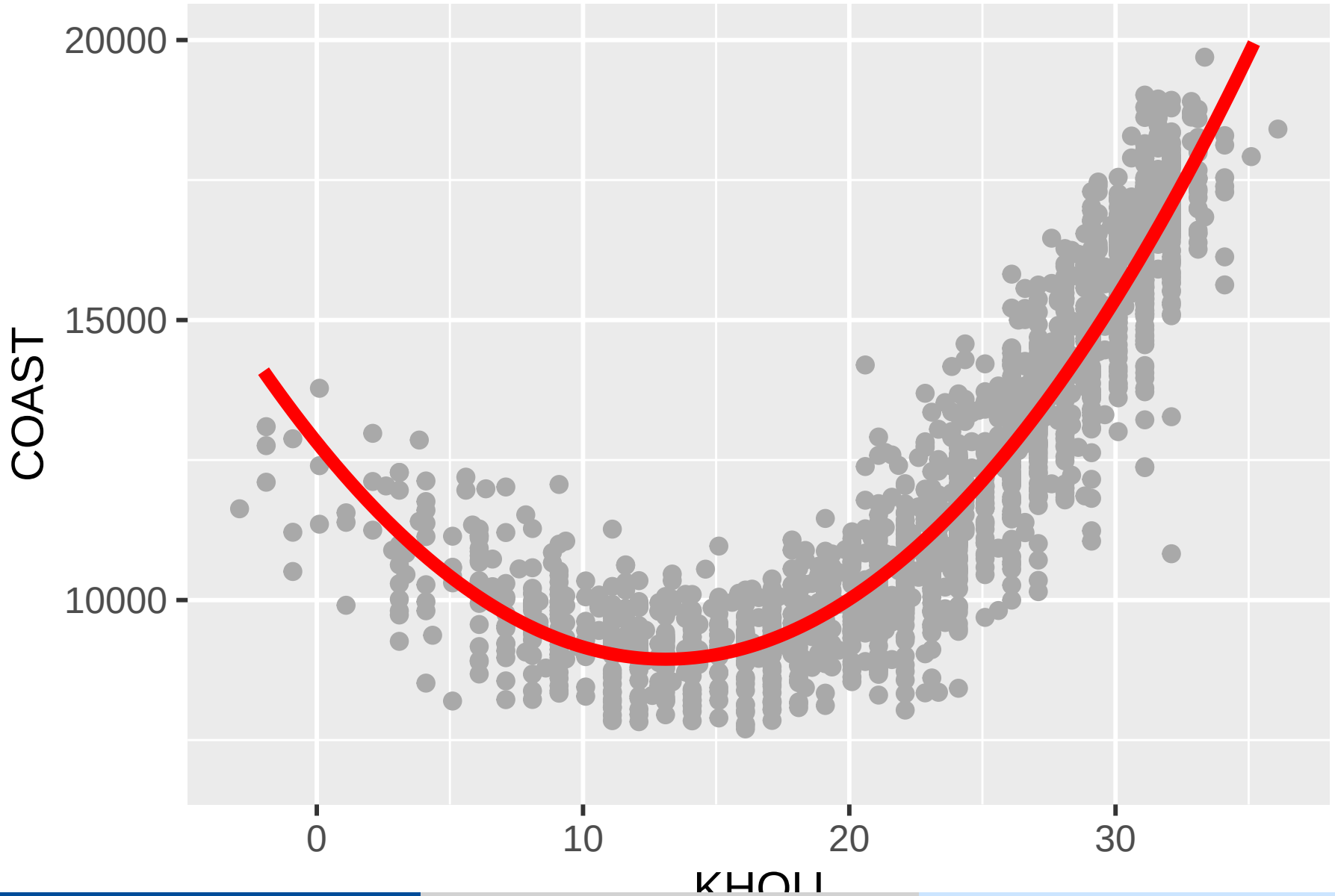
Linear model: train



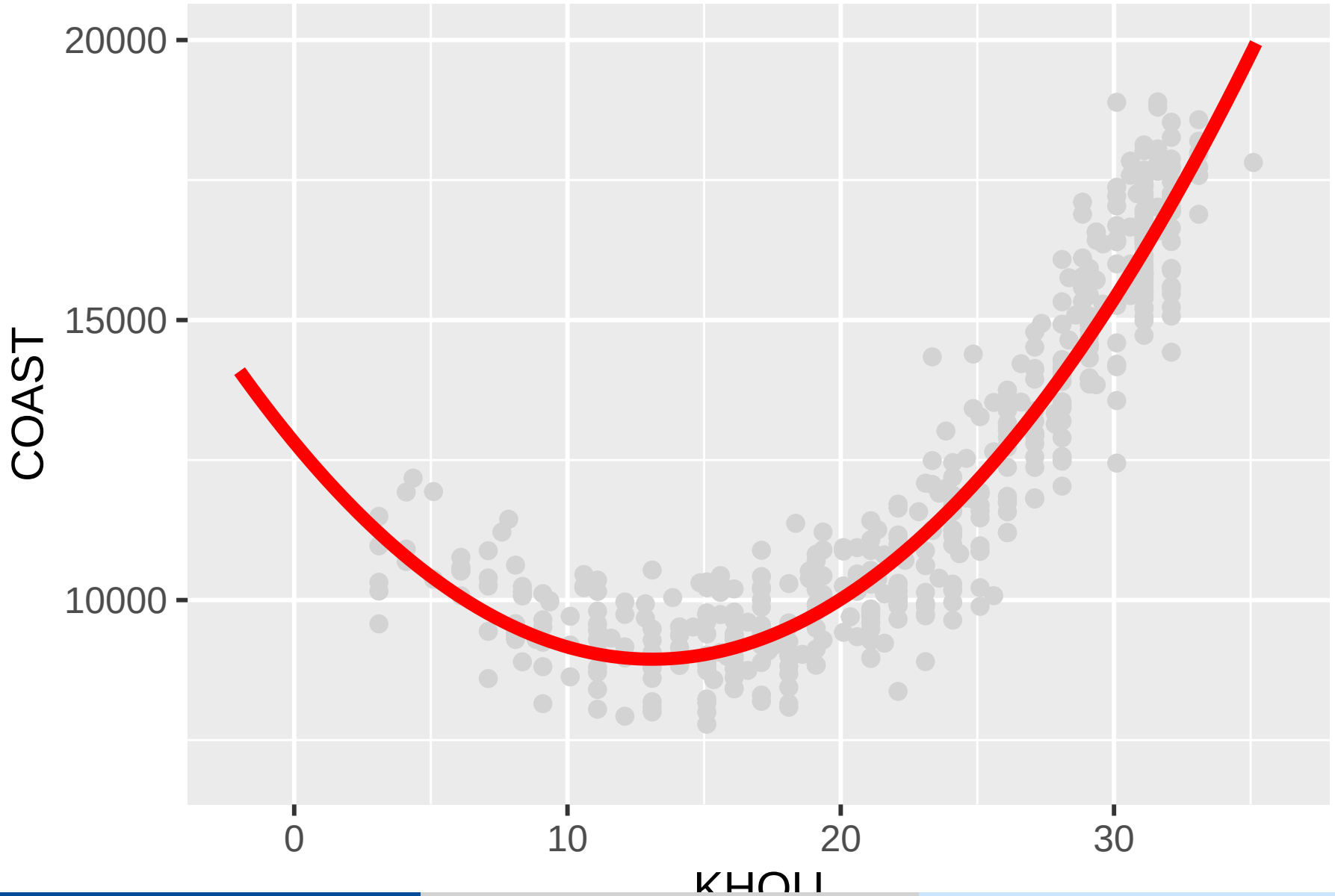
Linear model: test



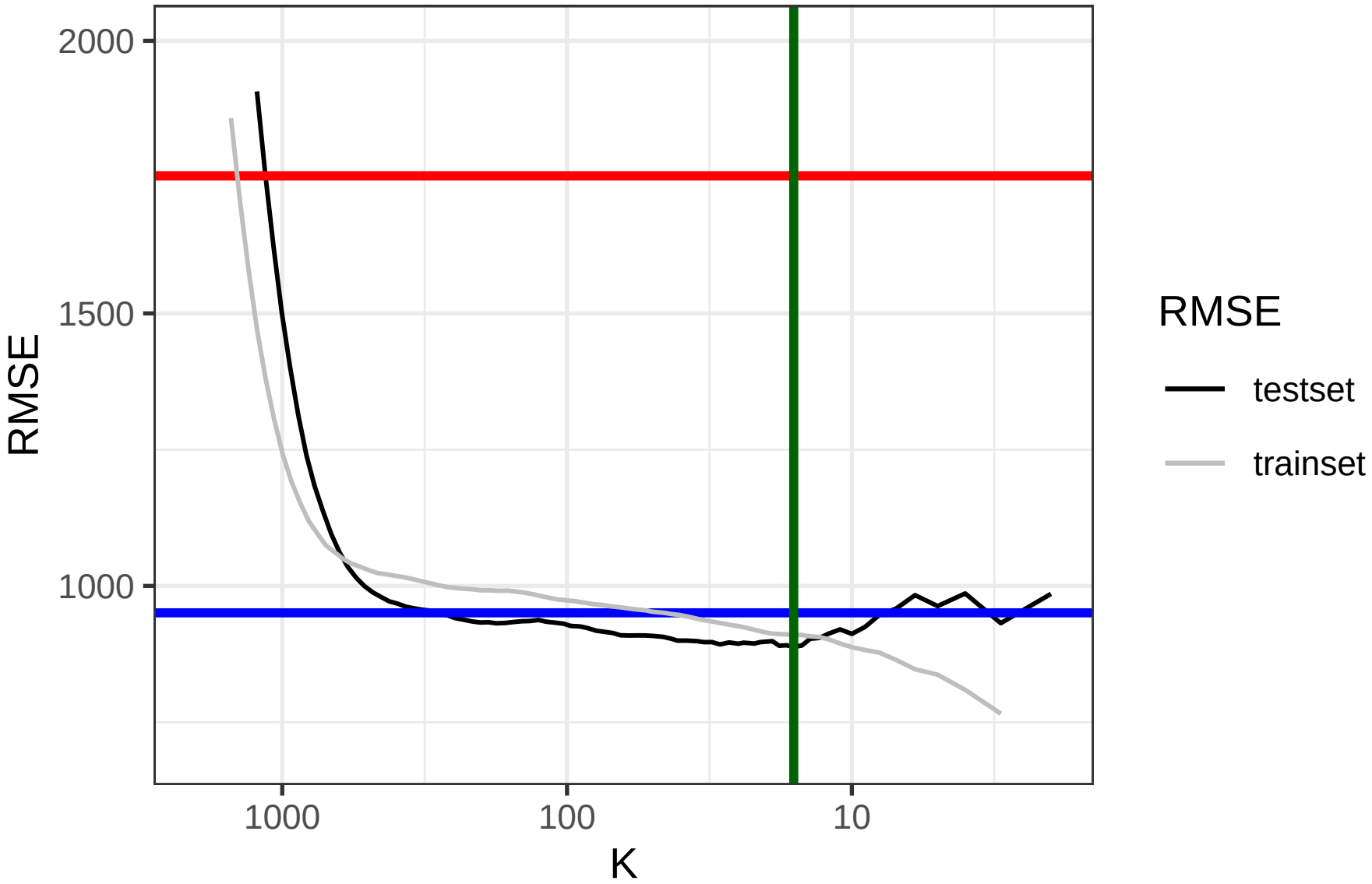
Quadratic model: train



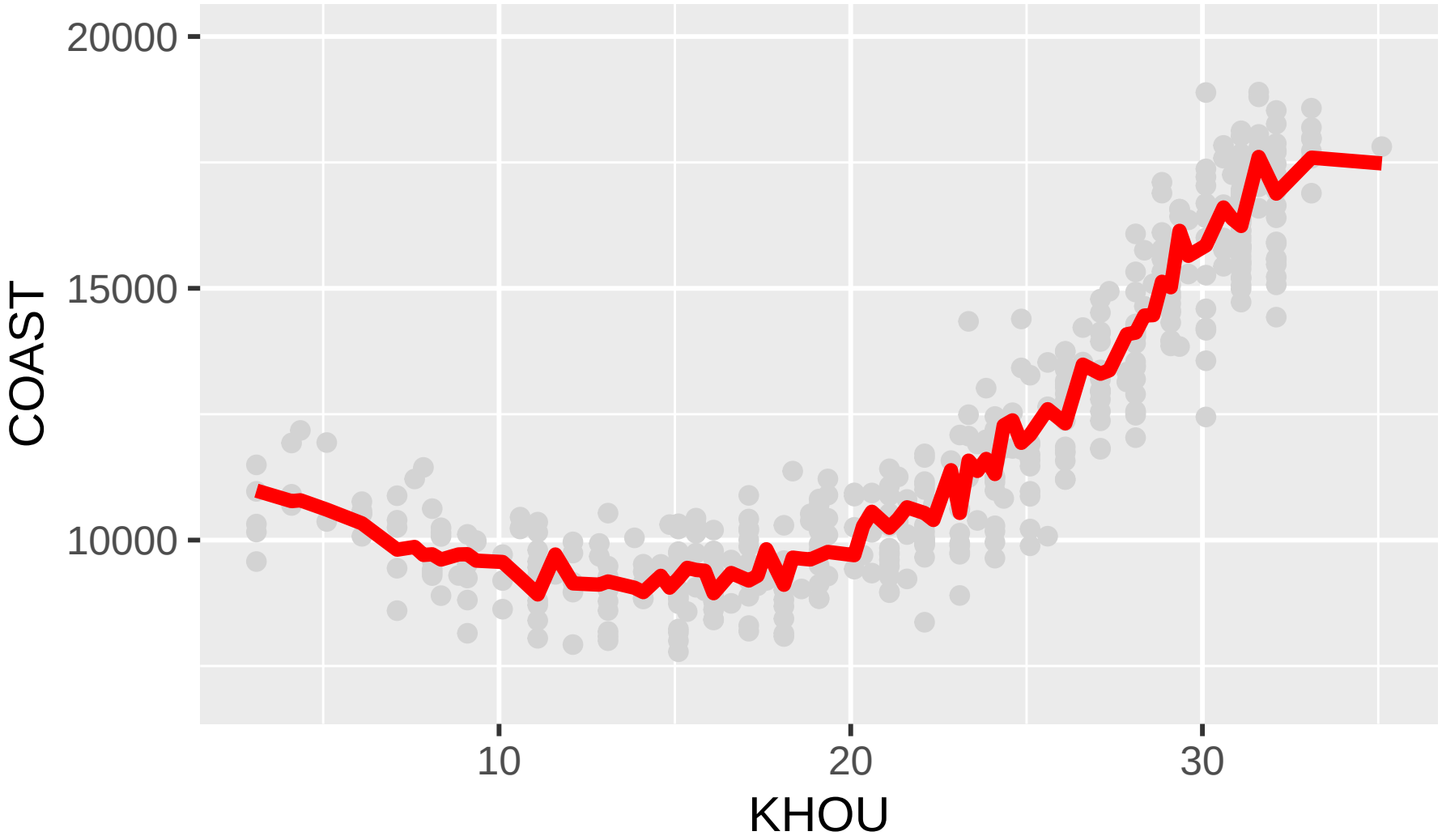
Quadratic model: test



K-nearest neighbors: test



K-nearest neighbors: test at the optimal k



RMSE: Linear v.s Quadratic vs.KNN

- Linear:

$$RMSE_{out} = 1752$$

- Quadratic

$$RMSE_{out} = 951$$

- K-Nearest Neighbors

$$\mathrm{RMSE}_{out} = 888$$

Measuring model accuracy, revisited

- Recall our definition of "true" out-of-sample MSE:

$$\text{EMSE}_{out} = E\left[\left(Y - \hat{f}(X)\right)^2\right]$$

- A simple way to estimate this quantity is to train our model $\hat{f}$ on in-sample and to calculate average performance on out-of-sample:

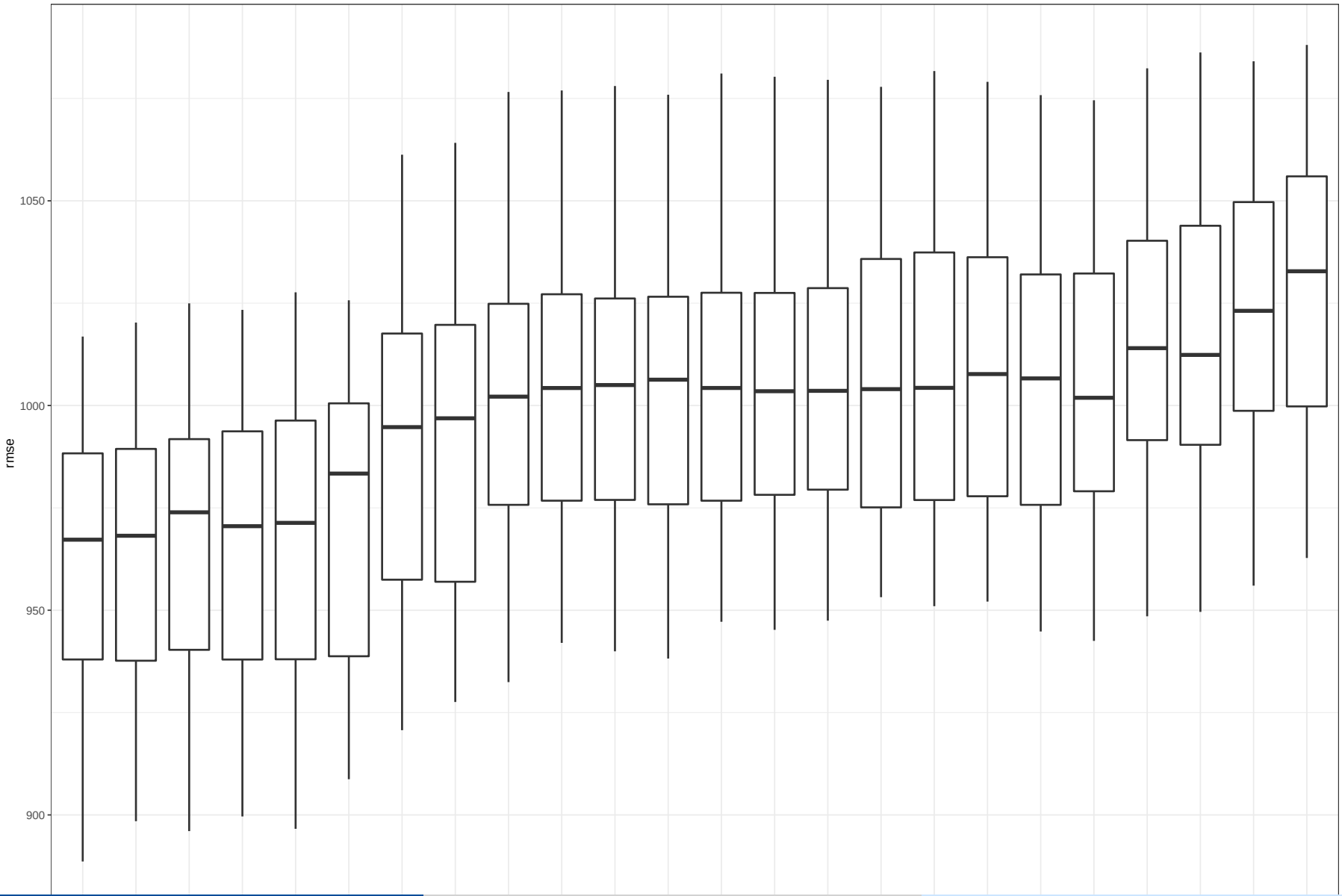
$$\widehat{\text{EMSE}} = \frac{1}{N_{\text{out}}} \sum_{i=1}^{N_{\text{out}}} \left(y_i - \hat{f}(x_i)\right)^2$$

- The key word here is **estimate.** There are two sources of randomness in our estimate:
  - $\hat{f}(x)$, the function estimate from in-sample.
  - the specific (x_i, y_i) pairs that end up in out-of-sample.

EMSE for 10 different random train/test splits

our-of-sample	RMSE
1	955.3252
2	894.4636
3	973.9200
4	956.9058
5	966.7150
6	952.9189
7	990.4689
8	931.7302
9	971.3224
10	983.6530

EMSE across multiple values of K



Measuring model accuracy, revisited

- Our $\widehat{\mathrm{EMSE}}$ differs from one train/test split to the next.
- This is intuitive: $\widehat{\mathrm{EMSE}}$ just an estimate of something unknown ($\mathrm{EMSE}^\star$), and all estimates of unknown quantities have variance.
- In fact, the variability of $\widehat{\mathrm{EMSE}}$ across different train/test splits *for fixed K* can be pretty large, compared to the differences across a wide range of K.
- Before we selected K using a single train/test split and picked the K with the lowest RMSE. **Should we modify our approach?**

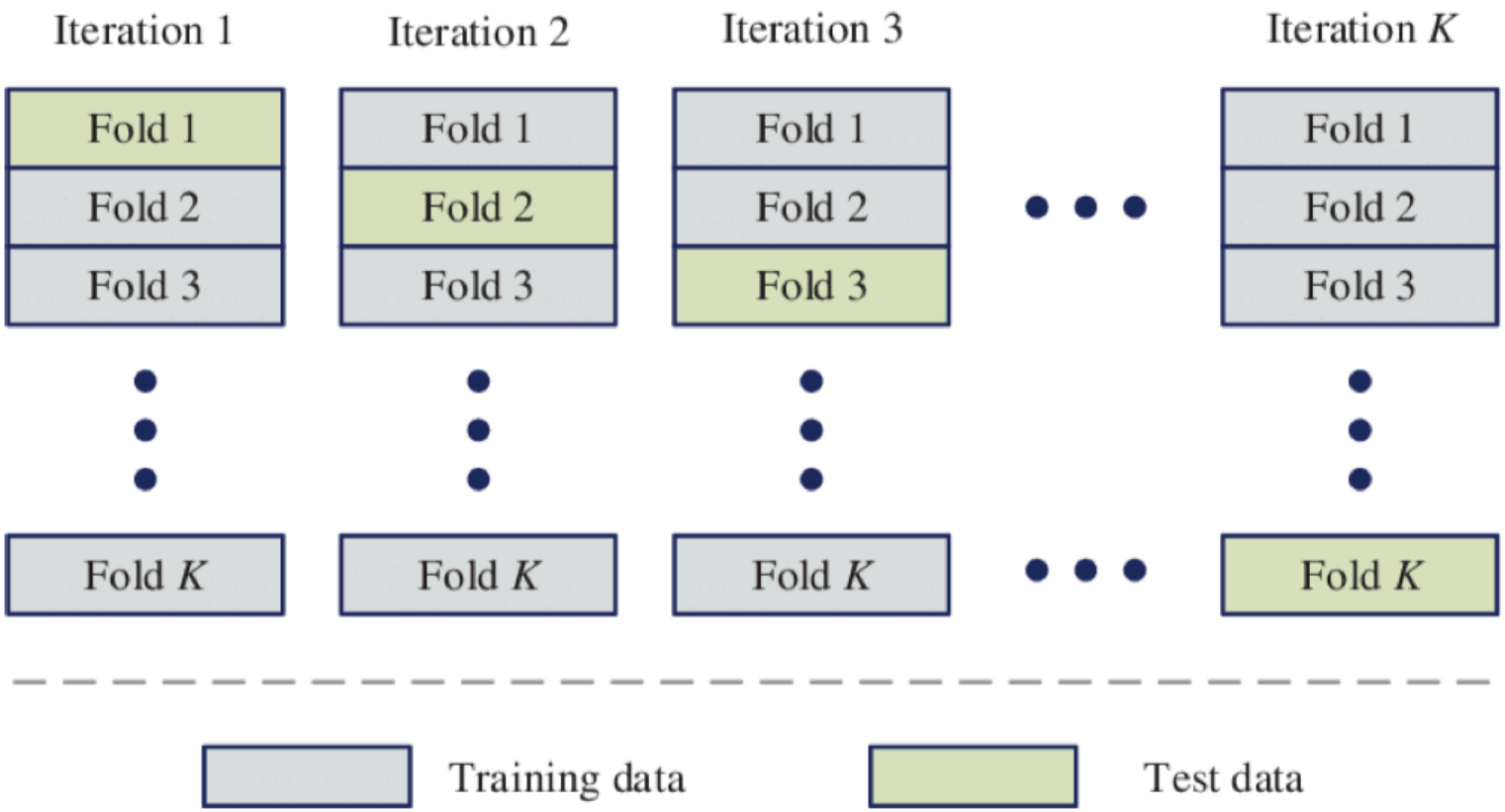
K-fold cross validation

A more efficient solution is K-fold cross-validation:

1. Randomly divide the data set into K **nonoverlapping** groups, or **folds**, of roughly equal size.
2. For fold $k = 1$ to K:
   - Fit the model using all data points **not in** fold k.
   - For all points (y_i, x_i) **in** fold k, predict $\hat{y}_i$ using the fitted model.
   - Calculate $\widehat{\mathrm{RMSE}}_k$, the average error on fold k.
3. Calculate the cross-validated error rate as:

$$\mathrm{CV}_{(K)} = \frac{1}{K} \sum_{k=1}^{K} \widehat{\mathrm{EMSE}}_k$$

K-fold cross validation



- The split of the data into folds is still random, but in a way that minimizes the overlap between each test set.

K-fold cross validation in practice

- Typical values of K are **5** or **10** in practice.
- All candidate models should be fit on the same set of folds.
  - That is, don't create a different split to evaluate different models.
- If $K = N$, i.e. the size of the data set, the resulting procedure is called **"leave-one-out" cross validation (LOOCV)**.

LOOCV vs K-fold CV

Note that LOOCV is a special case of k-fold cross-validation where $K = N$

- Generally k-fold CV with $K = 5$ or $K = 10$ is preferable to LOOCV.
- Then both training and holdout samples are of reasonable size, achieving a balance on both bias and variance.
- LOOCV tends to have a higher variance. This is because the N folds are highly correlated – any two folds always contain almost the same data points.

K-fold cross validation

There are two typical ways to select a model using cross validation:

1. The **min** rule: choose the model with the best cross-validated error.
2. The **1SE** rule: choose the simplest model whose cross-validated error is within one standard error of the minimum.

For each model, we estimate the standard error of that model's cross-validated EMSE as:

$$\text{S.E} \approx \frac{\text{sd}\left(\widehat{\text{EMSE}}_1, \widehat{\text{EMSE}}_2, \ldots, \widehat{\text{EMSE}}_K\right)}{\sqrt{K}}$$

Summary

- Nonparametric models (like KNN):
  - can't be written down in terms of simple math functions
  - can adapt to complex functions, but have to be reined in somehow, so that they don't overfit
  - usually have "tuning" parameters (like K in KNN) that govern the trade-off between bias and variance.
- Out-of-sample performance is the true test of a model:
  - In-sample RMSE is too optimistic, often wildly so.
  - Cross-validation using K=5 or K=10 is a practical way to quantify out-of-sample performance.
  - For close calls, the 1SE rule is a widely accepted way to actually pick the model.
  - Simplicity is a virtue, and the 1SE rule finds the simplest model that doesn't forfeit any statistically noticeable gains in performance.

Model Selection

Improved prediction accuracy

A linear model is generally thought of as a "high bias, low variance" kind of estimator, at least compared with alternatives we've seen (and others we have yet to see).

- But if the number of observations n is not much larger than the number of features p, even OLS can have high estimation variance.
- The result: overfitting and poor prediction accuracy.

One solution:

- *Shrinking* or *regularizing* the coefficient estimates so that they don't just chase noise in the training data.
- Extreme case: if $p > n$, there isn't even a unique OLS solution! No option but to do something else.

Model selection

The seemingly obvious approach is *exhaustive enumeration*:

- fit all possible models under to a training set
- measure the generalization error of each one on a testing set.
- choose the best one.

But this can be *too* exhausting. What are the limiting performance factors here?

- it might take a long time to fit each model (when?)
- if so, it will take even longer to repeatedly re-fit each model to multiple training sets
- and there might be way too many models to check them all.

Iterative model selection

Thus a more practical approach to model-building is *iterative*.

Start with a *working model.* The iterate the following steps:

1. Consider a limited set of "small" changes to the working model.
2. Measure the performance gains/losses resulting from each small change.
3. Choose the "best" small change to the working model. This becomes the new working model.
4. Repeat until you can't make any more changes that make the model better.

Iterative model selection

- Forward selection
- Backward selection
- Stepwise selection
- AIC and BIC

Different approaches to iterative model selection answer these questions in different ways but still be demanding.

Regularization:some intuition

- The key to modern statistical learning is **regularization**: departing from optimality to stabilize a system.
- two common penalties
- Ridge regression:
- Lasso regression:

Application: prediction using many predictors for test scores

- Predicting test scores for a school using variable describing the school, its students, and its community.
- the full data set consists of data gathered on 3932 elementary schools in CA in 2013
- The task is to use these data to develop a prediction model that will provide good out-of-sample predictions.
- The variable to be predicted is the average fifth-grade test score at the school.

Application: 817 predictors

TABLE 14.1 Variables in the 817-Predictor School Test Score Data Set

Main variables (38)	
Fraction of students eligible for free or reduced-price lunch	Ethnicity variables (8): fraction of students who are American Indian, Asian, Black, Filipino, Hispanic, Hawaiian, two or more, none reported
Fraction of students eligible for free lunch	Number of teachers
Fraction of English learners	Fraction of first-year teachers
Teachers' average years of experience	Fraction of second-year teachers
Instructional expenditures per student	Part-time ratio (number of teachers divided by teacher full-time equivalents)
Median income of the local population	Per-student expenditure by category, district level (7)
Student–teacher ratio	Per-student expenditure by type, district level (5)
Number of enrolled students	Per-student revenues by revenue source, district level (4)
Fraction of English-language proficient students	
Ethnic diversity index	
+ Squares of main variables (38)	
+ Cubes of main variables (38)	
+ All interactions of main variables $(38 \times 37/2 = 703)$	
Total number of predictors $= k = 38 + 38 + 38 + 703 = 817$	

Ridge Regression

Ridge regression and OLS regression

- Recall in OLS regression, our model is

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + u_i$$

- Least-squares regression finds $\hat{\beta}_j$ by minimizing SSR

$$\begin{aligned}\min_{\hat{\beta}} \text{SSR} &= \min_{\hat{\beta}} \sum_{i=1}^{n} \hat{u}_i^2 \\ &= \min_{\hat{\beta}} \sum_{i=1}^{n} \left(Y_i - \underbrace{\left[\hat{\beta}_1 X_{i,1} + \cdots + \hat{\beta}_p X_{i,p}\right]}_{=\hat{Y}_i} \right)^2\end{aligned}$$

- **Ridge regression** makes a small change
  - adds a shrinkage penalty: the sum of squared coefficents $\left(\lambda \sum_j \beta_j^2\right)$
  - minimizes the (weighted) sum of SSR and the shrinkage penalty.

Ridge regression and OLS regression

- The ridge regression estimator minimizes the **penalized sum of squared residuals** $SSR^{ridge}(b)$

$$\min_{\hat{\beta}} R^{ridge} = \min_{\hat{\beta}} \sum_{i=1}^{n} \left(Y_i - \underbrace{\left[\hat{\beta}_1 X_{i,1} + \cdots + \hat{\beta}_p X_{i,p} \right]}_{=\hat{y}_i} \right)^2 + \lambda_R \sum_{j=1}^{p} \beta_j^2$$

- λ (≥ 0) is a tuning parameter for the harshness of the penalty.
- $\lambda = 0$ implies no penalty: back to OLS.
- Each value of λ produces a new set of coefficients.
- Ridge's approach to the bias-variance tradeoff: Balance
  - reducing SSR, *i.e.*, $\sum_i \left(Y_i - \hat{Y}_i \right)^2$
  - reducing coefficients'magnitudes.
  - λ determines how much ridge "cares about" these two quantities.

Ridge regression estimator

- When $k = 1$, based on F.O.C, then

$$\frac{\partial SSR^{ridge}}{\partial b} = -2\sum_{i=1}^{n} X_i \left(Y_i - bX_i\right) + 2\lambda b = 0$$

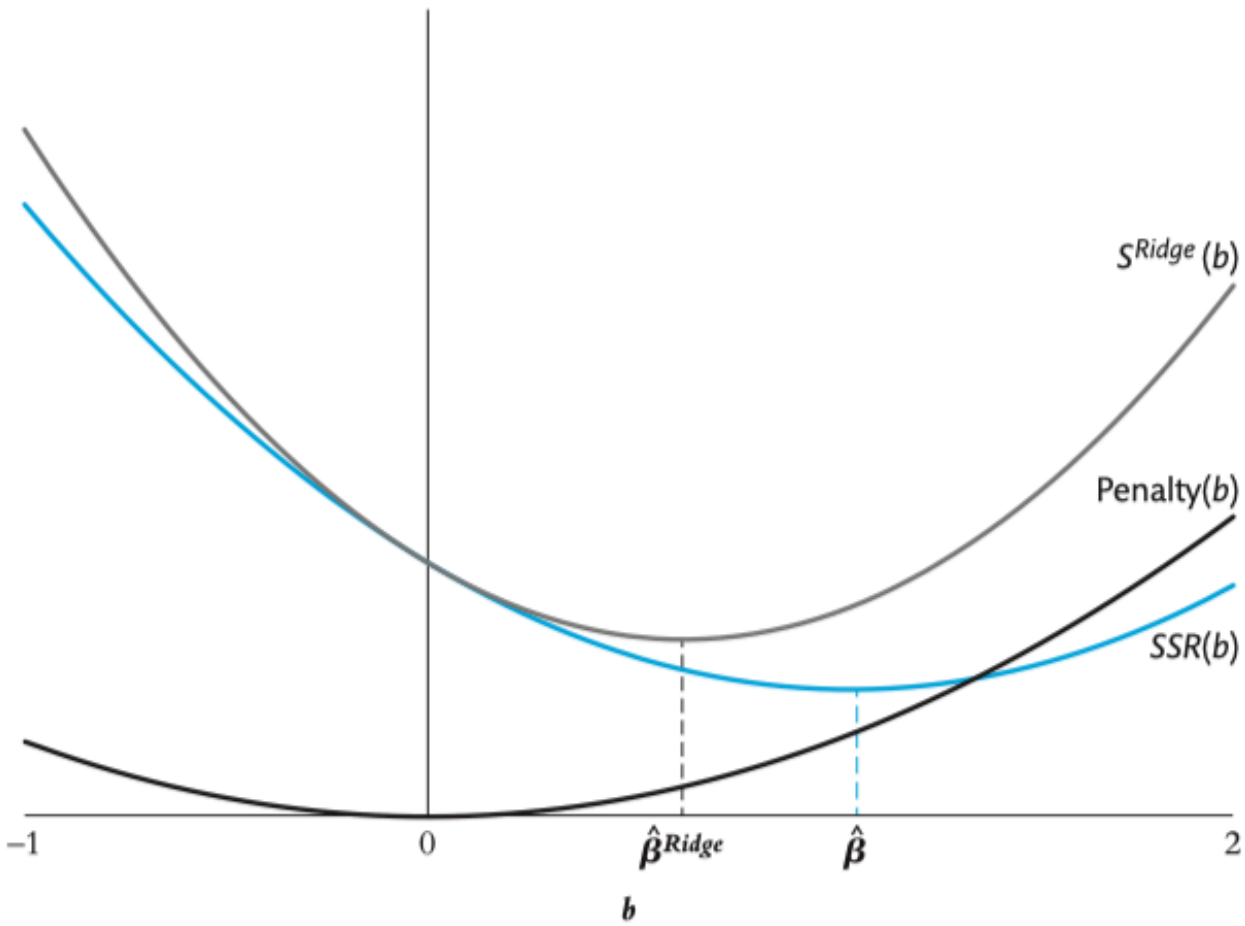
- Then we have Ridge regression estimator

$$\begin{aligned}
\hat{\beta}^{\text{Ridge}} &= \frac{\sum_{i=1}^{n} X_i Y_i}{\left(\sum_{i=1}^{n} X_i^2 + \lambda\right)} \\
&= \left(\frac{1}{1 + \lambda / \sum_{i=1}^{n} X_i^2}\right) \frac{\sum_{i=1}^{n} X_i Y_i}{\sum_{i=1}^{n} X_i^2} \\
&\equiv \left(\frac{1}{1 + \lambda / \sum_{i=1}^{n} X_i^2}\right) \hat{\beta} < \hat{\beta}
\end{aligned}$$

Ridge regression estimator

FIGURE 14.1 Components of the Ridge Regression Penalty Function

The ridge regression estimator minimizes $S^{Ridge}(b)$, which is the sum of squared residuals, $SSR(b)$, plus a penalty that increases with the square of the estimated parameter. The SSR is minimized at the OLS estimator, $\hat{\beta}$. Including the penalty shrinks the ridge estimator, $\hat{\beta}^{Ridge}$, toward 0.



How to obtain the Ridge Shrinkage Parameter λ

- Can we use the F.O.C to obtain it as we did for β No.
  - Then the optimal value of λ will be ZERO.
- Instead, it can be chosen by minimizing the **m-fold cross-validated** estimate of the RMSE.
  - Choose some value of λ , and estimate the MSPE by m-fold cross-validation
  - Repeat for many values of λ, and choose the one that yields **the lowest** RMSE.

Penalization and standardization

- Note: Scale of X can drastically affect ridge regression results.
- Because the scale of X will affect $\hat{\beta}$ and ridge is very sensitive to β.
  - Ridge regression pays a much larger penalty for different β
- Therefore, you have to **standardize** variables firstly before you use ridge regression.

Application: Predicting School-level test scores

- λ estimated by minimizing the 10-fold cross-validated RMSE.

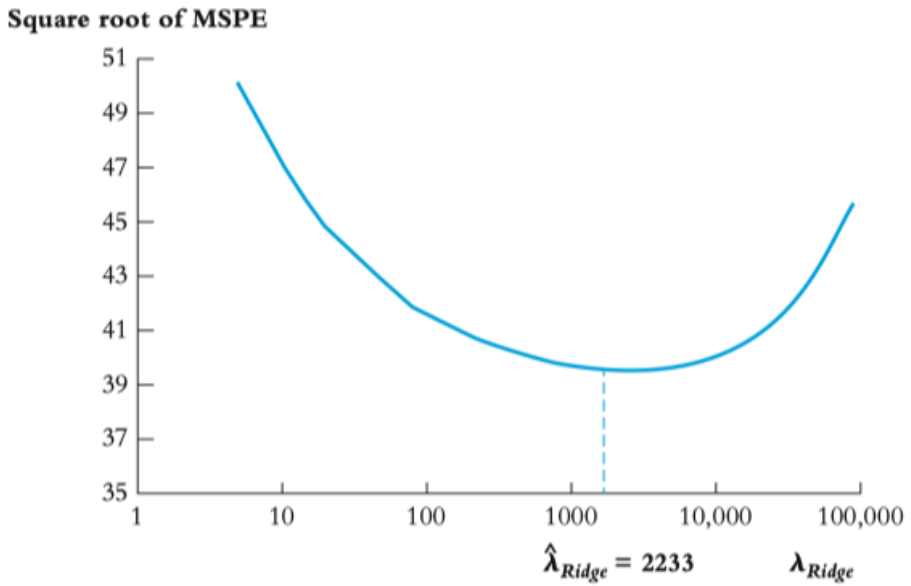
$$\hat{\lambda} = 2233$$

- Ridge have a smaller RMSE than OLS

$$RMSE_{ols} = 78.2; RMSE_{ridge} = 39.5$$

FIGURE 14.2 Square Root of the MSPE for the Ridge Regression Prediction as a Function of the Shrinkage Parameter (Log Scale for λ_{Ridge})

The MSPE is estimated using 10-fold cross validation for the school test score data set with $k = 817$ predictors and $n = 1966$ observations.



Lasso

Introduction

- Least Absolute Shrinkage and Selection Operator(LASSO): it simply replaces ridge's squared coefficients with **absolute values**.

$$\min_{\hat{\beta}} R^{lasso} = \min_{\hat{\beta}} \sum_{i=1}^{n} \left(Y_i - \underbrace{\left[\hat{\beta}_1 X_{i,1} + \cdots + \hat{\beta}_p X_{i,p} \right]}_{=\hat{y}_i} \right)^2 + \lambda_L \sum_{j=1}^{p} |\beta_j|$$

- where $\lambda_L \sum_{j=1}^{p} |\beta_j|$ is the penalty.

Lasso estimator

- Unlike ridge, lasso's penalty does not increase with the size of β.
- The only way to avoid lasso's penalty is to set βs to zero.
- This feature has two benefits
  1. Some coefficients will be set to zero.
  2. Lasso can be used for subset/feature selection.

Estimate the Lasso estimator

- For simplicity, $p = 1$

$$R^{lasso} = \sum_{i=1}^{n} \left(Y_i - \hat{\beta} X_{i,1}\right)^2 + \lambda_L |\beta|$$

- Suppose $\hat{\beta} > 0$, then

$$R^{lasso} = \sum_{i=1}^{n} \left(Y_i - \hat{\beta} X_{i,1}\right)^2 + \lambda_L \beta$$

- Suppose $\hat{\beta} < 0$, then

$$R^{lasso} = \sum_{i=1}^{n} \left(Y_i - \hat{\beta} X_{i,1}\right)^2 - \lambda_L \beta$$

- Then F.O.C

$$(-2)\sum_{i=1}^{n} X_i \left(Y_i - \hat{\beta}_{\text{Lasso}}\, X_i\right) + \lambda_{\text{Lasso}} = 0$$

$$\Rightarrow \sum_{i=1}^{n} X_i^2 \hat{\beta}_{\text{Lasso}} - \sum_{i=1}^{n} X_i Y_i = -\frac{1}{2}\lambda_{\text{Lasso}}$$

$$\Rightarrow \hat{\beta}_{\text{Lasso}} = \frac{\sum_{i=1}^{n} X_i Y_i}{\sum_{i=1}^{n} X_i^2} - \frac{1}{2}\frac{\lambda_{\text{Lasso}}}{\sum_{i=1}^{n} X_i^2} = \hat{\beta}_{ols} - \frac{1}{2}\frac{\lambda^{\text{Lasso}}}{\sum_{i=1}^{n} X_i^2}$$

- Because we suppose $\hat{\beta} > 0$, then

$$\hat{\beta}_{\text{Lasso}} = \max\left(\hat{\beta} - \frac{1}{2}\frac{\lambda_{\text{Lasso}}}{\sum_{i=1}^{n} X_i^2}, 0\right) \text{ when } \hat{\beta} > 0$$

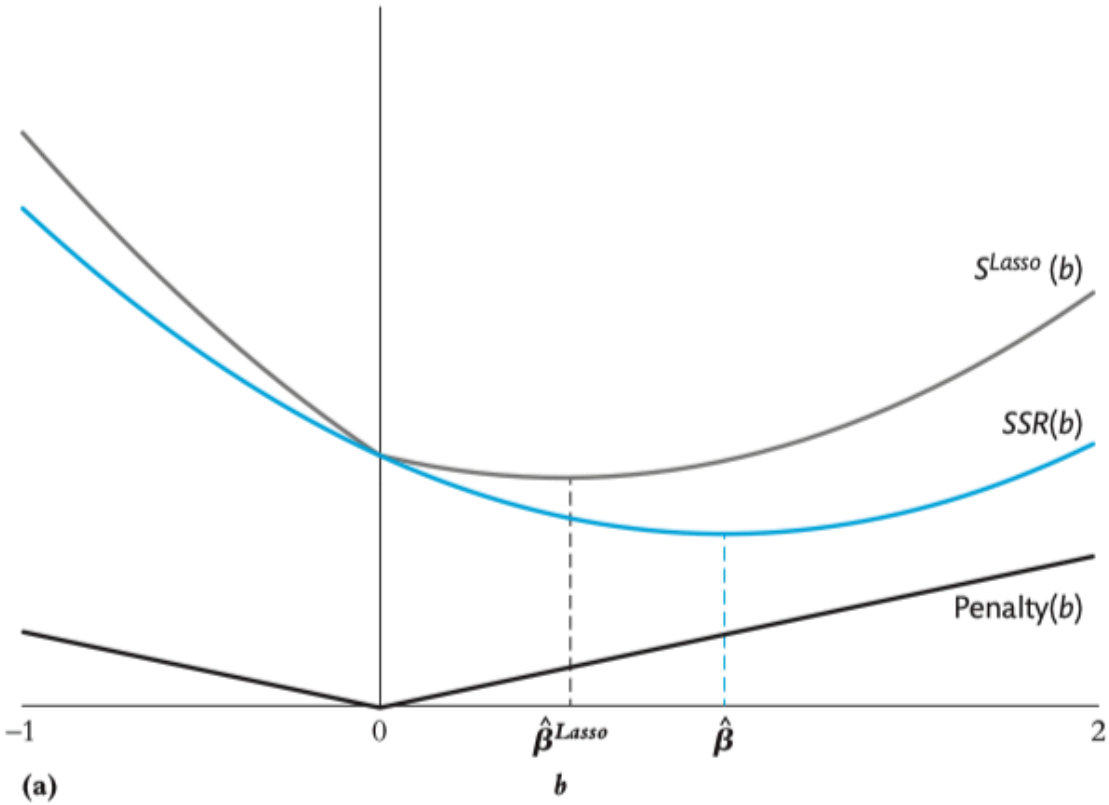
- Similar reasoning shows that when $\hat{\beta} < 0$

$$\hat{\beta}_{\text{Lasso}} = \min\left(\hat{\beta} + \frac{1}{2}\frac{\lambda_{\text{Lasso}}}{\sum_{i=1}^{n} X_i^2}, 0\right) \text{ when } \hat{\beta} < 0$$

The Lasso estimator

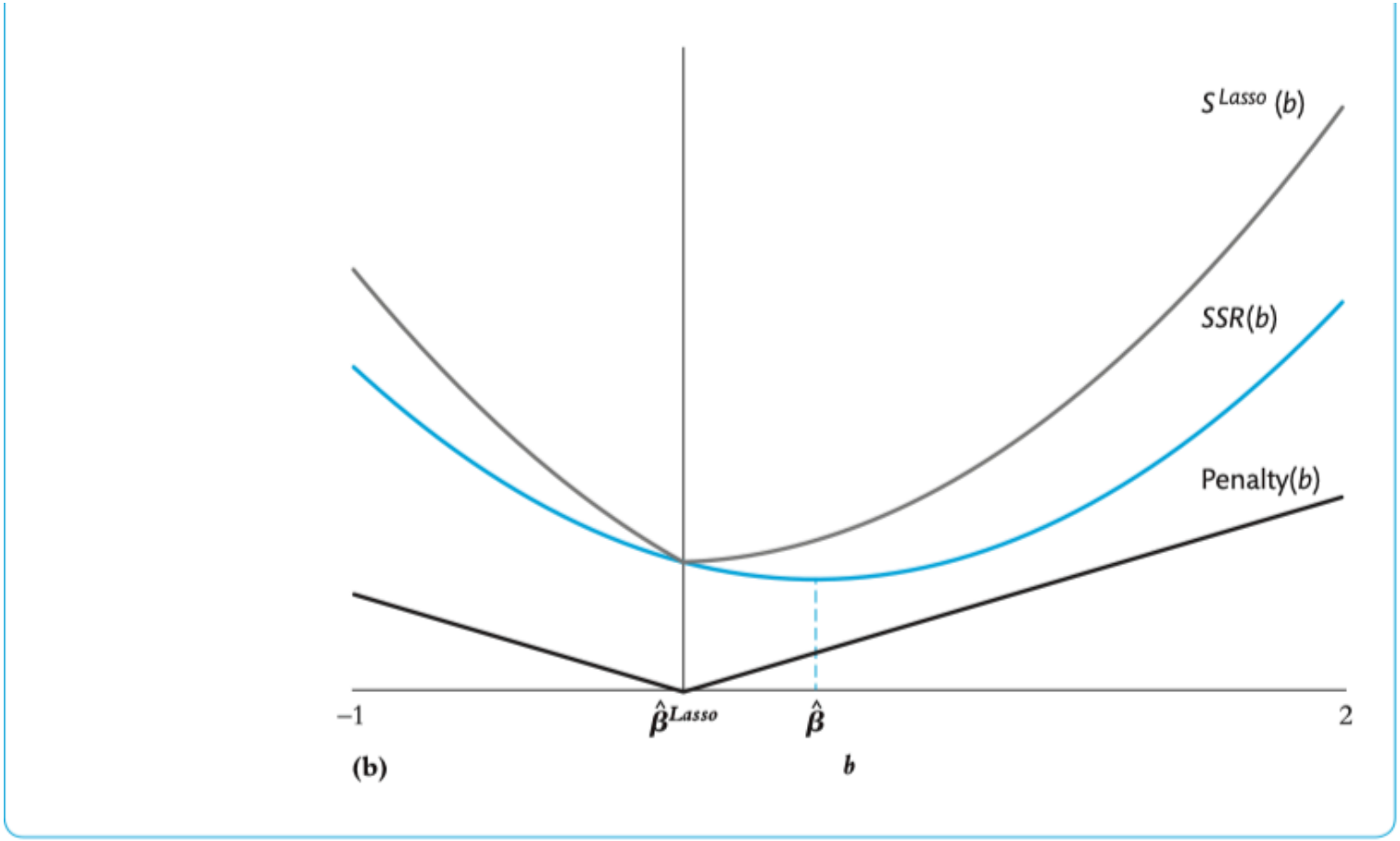
FIGURE 14.3 The Lasso Estimator Minimizes the Sum of Squared Residuals Plus a Penalty That Is Linear in the Absolute Value of b

For a single regressor, (a) when the OLS estimator is far from zero, the Lasso estimator shrinks it toward 0; (b) when the OLS estimator is close to 0, the Lasso estimator becomes exactly 0.



- When the OLS estimator is large, the Lasso estimator shrinks it slightly towards zero— less than ridge.

The Lasso estimator



- But when the OLS estimator is small, the Lasso estimator shrinks it all the way to zero, so that the Lasso estimator is exactly zero.

The Lasso estimator

- Lasso sets many βs to ZERO, which means "select" some useful predictors for prediction and drops the others.
- It means that Lasso can work especially well when in reality many of the predictors are **irrelevant**.
- Models in which most of the true β are zero are called **sparse models**.

Application: Predicting School-level test scores

- λ estimated by minimizing the 10-fold cross-validated RMSE.

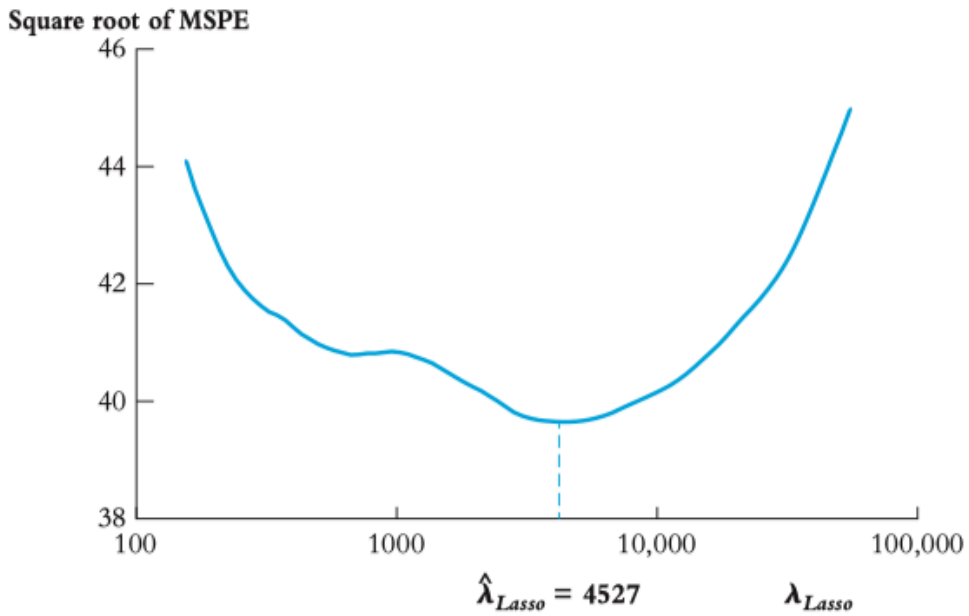
$$\hat{\lambda}_{Lasso} = 4527$$

- Only use $p = 56$ predictors and Lasso have a smaller RMSE than OLS

$$RMSE_{Lasso} = 39.7$$

FIGURE 14.4 Square Root of the MSPE for the Lasso Prediction as a Function of the Lasso Shrinkage Parameter (Log Scale for λ_{Lasso})

The MSPE is estimated by 10-fold cross validation using the school test score data set with $k = 817$ predictors and $n = 1966$ observations.



Principal Components

Introduction

- Ridge and Lasso reduce the RMSE by shrinking (biasing) the estimated coefficients to zero.
- In the case of Lasso, by eliminating many of the regressors entirely.
- Instead, Principal components regression(PCR) collapses the very many predictors(k) into a much smaller number(p) of linear combinations of the predictors.
- These linear combinations – called the principal components of X— are computed so that they capture as much of the variation in the original X's as possible.
- Because the number p of principal components is small, OLS can be used, with the principal components as (new) regressors.

Principal Components(k=2)

- The easy way to combine X_1 and X_2 is a linear equation

$$aX_1 + bX_2$$

- **Q** : What values of a and b should be used?
- The **Principal Components** solution is to choose a and b to solve

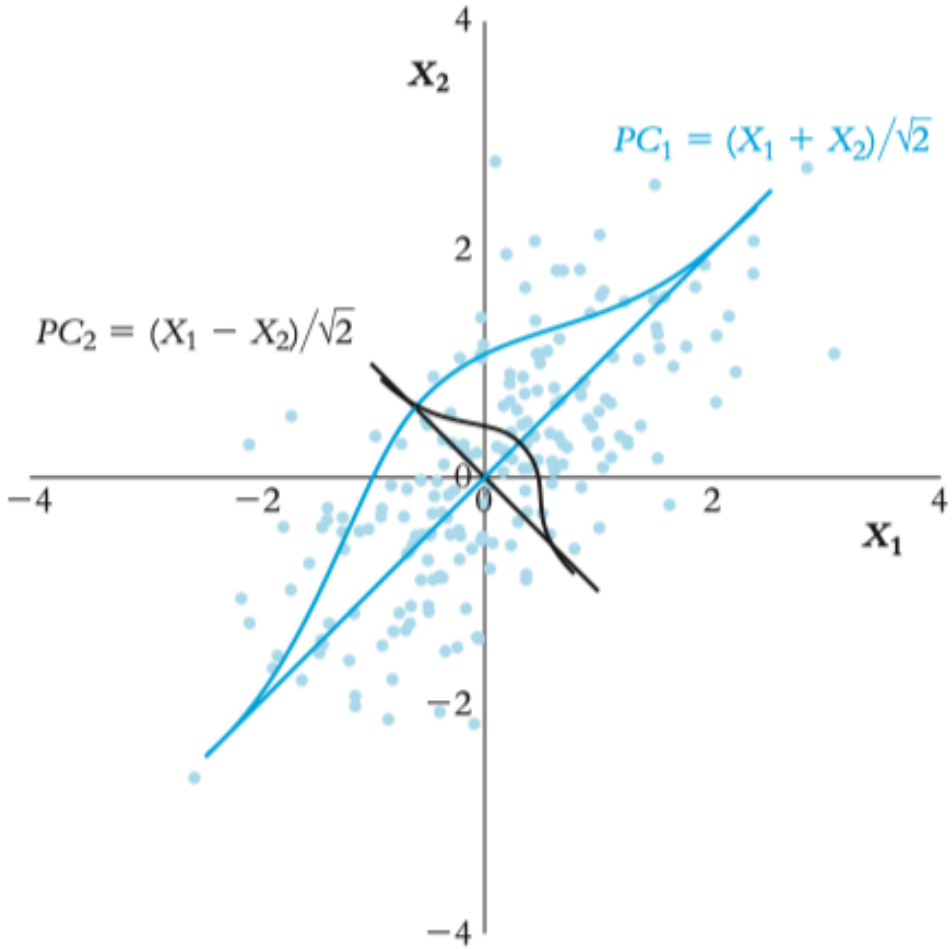
$$\max \mathrm{Var}\left(aX_1 + bX_2\right), \text{subject to } a^2 + b^2 = 1$$

- For X_1 and X_2 are positively correlated, then $a = b = \frac{1}{\sqrt{2}}$, the first principal component(PC_1) is $(X_1 + X_2)/\sqrt{2}$
- the second principal component(PC_2) is $(X_1 - X_2)/\sqrt{2}$, which is uncorrelated with the first.
- The principal component weights are normalized so that the sum of squared weights adds to 1.

Principal Components(k=2)

FIGURE 14.5 Scatterplot of 200 Observations on Two Standard Normal Random Variables, X_1 and X_2, with Population Correlation 0.7

The first principal component (PC_1) maximizes the variance of the linear combination of these variables, which is done by adding X_1 and X_2. The second principal component (PC_2) is uncorrelated with the first and is obtained by subtracting the two variables. The principal component weights are normalized so that the sum of squared weights adds to 1.



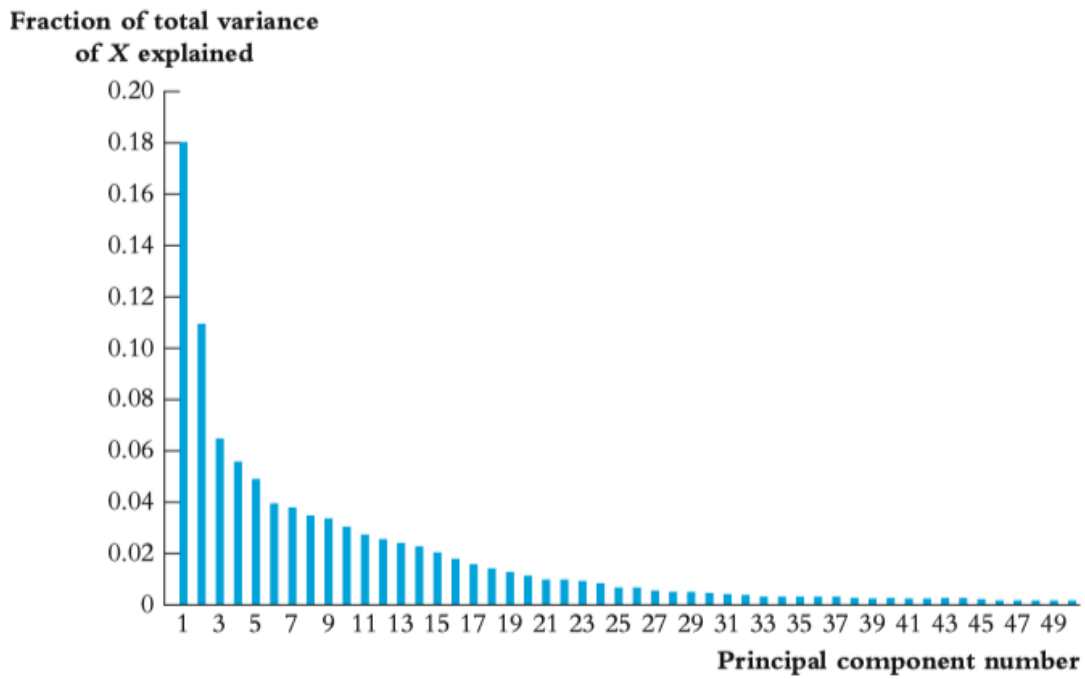
Principal Components

- Principal components can be thought of as a **data compression** tool, so that the compressed data have fewer regressors with as little information loss as possible.
- Data compression is used all the time to reduce very large data sets to smaller ones.
  - eg. image compression, where the goal is to retain as many of the features of the image (photograph) as possible, while reducing the file size.

Principal Components

- PC_1 explains 18% of the variation of all Xs.
- The first 10 PCs thus PC_1...PC_10 explains 63%.
- The first 40 PCs explains 92%.

FIGURE 14.6 Scree Plot for the 817-Variable School Data Set (First 50 Principal Components)



Plotted values are the fraction of the total variance of the 817 regressors explained by the indicated principal component. The first principal component explains 18% of the total variance of the 817 *X*'s, and the first 10 principal components together explain 63% of the total variance.

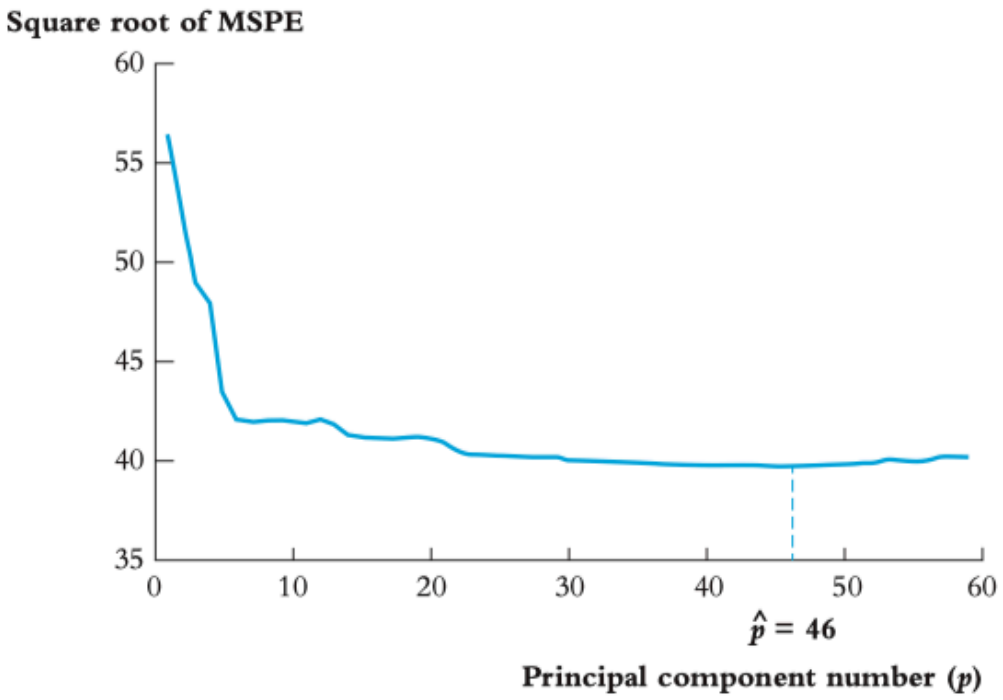
Principal Components

- It turns out that $p = 46$ by CV method.
-

$$RMSE_{PC} = 39.7$$

FIGURE 14.7 Square Root of the MSPE for the Principal Components Prediction as a Function of the Number of Principal Components p Used as Predictors

The MSPE is estimated using 10-fold cross validation for the school test score data set with $k = 817$ predictors and $n = 1966$ observations.



Predicting School Test Scores

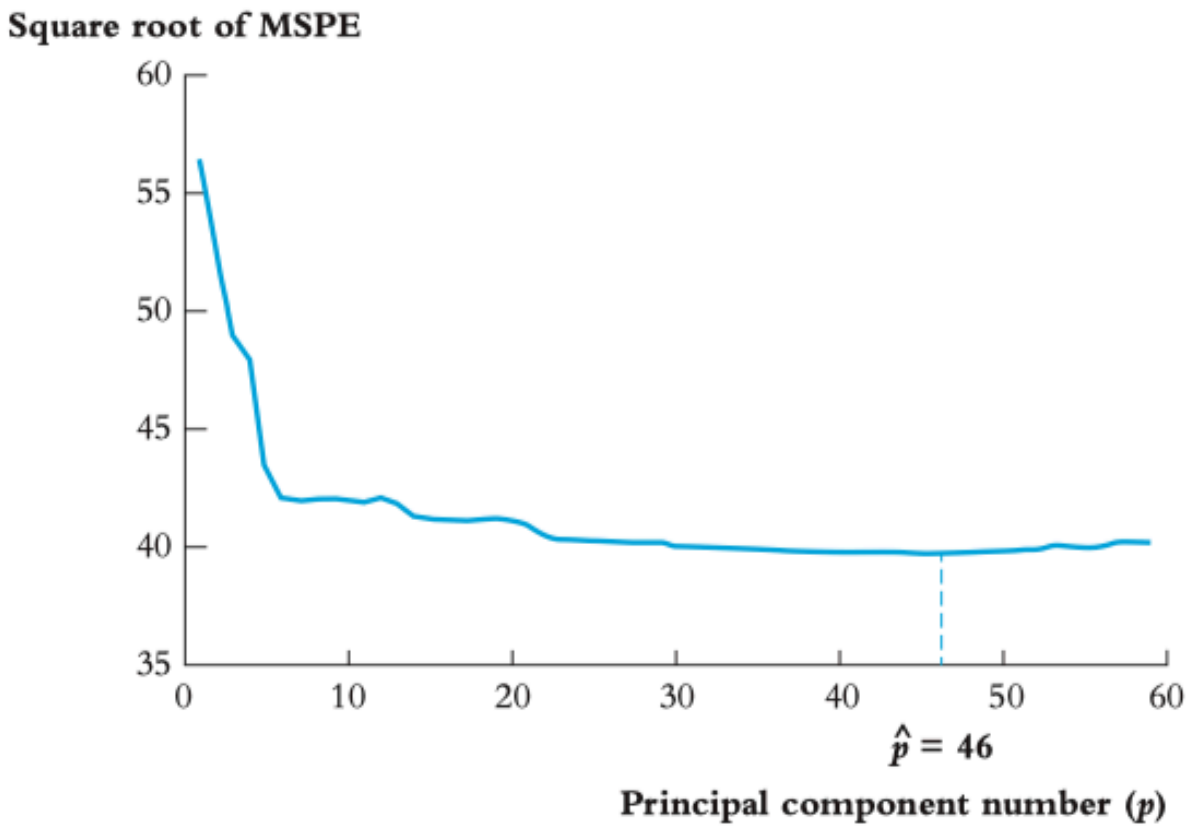
Basic procedure

- Split observations into two parts.
  1. The first half for model training.
  2. The second half for testing.
- There sets of predictors are used
  1. Small(k=4): Student-teacher ratio, median local income, teacher's average years of experience, instructional expenditures per student.
  2. Large(k=817)
  3. Vary Large(k=2065):Additional school and demographic variables, squares and cubes, and interactions. Note $k > n$.

The Three Sets of Predictors

FIGURE 14.7 Square Root of the MSPE for the Principal Components Prediction as a Function of the Number of Principal Components p Used as Predictors

The MSPE is estimated using 10-fold cross validation for the school test score data set with $k = 817$ predictors and $n = 1966$ observations.



Compare predictive models by predictions

TABLE 14.3 Out-of-Sample Performance of Predictive Models for School Test Scores

Predictor Set	OLS	Ridge Regression	Lasso	Principal Components
Small ($k = 4$)				
Estimated λ or p	—	—	—	—
In-sample root MSPE	53.6	—	—	—
Out-of-sample root MSPE	52.9	—	—	—
Large ($k = 817$)				
Estimated λ or p	—	2233	4527	46
In-sample root MSPE	78.2	39.5	39.7	39.7
Out-of-sample root MSPE	64.4	38.9	39.1	39.5
Very large ($k = 2065$)				
Estimated λ or p	—	3362	4221	69
In-sample root MSPE	—	39.2	39.2	39.6
Out-of-sample root MSPE	—	39.0	39.1	39.6

Notes: The in-sample MSPE is the 10-fold cross-validation estimate computed using the 1966 observations in the estimation

Compare predictive models by predictions

TABLE 14.4 Coefficients on Selected Standardized Regressors, 4- and 817-Variable Data Sets

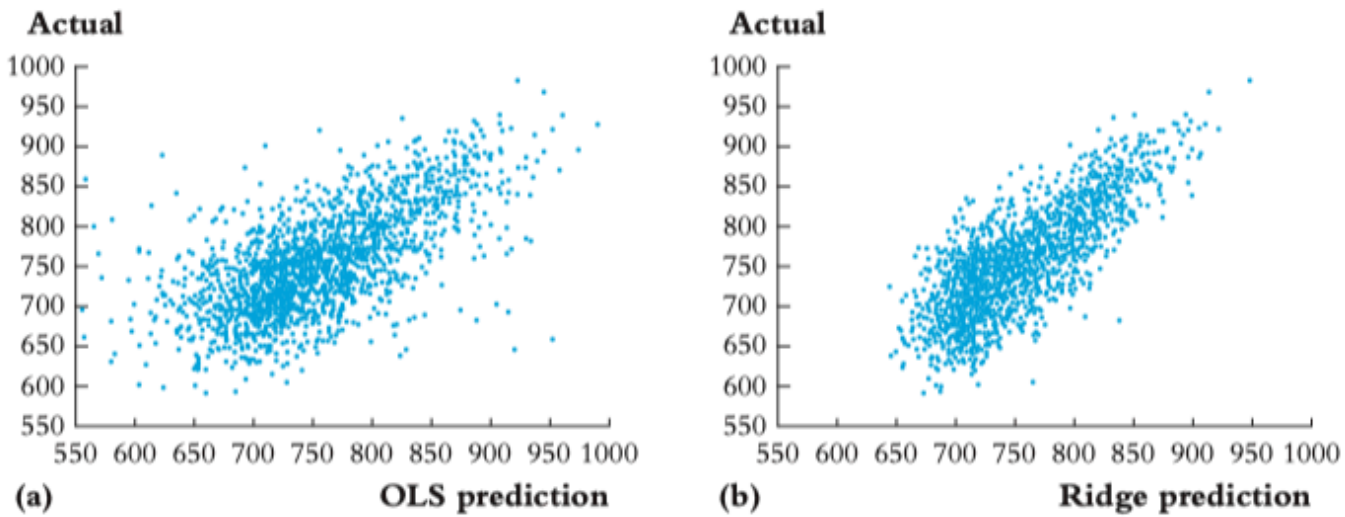
Predictor	$k = 4$	$k = 817$			
	OLS	OLS	Ridge Regression	Lasso	Principal Components
Student–teacher ratio	4.51	118.03	0.31	0	0.25
Median income of the local population	34.46	−21.73	0.38	0	0.30
Teachers' average years of experience	1.00	−79.59	−0.11	0	−0.17
Instructional expenditures per student	0.54	−1020.77	0.11	0	0.19
Student–teacher ratio × Instruction expenditures per student		−89.79	0.72	2.31	0.84
Student–teacher ratio × Fraction of English learners		−81.66	−0.87	−5.09	−0.55
Free or reduced-price lunch × Index of part-time teachers		29.42	−0.92	−8.17	−0.95

Notes: The index of part-time teachers measures the fraction of teachers who work part-time. For OLS, ridge, and Lasso, the coefficients in Table 14.4 are produced directly by the estimation algorithms. For principal components, the coefficients in Table 14.4 are computed from the principal component regression coefficients (the γ's in Equation ((14.13)), combined with the principal component weights. The formula for the β coefficients for principal components is presented using matrix algebra in Appendix 19.7.

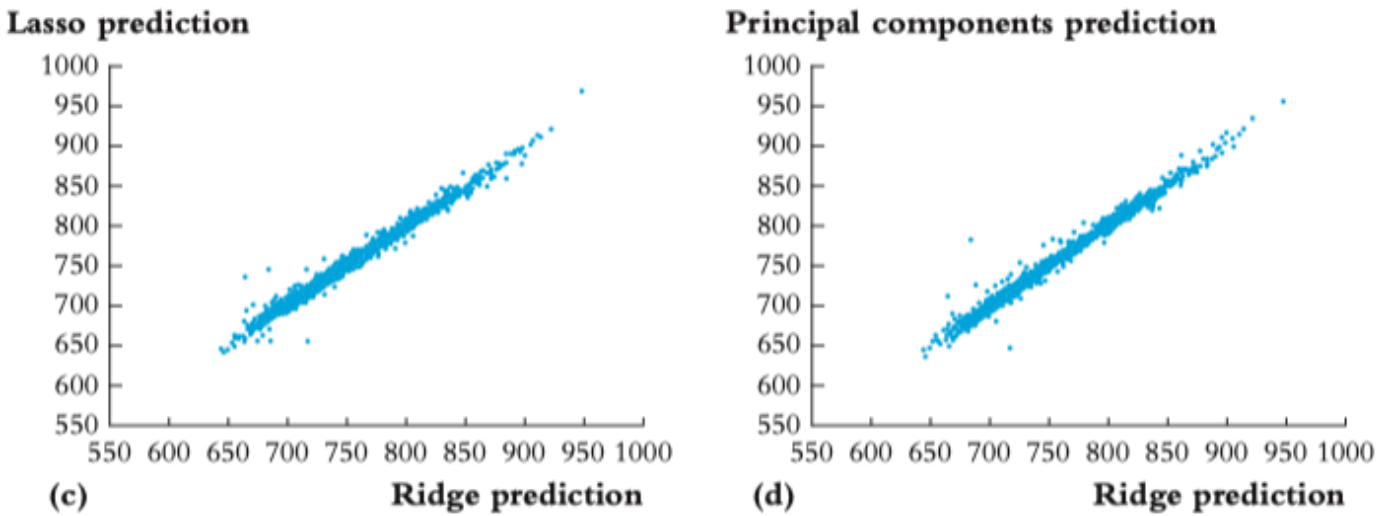
Compare predictive models by predictions

- THe tighter the spread of the scatter along the 45 degree line, the better the prediction.

FIGURE 14.8 Scatterplots for Out-of-Sample Predictions Using the 817-Predictor Data Set



Compare predictive models by predictions



(a) Actual versus OLS, (b) actual versus ridge, (c) Lasso versus ridge, and (d) principal components versus ridge.

Compare predictive models by predictions

- The most important conclusion from this application is that for the large data set the many-predictor methods succeed where OLS fails.
- Because the many-predictor methods allow the coefficients to be biased in a way that reduces their variance by enough to compensate for the increased bias.
- One finding that may not generalize: three methods happen to perform equally well in these data.

Summary

- With many predictors, OLS will produce poor out-of-sample predictions.
- By introducing the right type of bias— shrinkage towards zero— the variance of the prediction can be reduced by enough to offset the bias and result in smaller RMSE.
- Ridge and Lasso reduce the RMSE by shrinking (biasing) the estimated coefficients to zero— and in the case of Lasso, by eliminating many of the regressors entirely.
- Principal components collapses X into fewer uncorrelated linear combinations that capture as much of the variation of the X's as possible. Predictions are then made using the OLS regression of Y on the principal components.