Lecture 3:
OLS & Prediction in/out of Sample

Big Data and Machine Learning for Applied Economics
Econ 4676

Ignacio Sarmiento-Barbieri

Universidad de los Andes

August 18, 2020

Agenda

1. Recap
2. Motivation
3. Linear Regression
   - Prediction vs Estimation
   - Prediction vs Predictive Error
4. Train and Test Samples
5. Example: Predicting House Prices in R
6. Review
7. Further Readings

Recap

- We started shifting paradigms
- Decision Theory: Risk with square error loss $\rightarrow$ MSE
- Objective minimize the reducible error
- Irreducible error our unknown bound
- Machine Learning best kept secret: some bias can help lower MSE

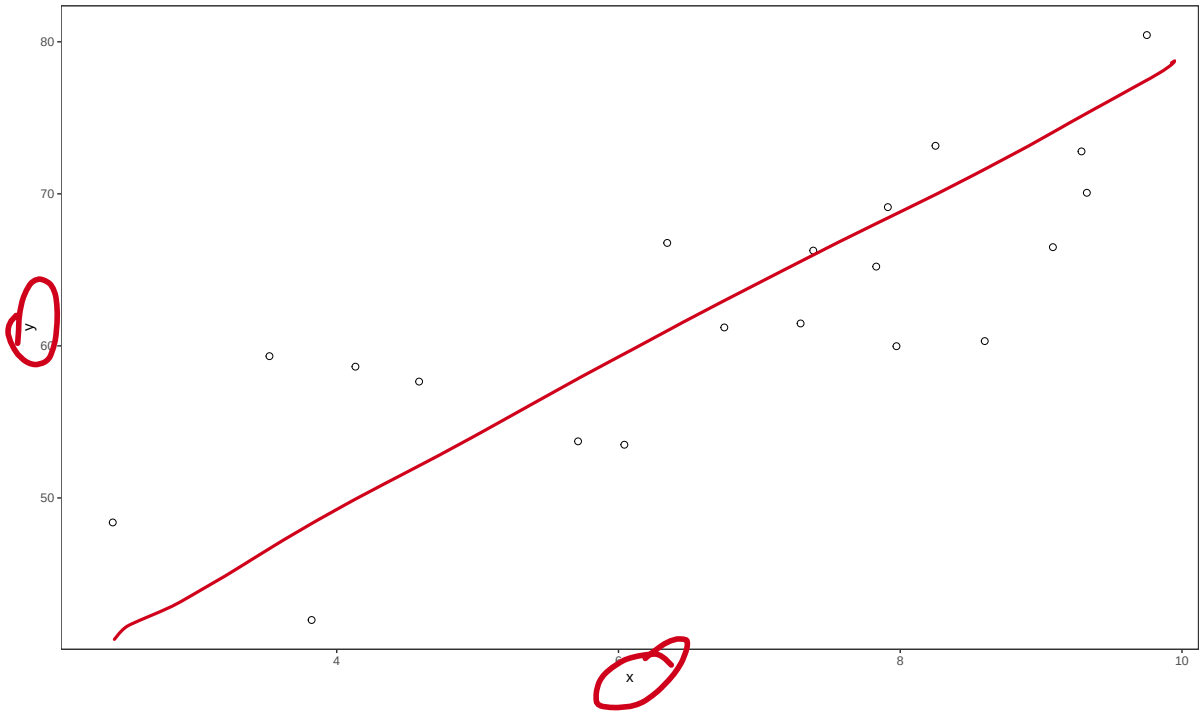
Motivation

- Linear regression is the "work horse" of econometrics and (supervised) machine learning.
- Very powerful in many contexts:
  - Model Spatial Relationships
  - Non linear relationships
- It can be used also for classification.
- Big payday' to study this model in detail.

Linear Regression Model

$$y = f(x) + u$$
$$f(x) = \beta x$$

Problem



Linear Regression Model

$f(X) = X\beta$ and the interest is on estimating β

$$y = X\beta + u \tag{1}$$

where

- y is a vector $n \times 1$ with typical element y_i
- X is a matrix $n \times k$
  - Note that we can represent it as a column vector $\underset{n\times k}{X} = [\underset{n\times 1}{X_1}\ \underset{n\times 1}{X_2} \dots \underset{n\times 1}{X_k}]$
- β is a vector $k \times 1$ with typical element β_j

Thus

$$y_i = X_i'\beta + u_i \tag{2}$$

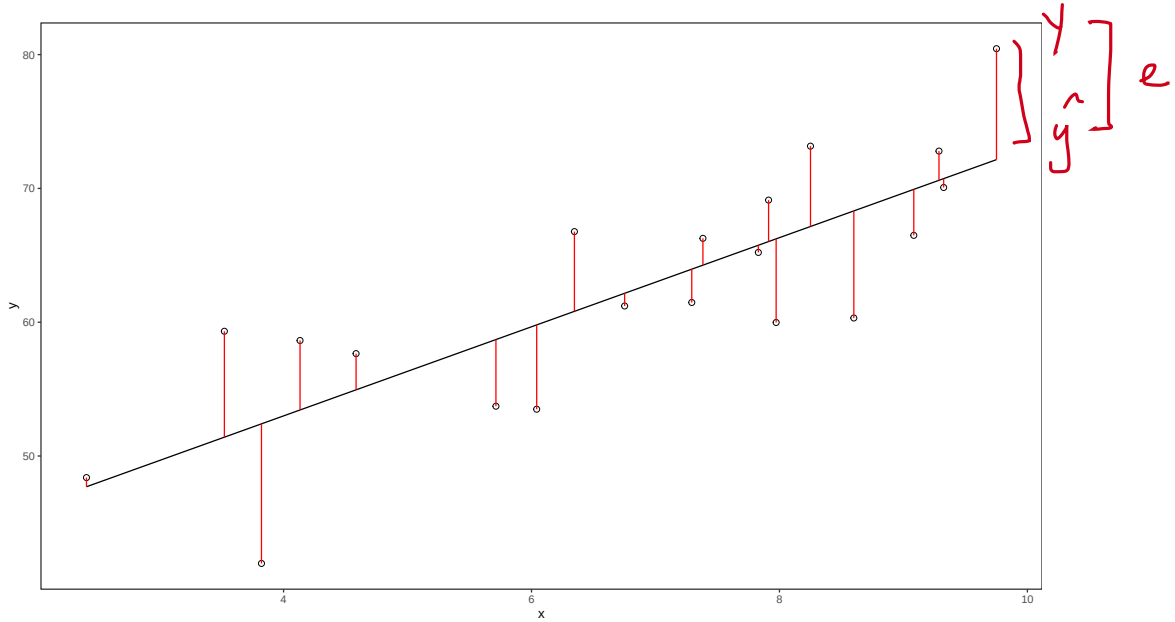
$$= \sum_{k=1}^{k} \beta_j X_{ji} + u_i$$

Linear Regression Model

How do we estimate β?

- Method of Moments (for HW)
- MLE (more on this later)
- OLS: minimize risk squared error loss $\rightarrow$ minimizes RSS ($e'e$)
  - where $e = Y - \hat{Y} = Y - X\hat{\beta}$
  - In the HW, you will show that min RSS same as max R^2

OLS solution: $\hat{\beta} = (X'X)^{-1}X'y$



Gauss Markov Theorem

Gauss-Markov Theorem says that

$$\hat{\beta} = (X'X)^{-1}X'y \tag{3}$$

- The OLS estimator ($\hat{\beta}$) is BLUE, the more efficient than any other linear unbiased estimator,
- Efficiency in the sense that $Var(\tilde{\beta}) - Var(\hat{\beta})$ is positive semidefinite matrix.

Proof: HW. Tip: a matrix $M_{p\times p}$ is positive semi-definite iff $c'Mc \geq 0 \; \forall x \in \mathbb{R}^p$

Gauss Markov Theorem

- Gauss Markov Theorem that says OLS is BLUE is perhaps one of the most famous results in statistics.
  - $E(\hat{\beta}) = \beta$
  - $V(\hat{\beta}) = \sigma^2(X'X)^{-1}$

$V(\tilde{\beta}) - \sigma^2(X'X)^{-1} \geq 0$

- However, it is essential to note the limitations of the theorem.
  - Correctly specified with exogenous Xs,
  - The term error is homoscedastic
  - No serial correlation.
  - Nothing about the OLS estimator being the more efficient than any other estimator one can imagine.

solve → white, robust

Prediction vs Estimation

- **Predicting well means estimating well**
  - Note that the prediction of y will be given by $\hat{y} = X\hat{\beta}$
  - Under Gauss-Markov framework
    - $E(\hat{y}) = X\beta$
    - $V(\hat{y}) = \sigma^2 X'(X'X)^{-1}X$
- Then if $\hat{\beta}$ is unbiased and of minimum variance,
- then $\hat{y}$ is an unbiased predictor and minimum variance, from the class of unbiased linear estimators/predictors
  - Proof: for HW similar to $\hat{\beta}$ proof

Prediction vs Estimation

- **Estimation Accuracy**

$$MSE(\beta) = E(\hat{\beta} - \beta)^2 \tag{4}$$

$$= E(\beta - E(\hat{\beta}))^2 + Var(\hat{\beta}) \tag{5}$$

- Intuitively, the result says that how wrong is the estimate (MSE) depends on:
  - how uncentered it is (bias) and
  - how dispersed it is around its center (variance).

Prediction and Predictive Error

- Now suppose that the goal is to predict Y with another random variable $\hat{Y}$.
- The *prediction error* is defined as:

$$Err(\hat{Y}) \equiv E\left(Y - \hat{Y}\right)^2 \tag{6}$$

- Conceptually the prediction error is equal to the MSE
- However, MSE compares a RV ($\hat{\beta}$) with a parameter (β)
- $Err(\hat{Y})$ involves two RV

Prediction and predictive error

- More concretely, the goal is to predict Y given another variable X.
- We assume that the link between Y and X is given by the simple model:

$$Y = f(X) + u \tag{7}$$

- where $f(X)$ is any function,
- u is an unobserved random variable with $E(u) = 0$ and $V(u) = \sigma^2$

Prediction and predictive error

- In practice we don't observe $f(X)$
- We need to estimate it with $\hat{f}(X)$ a RV

Then

$$Err(\hat{Y}) = MSE(\hat{f}) + \sigma^2 \tag{8}$$

$$= Bias^2(\hat{f}) + V(\hat{f}) + \sigma^2 \tag{9}$$

Two parts:

- the error from estimating f with $\hat{f}$. (*reducible*)
- the error from not being able to observe u. (*irreducible*)

This is an important result, predicting Y properly we need a good estimate of f.

Prediction and Linear regression

- Linear regression sets

$$f(X) = \beta_1 + \beta_2 X_2 + \cdots + \beta_k X_k \tag{10}$$

- In classical econometrics this model is given
- Focus is on the estimation of the unknown parameters $\beta_1, \ldots, \beta_k$.
- The prediction for Y is given by:

$$\hat{Y} = \hat{\beta}_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_k X_k \tag{11}$$

- where $\hat{\beta}_1, \ldots, \hat{\beta}_k$ are estimates.

Prediction and linear regression

- Under the classical assumptions the OLS estimator is unbiased, hence

$$\hat{f} = \hat{\beta}_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_k X_k$$

$$E(\hat{f}) = E(\hat{\beta}_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_k X_k) \tag{12}$$

$$= E(\hat{\beta}_1) + E(\hat{\beta}_2) X_2 + \cdots + E(\hat{\beta}_k) X_k \tag{13}$$

$$= f \tag{14}$$

Then,

$$E(\hat{\beta}_1) = \beta_1$$

- $MSE(\hat{f})$ reduces to just $V(\hat{f})$

Complexity and the variance/bias trade off

- When the focus switches from estimating f to predicting Y,
- f plays a secondary role, as just a tool to improve the prediction based on X.
- Predicting Y involves *learning* f, that is, f is no longer taken as given, as in the classical view.
- Now it implies an iterative process where initial choices for f are revised in light of potential improvements in predictive performance.
- Model choice or learning involves choosing both f and a strategy to estimate it ($\hat{f}$), guided by predictive performance.

Complexity and the variance/bias trade off

- Classical econometrics, model choice involves deciding between a smaller and a larger linear model.
- Consider the following competing models for y:

$$Y = \beta_1 X_1 + u_1 \tag{15}$$

and

$$Y = \beta_1 X_1 + \beta_2 X_2 + u_2 \tag{16}$$

- $\hat{\beta}_1^{(1)}$ the OLS estimator of regressing y on X_1
- $\hat{\beta}_1^{(2)}$ and $\hat{\beta}_2^{(2)}$ the OLS estimators of β_1 and β_2 of regressing Y on X_1 and X_2.

Complexity and the variance/bias trade off

The corresponding predictions will be

$$\hat{Y}^{(1)} = \hat{\beta}_1^{(1)} X_1 \tag{17}$$

and

$$\hat{Y}^{(2)} = \hat{\beta}_1^{(2)} X_1 + \hat{\beta}_2^{(2)} X_2 \tag{18}$$

Complexity and the variance/bias trade off

- An important discussion in classical econometrics is that of omission of relevant variables vs. inclusion of irrelevant ones.
  - If model (1) is true then estimating the larger model (2) leads to inefficient though unbiased estimators due to unnecessarily including X_2.
  - If model (2) holds, estimating the smaller model (1) leads to a more efficient but biased estimate if X_1 is also correlated with the omitted regressor X_2.

- this discussion of small vs large is always with respect to a model that is supposed to be true.

- But in practice the true model is unknown.

Complexity and the variance/bias trade off

- Choosing between models involves a *bias/variance trade off*
- Classical econometrics tends to solve this dilemma abruptly,
  - requiring unbiased estimation, and hence favoring larger models to avoid bias
- In this simple setup, larger models are 'more complex', hence more complex models are less biased but more inefficient.
- Hence, in this very simple framework complexity is measured by the number of explanatory variables.
- A central idea in machine learning is to generalize the idea of complexity,
  - Optimal level of complexity, that is, models whose bias and variance led to minimum MSE.

Train and Test Samples

- A goal of machine learning is *out of sample* prediction
- OLS estimator minimizes the sum of squared residuals and hence maximizes R^2 through maximizing the explained sum of squares.
- OLS is designed to optimize the predictive power of the model, for the data used for estimation.
- But in most predictive situations what really matters is the ability to predict new data.

Train and test samples

- A simple alternative would be to split the data into two groups
  - Training sample: to build/estimate/train the model
  - Test sample: to evaluate its performance

- From a strictly classical perspective
  - Makes sense if training data is iid from the population, even works if it is iid conditional on X
  - Two problems with this idea:
    - The first one is that given an original data set, if part of it is left aside to test the model, less data is left for estimation.
    - A second problem is how to decide which data will be used to train the model and which one to test it. (more on how cross validation helps later)

→ cutting the sample → more imprecise estimates

Train and test samples

The *estimated prediction error* is defined as

$$\hat{Err}(\hat{Y}) = \sum_{i \in Test\ Sample} (Y_i - \hat{Y}_i)^2 \tag{19}$$

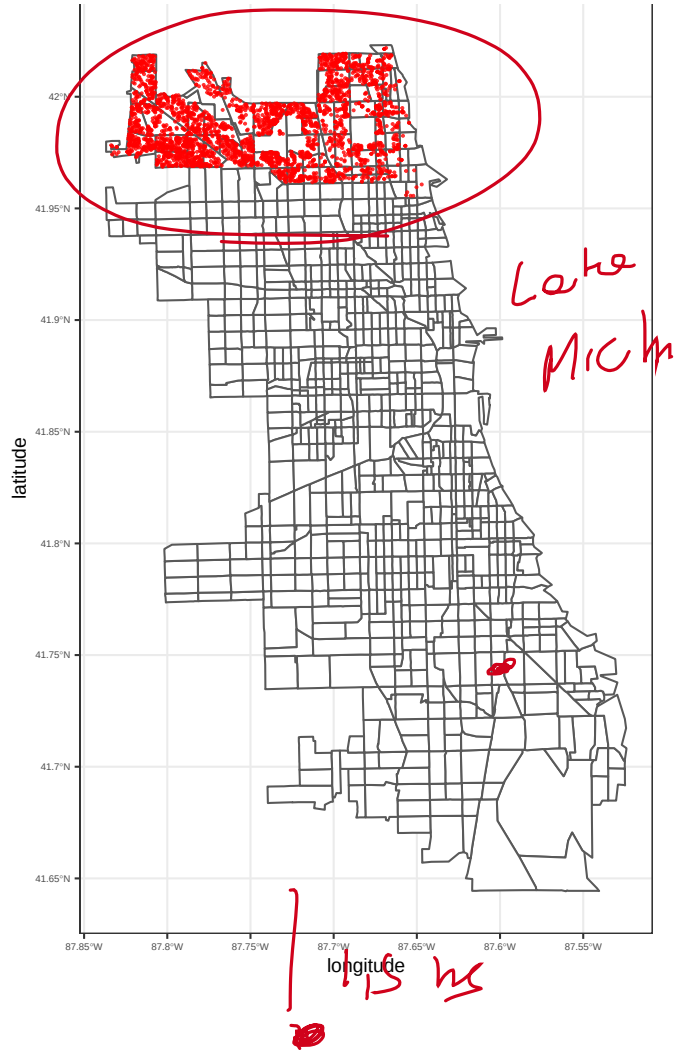
- $i \in Test\ Sample$ refers to all the observations in the test sample.
- $Test\ Sample \cup Training\ Sample = Full\ Sample$

- Note that:
  - No obvious way on how to partition this
  - In some cases is exogenously given. `Kaggle Competition`, `Netflix Challenge`
  - This idea is almost inexistent (or trivial) in classical econometrics

Example: Predicting House Prices in R

- `matchdata` in the *McSpatial package* for R.
- 3,204 sales of SFH Far North Side of Chicago in 1995 and 2005.
- This data set includes 18 variables/features about the home,
  - price sold
  - number of bathrooms, bedrooms,
  - latitude and longitude,
  - etc.
- in R:

```
require(mcmspatial)#loads the package
data(matchdata) #loads the data
?matchdata # help/info about the data
```



Example: Predicting House Prices in R

- Train and Test samples
- 70% / 30% split

```
set.seed(101010) #sets a seed
matchdata <- matchdata %>%
                        mutate(price=exp(lnprice),
                               #transforms log prices to standard prices
                               holdout= as.logical(1:nrow(matchdata)
                               %in% sample(
                               nrow(matchdata), nrow(matchdata)*.7))
                               #generates a logical indicator
                               )

test<-matchdata[matchdata$holdout==T,]
train<-matchdata[matchdata$holdout==F,]
```

Example: Predicting House Prices in R

- Naive approach: model with no covariates, just a constant
- $y = \beta_0 + u$

```
model1<-lm(price~1,data=train)
summary(model1)
```

```
##
## Call:
## lm(formula = price ~ 1, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -258018 -127093  -24018   92732  598482
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   284018       4782   59.39   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 148300 on 961 degrees of freedom
```

Example: Predicting House Prices in R

In this case our prediction for the log price is the average train sample average

$$\hat{y} = \hat{\beta}_0 = \frac{\sum y_i}{n} = m$$

```
coef(model1)
```

```
## (Intercept)
##    284017.6
```

```
mean(train$price)
```

```
## [1] 284017.6
```

Example: Predicting House Prices in R

$$Err(\hat{y}) = E(y - \hat{y})^2$$

- But we are concerned on predicting well our of sample,:

```
test$model1<-predict(model1,newdata = test)
with(test,mean((price-model1)^2))
```

$$(y - \hat{y})^2$$

```
## [1] 21935777917
```

- $\hat{A}Err(\hat{y}) = \frac{\sum((y-\hat{y})^2)}{n} = 2.1935778 \times 10^{10}$
- This is our starting point, Can we improve it?

Example: Predicting House Prices in R

- How to improve it?
  - One way is using econ theory as guide
  - hedonic house price function derived directly from the Rosen's theory of hedonic pricing
  - however, the theory says little on what are the relevant attributes of the house.
  - So we are going to explore the effects of adding house characteristics on our out $AErr(\hat{y})$
- The simple inclusion of a single covariate can improve with respect to the *naive* constant only model.

```
model2<-lm(price~bedrooms,data=train)
test$model2<-predict(model2,newdata = test)
with(test,mean((price-model2)^2))
```

```
## [1] 21695551442
```

menor ✓

Example: Predicting House Prices in R

- What about if we include more variables?

```
model3<-lm(price~bedrooms+bathrooms+centair+fireplace+brick,data=train)
test$model3<-predict(model3,newdata = test)
with(test,mean((price-model3)^2))
```

```
## [1] 21111169595
```

- Note that the *AErr* is once more reduced. If we include all?

```
model4<-lm(price~bedrooms+bathrooms+centair+fireplace+brick+
                 lnland+lnbldg+rooms+garage1+garage2+dcbd+rr+
                 yrbuilt+factor(carea)+latitude+longitude,data=train)
test$model4<-predict(model4,newdata = test)
with(test,mean((price-model4)^2))
```

```
## [1] 20191829518
```

- Is there a limit to this improvement? Can we keep adding features and complexity?

Example: Predicting House Prices in R

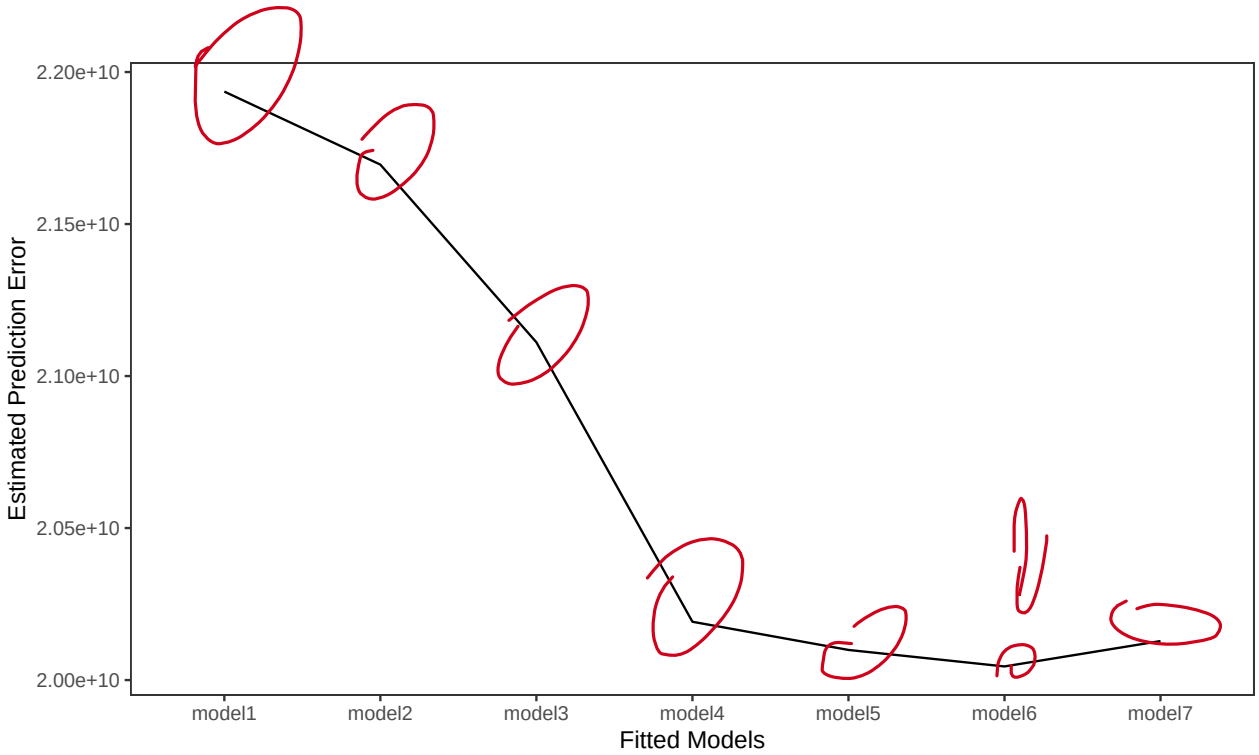
- Is there a limit to this improvement? Can we keep adding features and complexity?
- Let's try a bunch of models

```
model5<-lm(price~poly(bedrooms,2)+poly(bathrooms,2)+
                 centair+fireplace+brick+
                 lnland+lnbldg+rooms+
                 garage1+garage2+dcbd+rr+
                 yrbuilt+factor(carea)+poly(latitude,2)+
                 poly(longitude,2),data=train)
test$model5<-predict(model5,newdata = test)

model6<-lm(price~poly(bedrooms,2)+poly(bathrooms,2)+centair+fireplace+brick+
                 lnland+lnbldg+garage1+garage2+rr+
                 yrbuilt+factor(carea)+poly(latitude,2)+poly(longitude,2),
                 data=train)
test$model6<-predict(model6,newdata = test)

model7<-lm(price~poly(bedrooms,2)+poly(bathrooms,2)+centair+fireplace+brick+
                 lnland+lnbldg+garage1+garage2+rr+
                 yrbuilt+factor(carea)+poly(latitude,3)+poly(longitude,3),
                 data=train)
test$model7<-predict(model7,newdata = test)
```

Example: Predicting House Prices in R



- Take aways from the example
- Classical econometrics set up, choosing between smaller and larger models
- More complexity, the prediction error keeps getting smaller.
- The choice of a model's complexity faces a bias/variance trade-off.
- Open question: how to find the optimal complexity level?

(later on the course)

Review & Next Steps

- Linear Regression
- Prediction vs Estimation
- Train and Test Samples
- Example in `R`

- **Next Class:** Big Data intro, OLS Numerical Properties Computation.

- Questions? Questions about software?

Further Readings

- Davidson, R., & MacKinnon, J. G. (2004). Econometric theory and methods (Vol. 5). New York: Oxford University Press.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York: springer.
- Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1, No. 10). New York: Springer series in statistics.
- Murphy, K. P. (2012). Machine learning: a probabilistic perspective. MIT press.