Lecture 14: Prediction, Complexity, and the Bias-Variance Trade-off

Big Data and Machine Learning for Applied Economics
Econ 4676

Ignacio Sarmiento-Barbieri

Universidad de los Andes

September 23, 2021

Agenda

Recap

- Model

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

- We went over different estimation methods
  - OLS (max. in sample fit), numerical properties: big n, update β.
  - MM (match moments)
  - MLE (find parameters that maximizes the likelihood of the sample)
    - $y = \rho Wy + X\beta + u$
  - Bayesian (updating priors)
    - How to incorporate information
    - Explicit about priors.
- Now and for the rest of the semester we shift our attention to prediction (with some applications to inference)

Prediction and Linear Models

- Suppose we have a linear model

$$y = \beta_1 + \beta_2 X_2 + \cdots + \beta_k X_k + u \tag{2}$$

- With classical assumptions on u, $E(u) = 0$ and $V(u) = \sigma^2$
- In this context, estimating well means in predicting well
- The prediction for Y is given by:

$$\hat{y} = \hat{\beta}_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_k X_k \tag{3}$$

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

Prediction and Linear Models

- How we evaluate the performance of out predictor?
- The *prediction error* is defined as:

$$Err(\hat{y}) = E\left(L(y - \hat{y})\right) \tag{4}$$

- under a square loss $(\theta - a)^2$

$$Err(\hat{y}) = E\left(y - \hat{y}\right)^2 \tag{5}$$

- The prediction error is equal to the MSE

Prediction and Linear Models

- Model $y = X\beta + u$
- Prediction is given by $\hat{y} = X\hat{\beta}$
- The prediction error:

$$
\begin{aligned}
Err(\hat{y}) &= E\left(y - \hat{y}\right)^2 && (6)\\
&= E\left(y - E(\hat{y}) + E(\hat{y}) - \hat{y}\right)^2 && (7)\\
&= E\left(X\beta + u - E(X\hat{\beta}) + E(X\hat{\beta}) - X\hat{\beta}\right)^2 && (8)\\
&\vdots && (9)\\
&= X' Bias^2\left(\hat{\beta}\right) X + X' Var(\hat{\beta}) X + \sigma^2 && (10)
\end{aligned}
$$

- Three parts:
  - Bias of our estimator (*reducible*)
  - The variance of our estimator (*reducible*)
  - The error from not being able to observe u. (*irreducible*)

This is a sketch, do the missing steps as HW

Prediction and linear regression

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

$$
\begin{aligned}
E(X\hat{\beta}) &= E(\hat{\beta}_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_k X_k) && (11)\\
&= E(\hat{\beta}_1) + E(\hat{\beta}_2) X_2 + \cdots + E(\hat{\beta}_k) X_k && (12)\\
&= X\beta && (13)
\end{aligned}
$$

Then,

- $Err(\hat{y})$ reduces to $V(\hat{\beta})$

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:

$$Y = \beta_1 X_1 + u_1 \tag{14}$$

and

$$Y = \beta_1 X_1 + \beta_2 X_2 + u_2 \tag{15}$$

- $\hat{\beta}_1^{(1)}$ 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

$$\hat{Y}^{(1)} = \hat{\beta}_1^{(1)} X_1 \tag{16}$$

and

$$\hat{Y}^{(2)} = \hat{\beta}_1^{(2)} X_1 + \hat{\beta}_2^{(2)} X_2 \tag{17}$$

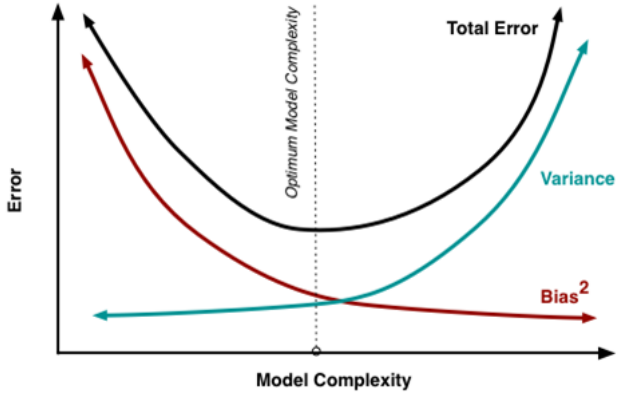
Complexity and the variance/bias trade off

- 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.

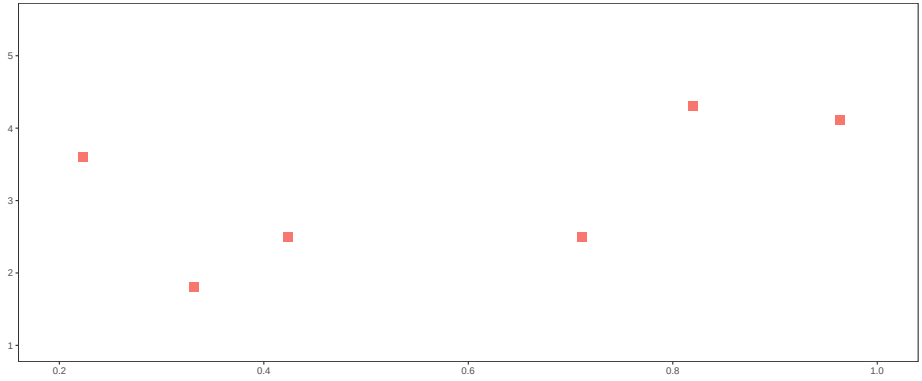
Complexity and the variance/bias trade off



Source: https://tinyurl.com/y4lvjxpc

A very interesting discussion in a recent Twitter thread by Daniela Witten: https://twitter.com/daniela_witten/status/1292293102103748609?s=20

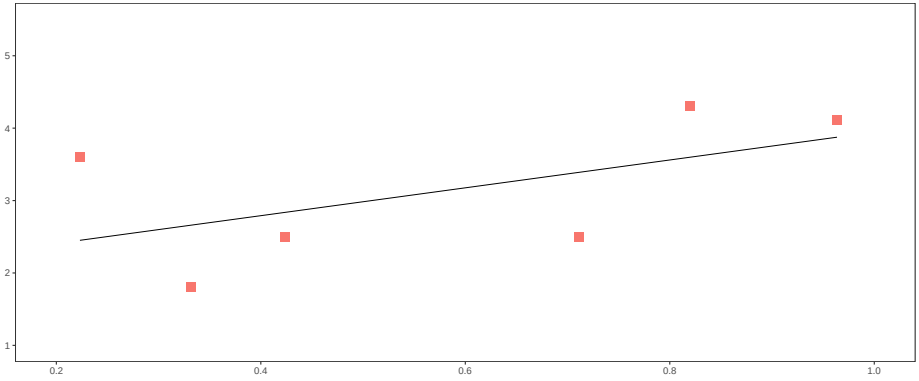
Complexity and the variance/bias trade off



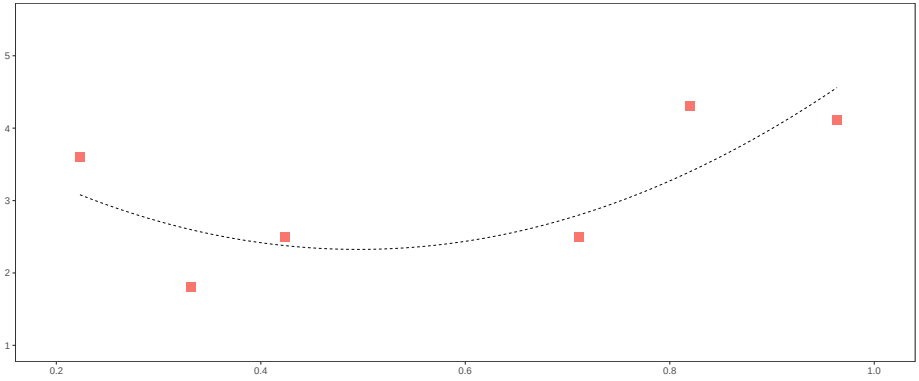
Complexity and the variance/bias trade off

- Suppose that the true model is $y = \sum x_0^{p^*} \beta_s + u$ with $E(u) = 0$ and $V(u) = \sigma^2$
- and p^* is finite but unknown
- We fit polynomials with increasing degrees $p = 1, 2, ...$
- What happens when we increase the degree of the polynomial?

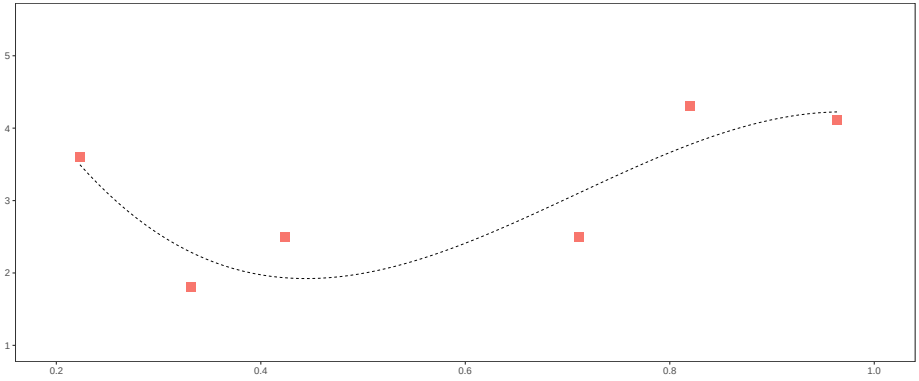
Complexity and the variance/bias trade off



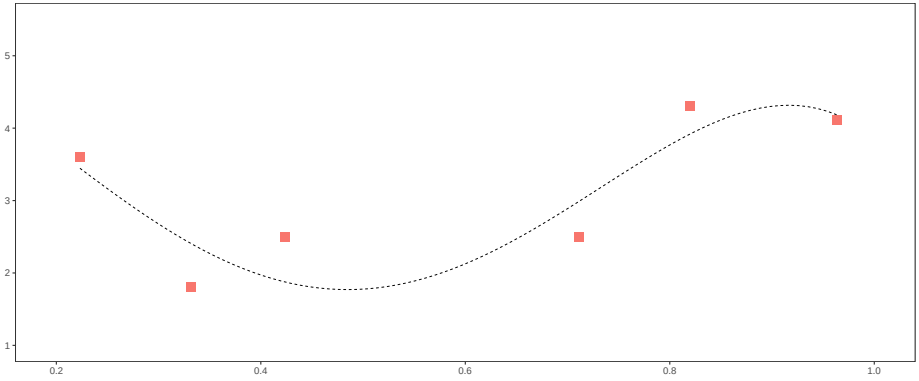
Complexity and the variance/bias trade off



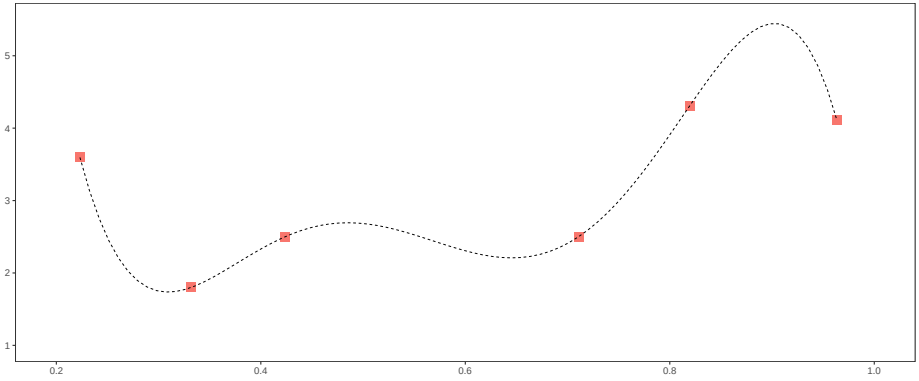
Complexity and the variance/bias trade off



Complexity and the variance/bias trade off



Complexity and the variance/bias trade off



Complexity and the variance/bias trade off

- Bias ?
- Variance:

$$\hat{f}(x_0) = \sum_{s=0}^{p} x_0^s \hat{\beta}_s = x_0' \hat{\beta} \tag{18}$$

- where $x_0' = (1, x_0, , x_0^2, \ldots, x_0^p)$

$$V(\hat{f}(x_0)) = V(x_0' \hat{\beta}) = x_0' \sigma^2 (X'X)^{-1} x_0 \tag{19}$$

- Then

$$\frac{1}{n} \sum_{i=1}^{n} \sigma^2 x_i' (X'X)^{-1} x_i = \sigma^2 \frac{p}{n} \tag{20}$$

- After we "hit" p^* increasing complexity does not reduce the bias, but variance increases monotonically for σ^2 and n given

Proof

- The fitted model for a polynomial of degree p for observation i is :

$$\hat{y}_i = x_i'\hat{\beta} \tag{21}$$

with $x_i' = (1, x_i, , x_i^2, \ldots, x_i^p)$

- Then $V(y_i) = V(x_i'\hat{\beta}) = \sigma^2 x_i'(X'X)^{-1}x_i$.

$$Average\, V(x_i'\hat{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \sigma^2 x_i'(X'X)^{-1}x_i \tag{22}$$

Proof

- Trace.
  - If $A_{m\times m}$ with typical element a_{ij}. The **trace** of A, $tr(A)$ is the sum of the elements of its diagonal: $tr(A) \equiv \sum_{s=1}^{m} a_{ii}$
  - Properties
    - For any square matrices A, B, and C: $tr(A+B) = tr(A) + tr(B)$
    - Cyclic property: $tr(ABC) = tr(BCA) = tr(CAB)$
    - If $m = 1$ tr(A)=A

- Let's use traces.
- Note that $x_i'(X'X)^{-1}x_i$ is a scalar, using the third property of traces

$$Average\, V(x_i'\hat{\beta}) = \frac{1}{n}\sum_{i=1}^{n} \sigma^2 tr(x_i'(X'X)^{-1}x_i) \tag{23}$$

Proof

- Using the cyclic property,

$$tr(x_i'(X'X)^{-1}x_i) = tr((X'X)^{-1}x_i'x_i) \tag{24}$$

- and the first property of traces,

$$\sum_{i=1}^{n} tr((X'X)^{-1}x_i'x_i) = tr(\sum_{i=1}^{n}(X'X)^{-1}x_i'x_i) \tag{25}$$

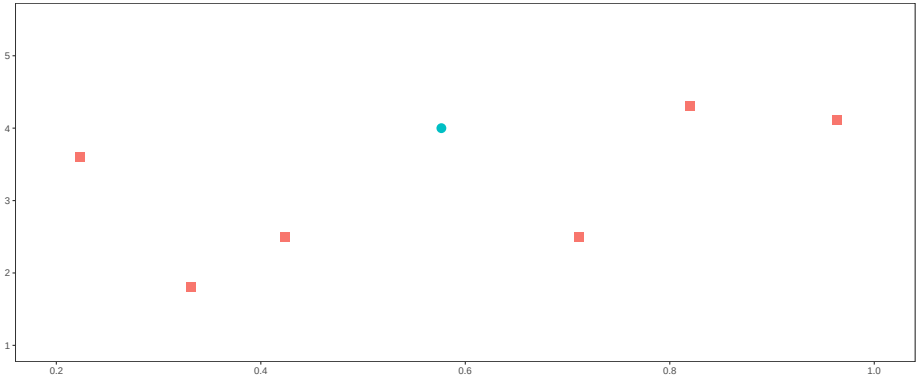
$$= tr((X'X)^{-1}(X'X)) \tag{26}$$

$$= p \tag{27}$$

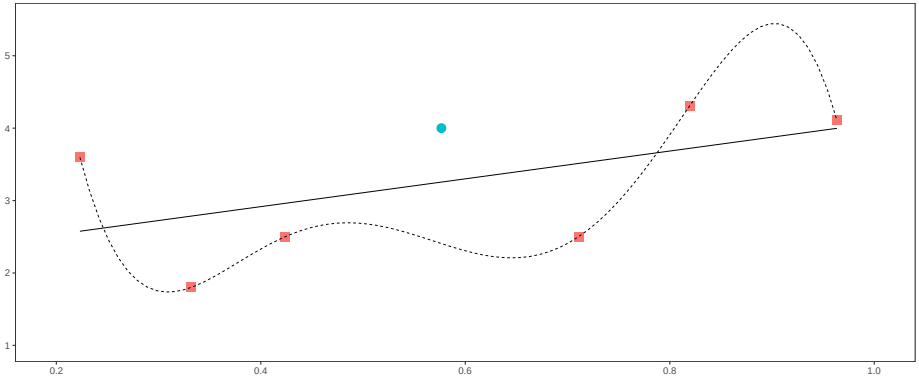
Overfit, Train and Test Samples

- 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.

Overfit, Train and Test Samples



Overfit, Train and Test Samples



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 (leading to less efficiency).
    - 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)

Train and test samples

The *estimated prediction error* is defined as

$$\hat{Err}(\hat{y}) = \sum_{i \in Test\ Sample} (y_i - \hat{y}_i)^2 \tag{28}$$

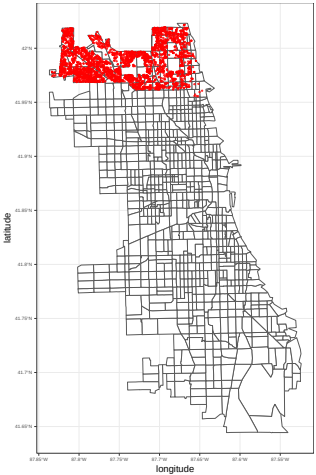
- $i \in$ *Test Sample* refers to all the observations in the test sample.
- *Test Sample* $\cup$ *Training Sample* $=$ *Full Sample*

- 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(mcspatial)#loads the package
data(matchdata) #loads the data
?matchdata # help/info about the data
```



Example: Predicting House Prices in R

- Train and Test samples
- 30% / 70% 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                    to div
                            )

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

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

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

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

- $\hat{Err}(\hat{y}) = \frac{\sum((y-\hat{y})^2)}{n} = 2.1935778 \times 10^{10}$
- 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 $Err(\hat{y})$
- 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
```

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 *Err* 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

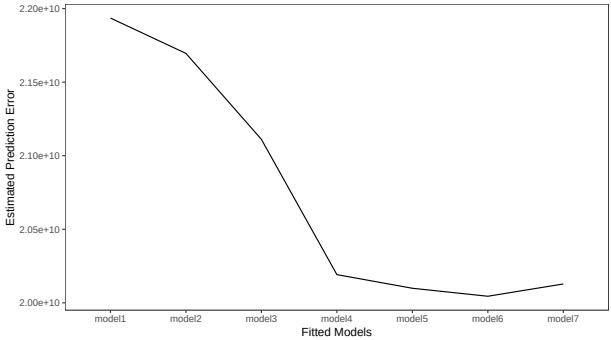
- Is there a limit to this improvement? Can we keep adding features and complexity?
- Let's try a bunch of models

```
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



Example: Predicting House Prices in R

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

(more on this in the coming weeks)

Review & Next Steps

- Prediction and Linear Models
- Complexity, Bias-Variance Trade off, Overfit.
- Train and Test Samples
- Example in R
- **Next Week:** Resampling Methods and Model Selection

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.