MS&E 226: Fundamentals of Data Science

Lecture 2: Introduction to prediction

Ramesh Johari
rjohari@stanford.edu

Prediction

In this part of the class we focus only on the prediction problem:

Given a sample $\mathbf{X}$ *and* $\mathbf{Y}$*, construct a fitted model that has low* error *in predicting outcomes on the entire population.*

This is commonly referred to as the *supervised learning* problem: our sample consists of feature vectors *and* associated outcomes, that we can use to "supervise" the process of finding a good fitted model for prediction.

The prediction problem: Formalism

- $\mathbf{X}, \mathbf{Y}$: Data we are given
- $\vec{X}$: Covariate vector of a *new* data point from the population
- Y: True outcome associated with $\vec{X}$
- $\hat{f}(\cdot)$: Fitted model (input: covariate vector; output: predicted outcome)

Goal: Using $\mathbf{X}$ and $\mathbf{Y}$, construct $\hat{f}$ so that $Y \approx \hat{f}(\vec{X})$.

The prediction problem: Example

- $\mathbf{X}, \mathbf{Y}$: 1,728 houses in Saratoga County
- $\vec{X}$: Features of a house drawn from the population, e.g., houses in upstate New York
- Y: True sales price of the house with covariates $\vec{X}$
- $\hat{f}(\cdot)$: E.g., linear regression model fit using $\mathbf{X}$ and $\mathbf{Y}$
- $\hat{f}(\vec{X})$: Predicted price

Goal: Construct $\hat{f}$ so that true price $\approx$ predicted price.

Classification vs. regression

Two broad classes of prediction problems:

1. *Regression*: Y is a continuous variable (numeric). Examples:
   - Predict wealth given demographic factors
   - Predict customer spend given profile
   - Predict earthquake magnitude given seismic characteristics
   - Predict level of antigen given biological markers
2. *Classification*: Y is a categorical variable (factor). Examples:
   - Is this e-mail spam or not?
   - What zip code does this handwriting correspond to?
   - Is this customer going to buy an item or not?
   - Does this patient have the disease or not?

We focus on *regression* problems first.

Prediction error

Measurement of prediction error depends on the type of prediction problem.

For *regression*, two common examples of prediction error measures include:

- *Squared error* $(Y - \hat{f}(\vec{X}))^2$;
- *Absolute deviation* $|Y - \hat{f}(\vec{X})|$.

We focus on squared error for now. (Mainly because it is very widely used, as a matter of convenience.)

A “good” fitted model

We start with a basic question:

How do we find a “good” fitted model?

Idea 1: The sample mean

Since we want small squared error, we could just pick the real number c that makes squared error smallest:

$$\text{minimize} \quad \sum_{i=1}^{n} (Y_i - c)^2.$$

Theorem

The number $\hat{c}$ that solves the optimization problem above is the sample mean:

$$\hat{c} = \overline{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i.$$

Exercise: Prove this.

Idea 1: The sample mean

```
> c = mean(sh$price)
> c
[1] 211966.7
> mean( (sh$price - c)^2 )
[1] 9685099417
> sqrt(mean( (sh$price - c)^2 ))
[1] 98412.9
```

The second quantity is the *RMSE* (root mean squared error);
we often use it because it's in the same *units* as the original outcome (price).

Idea 2: Group means

The mean is pretty simplistic, since it doesn't even use any features!

Why not at least take advantage of *grouping* in the data, e.g., by heating type?

Group-based predictions (three heating types):

```
c_hotair (mean for hot air heating): 226355.4
c_hotwater (mean for hot water/steam heating): 209132.5
c_electric (mean for electric heating): 161888.6
```

Idea 2: Group means – MSE and RMSE

For each house in our dataset, we make the prediction based on the group mean (i.e., based on the average price of houses with the same type of heating):

For each i, define $\hat{Y}_i =$ `c_hotair` if `heating = hot air`; etc.

The MSE is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}(Y_i - \hat{Y}_i)^2,$$

and the RMSE is $\sqrt{\text{MSE}}$.

With this formula the RMSE becomes $95429.27.

Idea 2: Group means

Could also group by “binning” `livingArea`:

Round `livingArea` to nearest 200 square feet, then compute group means within in “bin.”

Now the RMSE (same calculation as previous slide) is $\$67685.18$.

Complexity...

Obviously these “simple” manual approaches get complicated quickly...

Here’s an idea:

- Sample mean minimizes sum of squared errors.
- Why not do the same thing, but using more features?

First approach: fit a *linear* function of the features to predict price.

Idea 3: Ordinary least squares (OLS) linear regression

Given coefficients $\hat{\beta}$, make the *fitted* value for the i'th observation:

$$\hat{Y}_i = \sum_{j=1}^{p} \hat{\beta}_j X_{ij}.$$

Or could also add an additional *intercept* term (constant $\hat{\beta}_0$):

$$\hat{Y}_i = \beta_0 + \sum_{j=1}^{p} \hat{\beta}_j X_{ij}.$$

Ordinary least squares (OLS): Choose $\hat{\beta}$ so that:

$$\text{SSE} = \text{sum of squared errors} = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2.$$

(or equivalently, MSE) is **minimized**. *Terminology note:* $r_i = Y_i - \hat{Y}_i$ is also called the i'th *residual*.

Idea 3: Ordinary least squares (OLS) linear regression

How to interpret the coefficients?

- $\hat{\beta}_0$ is the fitted value when all the covariates are zero.
- $\hat{\beta}_j$ is the change in the fitted value for a one unit change in the j'th covariate, *holding all other covariates constant*.

Existence and uniqueness of OLS solution

OLS always has at least one solution, but the solution may not be unique.

If $n > p$ and the columns of $\mathbf{X}$ are *linearly independent*, there will be a unique OLS coefficient vector $\hat{\boldsymbol{\beta}}$.

(When the columns of $\mathbf{X}$ are *not* linearly independent, we say they are *collinear*.)

See appendix for more details on the linear algebra of the OLS solution.

OLS: Example 1

To start, here's an example using *just* `livingArea` and an intercept:

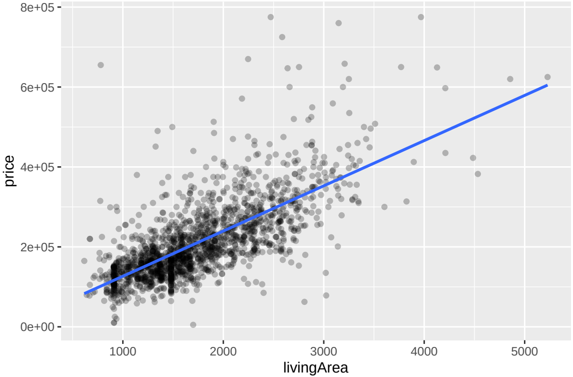
```
> fm = lm(data = sh, price ~ 1 +  livingArea)
> summary(fm)
...
Coefficients:
             Estimate ...
(Intercept) 13439.394 ...
livingArea    113.123 ...
...
```

In other words: `price` $\approx 13{,}439.394 + 113.123 \times$ `livingArea`.

Note: `summary(fm)` produces lots of other output too! We are going to gradually work in this course to understand what each of those pieces of output means.

OLS: Example 1 – Plot

Here is the model plotted against the data:

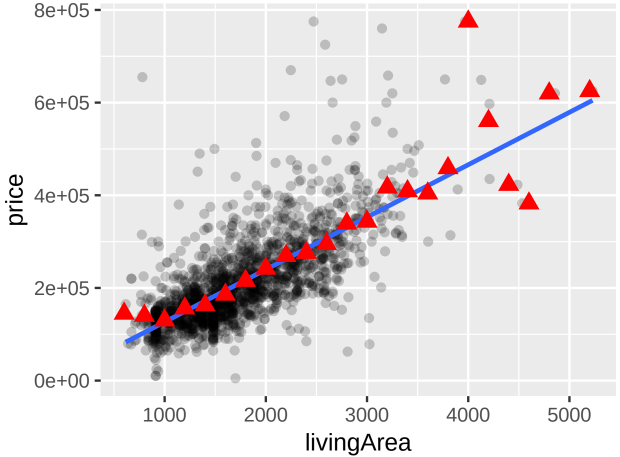


The RMSE is now $69064.56.

```
> ggplot(data = sh,
    aes(x = livingArea,
         y = price)) +
  geom_point() +
  geom_smooth(method="lm",
              se=FALSE)
```

Comparison to "binned" means

Now let's try plotting the model again, *together* with previously "binned" means of `livingArea`:



We are observing something that will pop up repeatedly:

OLS approximates the conditional average of the outcome, given the features.

OLS: Example 2 – Categorical variables

Let’s see this in action in another case.

```
> fm = lm(data = sh, price ~ 1 + heating)
> summary(fm)
...
Coefficients:
            Estimate ...
(Intercept)             226355  ...
heatinghot water/steam  -17223  ...
heatingelectric         -64467  ...
...
```

OLS: Example 2 – Categorical variables

Let's see this in action in another case.

```
> fm = lm(data = sh, price ~ 1 + heating)
> summary(fm)
...
Coefficients:
            Estimate ...
(Intercept)                 226355  ...
heatinghot water/steam      -17223  ...
heatingelectric             -64467  ...
...
```

What do these new variable names mean?

OLS: Example 2 – 1-hot encoding

Effectively, R creates two new binary variables (sometimes called “1-hot” encoding):

- The first is 1 if `heating` is `hot water/steam`, and zero otherwise.
- The second is 1 if `heating` is `electric`, and zero otherwise.

What if they are both zero? Why is there no variable `heatinghot air`?

OLS: Example 2 – Interpretation

What do the coefficients mean? Recall that:

```
c_hotair (mean for hot air heating): 226355.4
c_hotwater (mean for hot water/steam heating): 209132.5
c_electric (mean for electric heating): 161888.6
```

OLS: Example 2 – Interpretation

What do the coefficients mean? Recall that:

```
c_hotair (mean for hot air heating): 226355.4
c_hotwater (mean for hot water/steam heating): 209132.5
c_electric (mean for electric heating): 161888.6
```

The intercept (226355) corresponds to the baseline category: hot air heating.

The other coefficients are *differences* from the baseline:

- Hot water/steam: 226355 - 17223 = 209132
- Electric: 226355 - 64467 = 161888

OLS: Example 3

Now let's build a model using all six features.

```
> fm = lm(data = sh,
          price ~ 1 + livingArea + age + bedrooms +
                bathrooms + heating + new)
> summary(fm)
...
Coefficients:
                         Estimate ...
(Intercept)              4667.227 ...
livingArea                105.656 ...
age                       -39.883 ...
...
```

RMSE becomes $66820.67.

R^2

An alternative way to measure our fitted model is to ask:

How much lower is our sum of squared errors than the sample mean?

This leads to the definition of R^2 for fitted values $\hat{Y}_i$ from a predictive model:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{n}(Y_i - \overline{Y}_i)^2}.$$

R^2

If R^2 is close to 1, then our SSE is *much lower* than the sample mean.

If R^2 is *negative*, SSE is *higher* than the sample mean!
(For “reasonable” models, this will not happen.)

Each of our new models *increased* R^2. For our last model, $R^2 \approx 0.539$.

(In the R summary of a `lm` call, R^2 is reported as “multiple R-squared”.)

Idea 4: "Feature engineering"

Why stop there? We could use our existing features to create *new* features.

This is typically referred to as *feature engineering*. We look at a few examples:

- *Preprocessing steps*: Centering and standardization
- *Nonlinear transformations*: Higher order terms, interactions, logarithms
- *Complex models*: Embeddings,

Preprocessing: Centering and standardization

Two common steps in preprocessing are *centering* by removing the mean, and additionally *standardizing* by dividing by the sample standard deviation:

$$\tilde{X}_{ij} = \frac{X_{ij} - \overline{X}_j}{\hat{\sigma}_j}.$$

These steps have the effect of ensuring all covariates have *zero* sample mean, and sample standard deviation *one* (i.e., normalized dispersion).

On the problem set you will investigate some of the consequences of centering and standardizing.

Standardizing will not change the R^2 of your model, even though it changes coefficient values – check this!

Higher order terms

Even though OLS yields a “linear” model”, we can add features that are *nonlinear* transformations of the data.

Higher order terms

Even though OLS yields a “linear” model”, we can add features that are *nonlinear* transformations of the data.

Example: Suppose we add second power of `livingArea`.
R formula:

```
price ~ 1 + livingArea + age + bedrooms +
   bathrooms + heating + new + I(livingArea^2)
```

Higher order terms

Even though OLS yields a “linear” model”, we can add features that are *nonlinear* transformations of the data.

Example: Suppose we add second power of `livingArea`.
R formula:

```
price ~ 1 + livingArea + age + bedrooms +
   bathrooms + heating + new + I(livingArea^2)
```

New R^2: 0.5439 (higher than before).

Higher order terms

Even though OLS yields a “linear” model”, we can add features that are *nonlinear* transformations of the data.

Example: Suppose we add second power of `livingArea`.
R formula:

```
price ~ 1 + livingArea + age + bedrooms +
   bathrooms + heating + new + I(livingArea^2)
```

New R^2: 0.5439 (higher than before).

We could add more powers..., e.g.:
`... + I(livingArea^2) + I(livingArea^3) + I(livingArea^4)`This gives

R^2 of 0.5483 (slightly higher still).

Notice what’s happening to R^2...

Interaction terms

When changing the *value* of one covariate affects the *slope* of another, we can add an *interaction* term in the model.

Interaction terms

When changing the *value* of one covariate affects the *slope* of another, we can add an *interaction* term in the model.

E.g., consider a regression model with two covariates:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{i1} + \hat{\beta}_2 X_{i2}.$$

Interaction terms

When changing the *value* of one covariate affects the *slope* of another, we can add an *interaction* term in the model.

E.g., consider a regression model with two covariates:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{i1} + \hat{\beta}_2 X_{i2}.$$

The model with an interaction between the two covariates is:

$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_{i1} + \hat{\beta}_2 X_{i2} + \hat{\beta}_{1:2} X_{i1} X_{i2}.$$

See appendix for more details.

Adding nonlinear terms

Higher order terms and interactions are powerful techniques for making OLS models much more flexible.

A warning: now the effect of a single feature on the fitted value is captured by *multiple* coefficients!

Modern feature engineering

Besides higher order terms and interactions, there are many other transformations that abound, e.g.:

- *Logarithmic transformations*: Often used for positive, real-valued data that is strongly "skewed" (heavy-tailed), e.g., incomes, revenues/sales, counts, etc. (See appendix.)
- *Embeddings*: Algorithms that compress high-dimensional data (text, images, speech) into lower-dimensional representations

Modern feature engineering (continued)

- *Time series*: Low-dimensional representation of temporally varying data streams
- *Context-aware transformations*: Taking advantage of domain knowledge to transform data in meaningful ways (e.g., summary metrics that capture patient health based on test results)
- *LLM-generated features*: Using a foundation model to “create” features from high-dimensional data (especially text)
- Many more!

Feature engineering and R^2

When we add features and run OLS, we can only make $\boldsymbol{R}^2$ *higher*:

- OLS aims to *minimize SSE* given the features.
- If we *add* features, then the minimum SSE can only get *smaller*
- Smaller SSE $\implies$ higher $\boldsymbol{R}^2$.

Is this what we want? What would happen if we build an OLS model with as many features as we have data points?

Would such a model be a "good" predictive model?

The problem with R^2

The basic problem with $\boldsymbol{R}^2$ as a measure of predictive performance is that it is an *in-sample* measure of performance:

The same data that is used to *fit* the model is *also* used to evaluate how well it performs.

The problem with R^2

The basic problem with R^2 as a measure of predictive performance is that it is an *in-sample* measure of performance:

The same data that is used to *fit* the model is *also* used to evaluate how well it performs.
But this isn't prediction! This is just "summarizing" the data we already have.

For prediction, we need to measure how we do *out of sample*: i.e., on data that was not used to fit the model.

Out of sample evaluation

This is a central insight across all of data science:

In general, we should keep separate the data that is used to build models for prediction,
and data that is used to evaluate those same models.

Our goal is *generalization* beyond the sample, to the population.

The only way to evaluate how we generalize is to check how the model performs on data it *hasn't* already seen.

Appendix: Algebra of OLS

OLS solution

In this appendix, we assume that $\mathbf{X}$ includes an intercept term (i.e., the first column consists of all 1's).
Thus $\mathbf{X}$ has $p + 1$ columns.

We assume that $p < n$ and $\mathbf{X}$ has *full rank* $= p + 1$.

Theorem

The OLS linear regression coefficient vector $\hat{\boldsymbol{\beta}}$ that minimizes SSE *is given by:*

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y}.$$

(Check that dimensions make sense here: $\hat{\boldsymbol{\beta}}$ is $(p + 1) \times 1$.)

OLS solution: Geometry

The SSE is the squared Euclidean norm of $\mathbf{Y} - \hat{\mathbf{Y}}$:

$$\text{SSE} = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 = \|\mathbf{Y} - \hat{\mathbf{Y}}\|^2 = \|\mathbf{Y} - \mathbf{X}\hat{\boldsymbol{\beta}}\|^2.$$

Note that as we vary $\hat{\boldsymbol{\beta}}$ we range over *linear combinations of the columns of* $\mathbf{X}$.

The collection of all such linear combinations is the *subspace* spanned by the columns of $\mathbf{X}$.

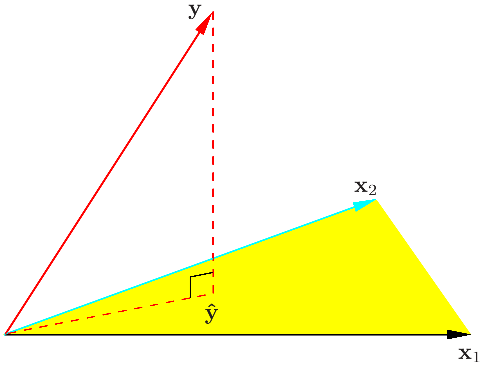
So the linear regression question is

What is the "closest" such linear combination to $\mathbf{Y}$*?*

OLS solution: Geometry

What is the "closest" such linear combination to $\mathbf{Y}$?

This "closest" combination is the *projection* of $\mathbf{Y}$ into the subspace spanned by the columns of $\mathbf{X}$:[1]



[1] Figure courtesy of *Elements of Statistical Learning*.

Hat matrix [$*$]

Since: $\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$, we have:

$$\hat{\mathbf{Y}} = \mathbf{H}\mathbf{Y},$$

where:

$$\mathbf{H} = \mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top.$$

$\mathbf{H}$ is called the *hat* matrix.

It *projects* $\mathbf{Y}$ into the subspace spanned by the columns of $\mathbf{X}$.

It is symmetric and *idempotent*, i.e., $\mathbf{H}^2 = \mathbf{H}$.

Key assumptions

We assumed that $p < n$ and $\mathbf{X}$ has *full rank* $p + 1$.

What happens if these assumptions are violated?

Collinearity

If $\mathbf{X}$ does not have full rank, then $\mathbf{X}^\top\mathbf{X}$ is *not invertible*.

In this case, the optimal $\hat{\boldsymbol{\beta}}$ that minimizes SSE is *not unique*.

The problem is that if a column of $\mathbf{X}$ can be expressed as a linear combination of other columns, then the coefficients of these columns are not uniquely determined.[2]

We refer to this problem as *collinearity*.

[2] In practice, $\mathbf{X}$ may have full rank but be *ill conditioned*, in which case the coefficients $\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{Y}$ will be very sensitive to the feature values in $\mathbf{X}$.

Collinearity: Example

If we run `lm` on a less than full rank design matrix, we obtain NA in the coefficient vector:

```
> sh$livingArea_copy = sh$livingArea
> fm = lm(data = sh, price ~ 1 + livingArea + livingArea_copy)
> coef(fm)
    (Intercept)      livingArea livingArea_copy
     13439.3940        113.1225              NA
```

High dimension

If $p \approx n$, then the number of covariates is of a similar order to the number of observations.

Assuming the number of observations is large, this is known as the *high-dimensional* regime.

When $p + 1 \geq n$, we have enough *degrees of freedom* (through the $p + 1$ coefficients) to perfectly fit the data. (What is the R^2 of such a model?)

Note that if $p \geq n$, then in general the model is nonidentifiable.

Appendix: Interaction terms and logarithmic transformations

Interaction terms

Consider the following example:

```
> fm = lm(data = sh, formula = price ~ 1 + new + livingArea)
> summary(fm)
...
Coefficients:
            Estimate ...
(Intercept) -3680.83 ...
newNo       15394.22 ...
livingArea    114.52 ...
...
```

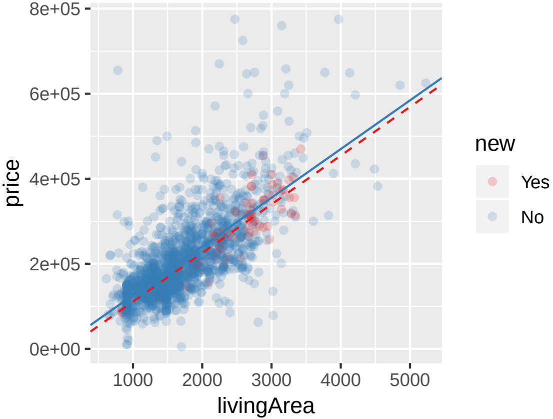
Interpretation:

- $\texttt{new} = \texttt{Yes} \implies \texttt{price} \approx \text{-}3681 + 115 \times \texttt{livingArea}$.
- $\texttt{new} = \texttt{No} \implies \texttt{price} \approx 11713 + 115 \times \texttt{livingArea}$.

Note that both have the *same slope*.

Interaction terms

Visualization:



The plot suggests *higher* slope when New = Yes.

Interaction terms

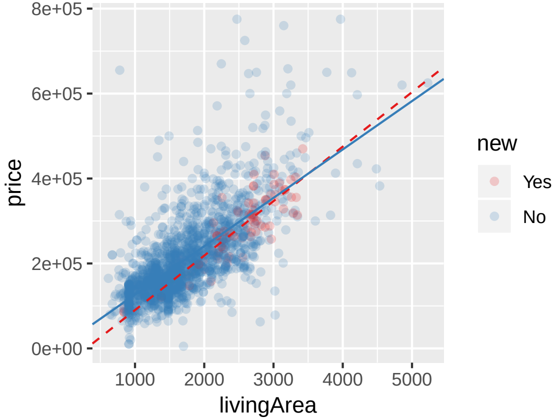
```
> fm = lm(data = sh,
          formula = price ~ 1 + new +
          livingArea + new:livingArea)
> summary(fm)
...
Coefficients:
                   Estimate ...
(Intercept)       -38045.90 ...
newNo              50804.51 ...
livingArea           128.28 ...
newNo:livingArea     -14.37 ...
...
```

Interpretation:

- When **new** = **Yes**,
  then `price` $\approx$ -38046 + 128 $\times$ `livingArea`.
- When **new** = **No**,
  then `price` $\approx$ 12759 + 114 $\times$ `livingArea`.

Interaction terms

Visualization:



This fit is improved over the previous model (slightly higher R^2).

Logarithmic transformations

In many contexts, outcomes are *positive*: e.g., physical characteristics (height, weight, etc.), counts, revenues/sales, etc.

For such outcomes linear regression can be problematic, because it can lead to a model where $\hat{Y}_i$ is negative for some $\mathbf{X}_i$.

One approach to deal with this issue is to take a *logarithmic* transformation of the data before applying OLS:

$$\log Y_i \approx \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j X_{ij}.$$

Logarithmic transformations

Exponentiating, this becomes a model that is *multiplicative*:

$$Y_i \approx e^{\hat{\beta}_0} e^{\hat{\beta}_1 X_{i1}} \dots e^{\hat{\beta}_p X_{ip}}.$$

So holding all other covariates constant, *a one unit change in X_{ij} is associated with a proportional change in the fitted value by $e^{\hat{\beta}_j}$*.

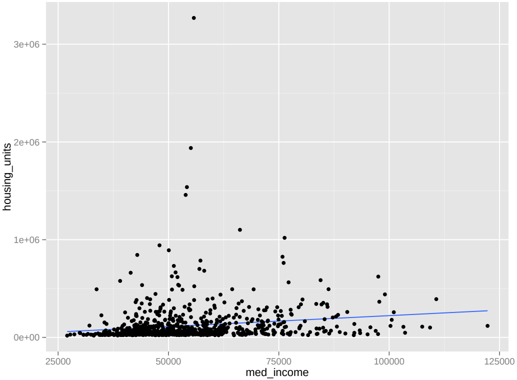
Useful intuition: $e^{\hat{\beta}_j} \approx 1 + \hat{\beta}_j$ for small $\hat{\beta}_j$, so:
A one unit change in X_{ij} is associated with a factor $\approx 1 + \hat{\beta}_j$ change in the fitted value.

Using a similar approach, can show that if both data and outcome are logged, then $\hat{\beta}_j$ gives the percentage change in the outcome associated with a one percent change in the covariate.

Logarithmic transformations: Example

Data: 2014 housing and income by county, from U.S. Census

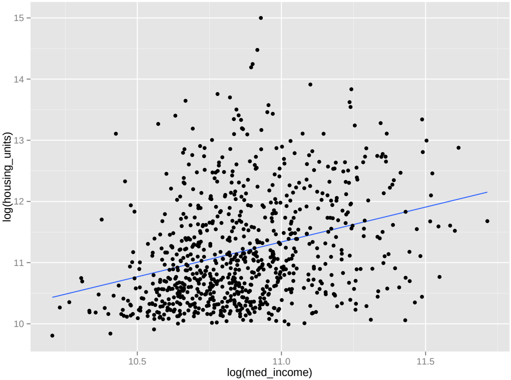
First plot number of housing units against median household income:



Logarithmic transformations: Example

Data: 2014 housing and income by county, from U.S. Census

Now do the same with logarthmically transformed data:



Logarithmic transformations: Example

Data: 2014 housing and income by county, from U.S. Census

The resulting model:

```
> fm = lm(data = income,
    formula = log(housing_units) ~ 1 + log(med_income))
> summary(fm)
...
Coefficients:
                Estimate ...
(Intercept)       -1.194 ...
log(med_income)    1.139 ...
...
```

The coefficient can be interpreted as saying that a 1% higher median household income is associated with a 1.14% higher number of housing units, on average.