Code

Linear Regression

Ruoqing Zhu

Last Updated: August 20, 2024

Example: real estate data

This Real Estate data (https://archive.ics.uci.edu/ml/datasets/Real+estate+valuation+data+set) is provided on the UCI machine learning repository (https://archive.ics.uci.edu/ml/index.php). The goal of this dataset is to predict the unit house price based on six different covariates:

- `date` : Transaction date (e.g., 2013.250 = March 2013, 2013.500 = June 2013, etc.)
- `age` : Age of the house (in years)
- `distance` : Proximity to the nearest MRT station (in meters)
- `stores` : Number of accessible convenience stores on foot (integer)
- `latitude` : Latitude (in degrees)
- `longitude` : Longitude (in degrees)
- `price` : Price per unit area of the house

Hide

```
    realestate = read.csv("realestate.csv", row.names = 1)
    library(DT)
    datatable(realestate, filter = "top", rownames = FALSE, options = list(pageLeng
th = 8))
```

Show 8 **entries** **Search:**

date	age	distance	stores	latitude	longitude	price
Al	A	All	A	All	All	A
2012.917	32	84.87882	10	24.98298	121.54024	37.9
2012.917	19.5	306.5947	9	24.98034	121.53951	42.2

date	age	distance	stores	latitude	longitude	price
2013.583	13.3	561.9845	5	24.98746	121.54391	47.3
2013.5	13.3	561.9845	5	24.98746	121.54391	54.8
2012.833	5	390.5684	5	24.97937	121.54245	43.1
2012.667	7.1	2175.03	3	24.96305	121.51254	32.1
2012.667	34.5	623.4731	7	24.97933	121.53642	40.3
2013.417	20.3	287.6025	6	24.98042	121.54228	46.7

Showing 1 to 8 of 414 entries Previous 1 2 3 4 5 … 52 Next

Hide

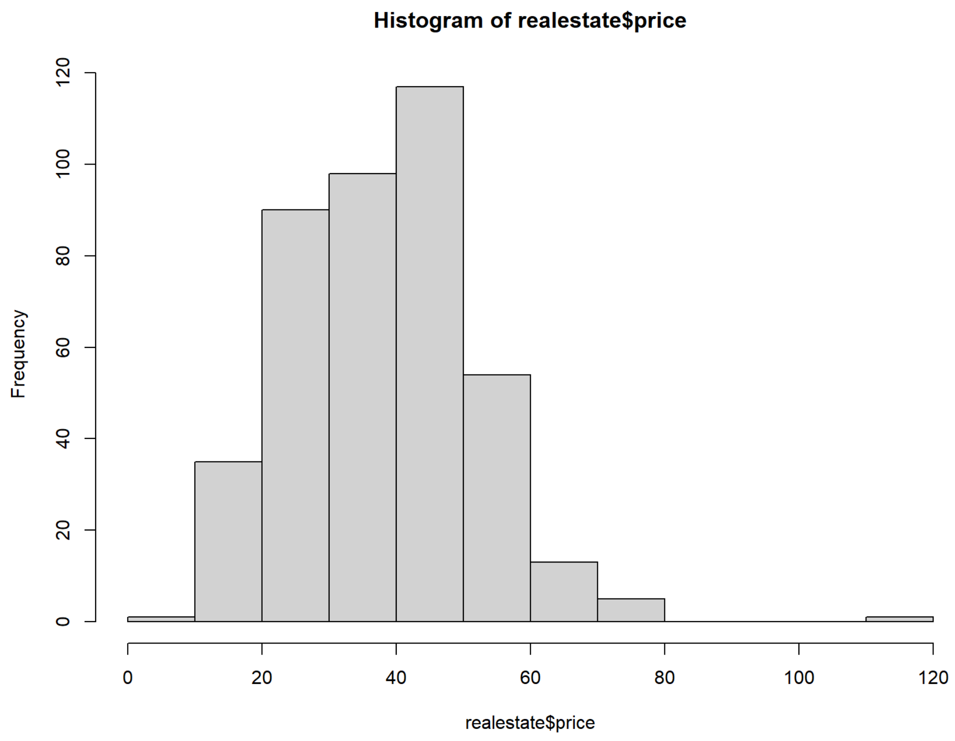
```
    dim(realestate)
## [1] 414   7
```

Summary Statistics

Understanding the data is the first crucial step in analysis. Summary statistics, visual figures, and tables can aid in this understanding. Identify categorical variables, consider transformations for continuous variables, and observe any outliers could be some initial steps you want to consider. These understandings could lead to a better data cleaning/processing.

Hide

```
    hist(realestate$price)
```



Hide

```
    apply(realestate, 2, class)
##       date       age  distance    stores  latitude longitude     price 
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
```

Though these variables are read in as numerical, `stores` behaves similarly to a categorical variable but can also be treated as continuous. Some categories might be relatively sparse. Reducing them to a continuous variable or combining some categories could reduce the number of variables in the model.and lead to better statistical performance (to be discussed later).

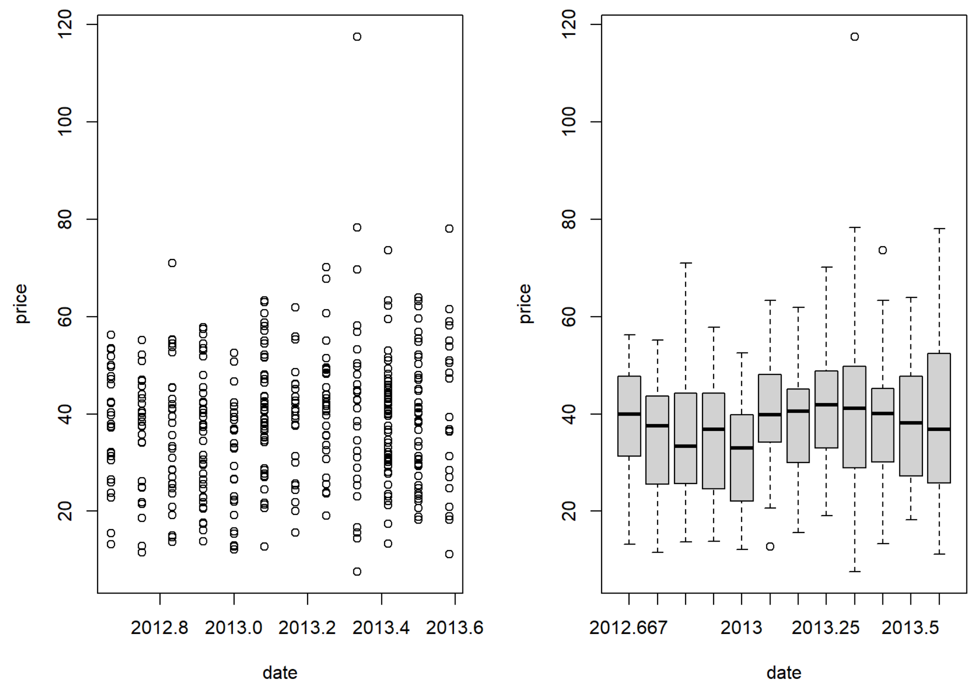
Hide

```
    table(realestate$stores)
## 
##  0  1  2  3  4  5  6  7  8  9 10 
## 67 46 24 46 31 67 37 31 30 25 10
    table(realestate$stores > 5, realestate$age > 10)
##        
##         FALSE TRUE
##   FALSE    62  219
##   TRUE     48   85
```

The `date` variable is continuous but formatted with ordered unique values. Boxplots can visualize such variables effectively.

Hide

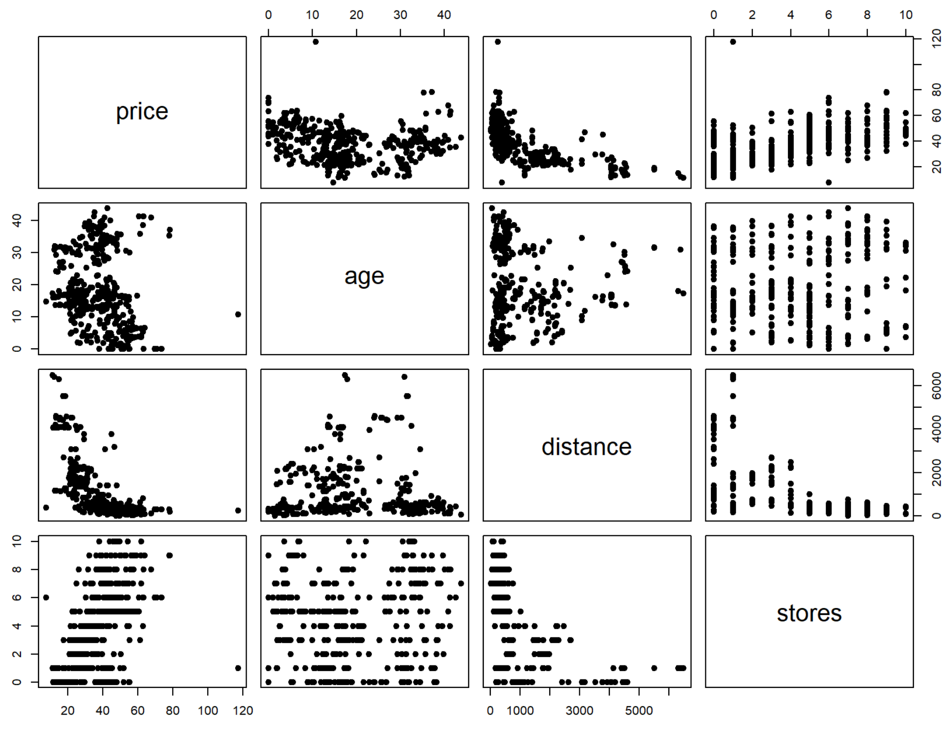
```
    par(mfrow = c(1, 2))
    plot(price ~ date, data = realestate)
    boxplot(price ~ date, data = realestate)
```



Correlation analysis is commonly used to understand the relationship between variables. The `pairs()` function can be used to visualize the pairwise scatterplots. We can see that `price` is negatively correlated with `distance`.

Hide

```
pairs(realestate[, c("price", "age", "distance", "stores")], pch = 19)
```



Basic Concepts

We generally represent the observed covariates information as the design matrix, denoted by $\mathbf{X}$, with dimensions $n \times p$. In this specific example, the dimension of $\mathbf{X}$ is 414×7. Each variable is represented by a column in this matrix; for instance, the jth variable corresponds to the jth column and is denoted as $\mathbf{x}_j$. The outcome $\mathbf{y}$ (price) is a vector of length 414. It's important to note that we typically use a "bold" symbol to represent a vector, whereas a single element (scalar), such as the jth variable of subject i, is represented as xij.

Linear regression models the relationship in matrix form as:

$$\mathbf{y}_{n\times 1} = \mathbf{X}_{n\times p}\boldsymbol{\beta}_{p\times 1} + \boldsymbol{\epsilon}_{n\times 1}$$

And we know that the solution is obtained by minimizing the residual sum of squares (RSS):

$$\begin{aligned}\widehat{\boldsymbol{\beta}} &= \underset{\boldsymbol{\beta}}{\arg\min} \sum_{i=1}^{n} \left(y_i - x_i^{\mathrm{T}}\boldsymbol{\beta}\right)^2 \\ &= \underset{\boldsymbol{\beta}}{\arg\min} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)^{\mathrm{T}}\left(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\right)\end{aligned}$$

This is an optimization problem, and in fact, it is a convex function of $\boldsymbol{\beta}$ if the matrix $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is invertible. This will be explained later in the optimization lecture. The classical solution of a linear regression can be obtained by taking the derivative of RSS w.r.t $\boldsymbol{\beta}$ and setting it to zero. This leads to the well known normal equation:

$$\begin{aligned} \frac{\partial \mathrm{RSS}}{\partial \boldsymbol{\beta}} &= -2\mathbf{X}^{\mathrm{T}}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \overset{\text{set}}{=} 0 \\ \Longrightarrow \quad \mathbf{X}^{\mathrm{T}}\mathbf{y} &= \mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\beta}\end{aligned}$$

Assuming that $\mathbf{X}$ is of full rank, then $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is invertible, leading to:

$$\widehat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Using the `lm()` Function

Let's consider a regression model using `age` and `distance` to predict `price`. We can use the `lm()` function to fit this linear regression model.

Hide

```
lm.fit = lm(price ~ age + distance, data = realestate)
```

This syntax contains three components:

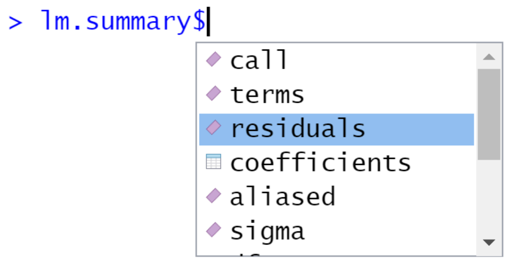
- `data =` specifies the dataset
- The outcome variable should be on the left-hand side of `~`
- The covariates should be on the right-hand side of `~`

Detailed model fitting results are saved in `lm.fit`. To view the model fitting results, we use the `summary()` function.

Hide

```
    lm.summary = summary(lm.fit)
    lm.summary
##
## Call:
## lm(formula = price ~ age + distance, data = realestate)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -36.032  -4.742  -1.037   4.533  71.930
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 49.8855858  0.9677644  51.547  < 2e-16 ***
## age         -0.2310266  0.0420383  -5.496 6.84e-08 ***
## distance    -0.0072086  0.0003795 -18.997  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.73 on 411 degrees of freedom
## Multiple R-squared:  0.4911, Adjusted R-squared:  0.4887
## F-statistic: 198.3 on 2 and 411 DF,  p-value: < 2.2e-16
```

This shows that both `age` and `distance` are highly significant for predicting the unit price. In fact, this fitted object (`lm.fit`) and the summary object (`lm.summary`) are both saved as a list. This is pretty common to handle an output object with many things involved. You can peek into these objects to explore what information they contain by using the `$` operator followed by the specific element's name.

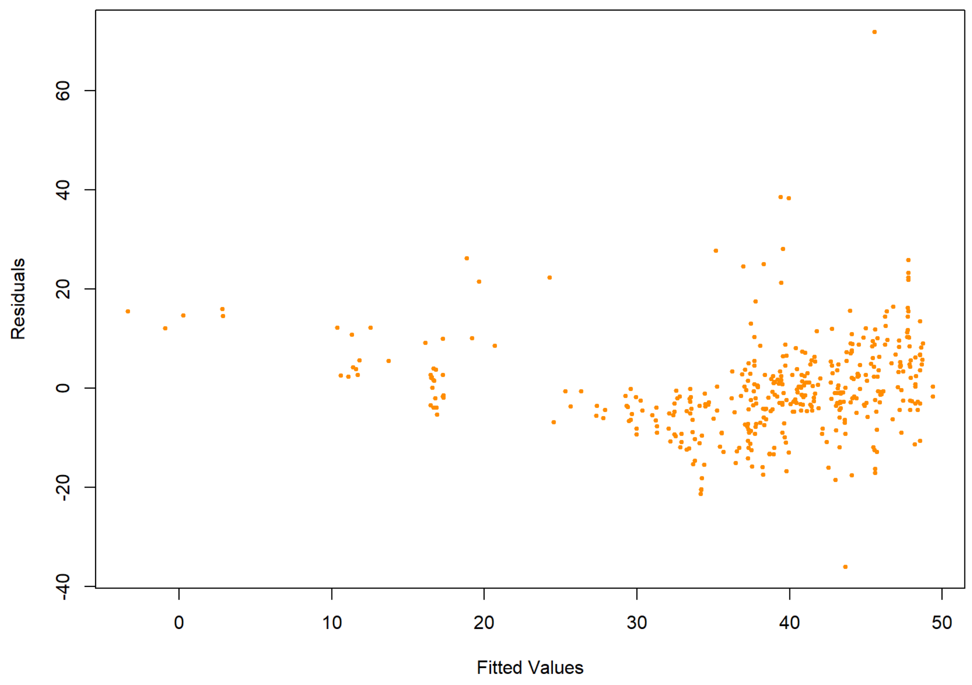


```
> lm.summary$
```

Usually, printing out the summary is sufficient. However, further details can be useful for other purposes. For example, if we are interested in the residual vs. fit plot, we may use

Hide

```
    plot(lm.fit$fitted.values, lm.fit$residuals,
         xlab = "Fitted Values", ylab = "Residuals",
         col = "darkorange", pch = 19, cex = 0.5)
```



It seems that the error variance is not constant (as a function of the fitted values). Hence additional techniques may be required to handle this issue. However, that is beyond the scope of this book.

Specifying Models

Higher-order terms and Interactions

It is pretty simple if we want to include additional variables. This is usually done by connecting them with the `+` sign on the right-hand side of `~`. R also provides convenient ways to include interactions and higher-order terms. The following code with the interaction term between `age` and `distance`, and a squared term of `distance` should be self-explanatory.

Hide

```
    lm.fit2 = lm(price ~ age + distance + age*distance + I(distance^2), data = real
estate)
    summary(lm.fit2)
##
## Call:
## lm(formula = price ~ age + distance + age * distance + I(distance^2),
##     data = realestate)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.117  -4.160  -0.594   3.548  69.683
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.454e+01  1.099e+00  49.612  < 2e-16 ***
## age           -2.615e-01  4.931e-02  -5.302 1.87e-07 ***
## distance      -1.603e-02  1.133e-03 -14.152  < 2e-16 ***
## I(distance^2)  1.907e-06  2.416e-07   7.892 2.75e-14 ***
## age:distance   8.727e-06  4.615e-05   0.189     0.85
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.939 on 409 degrees of freedom
## Multiple R-squared:  0.5726, Adjusted R-squared:  0.5684
## F-statistic:   137 on 4 and 409 DF,  p-value: < 2.2e-16
```

If you choose to include all covariates presented in the data, simply use `.` on the right-hand side of `~`. However, you should always be careful when doing this because some datasets would contain meaningless variables such as subject ID.

Hide

```
    lm.fit3 = lm(price ~ ., data = realestate)
```

Categorical Variables

The `store` variable has several different values. We can see that it has 11 different values. One strategy is to model this as a continuous variable. However, we may also consider discretizing it. For example, we may create a new variable, say `store.cat`, defined as follows.

Hide

```
  table(realestate$stores)
##
##  0  1  2  3  4  5  6  7  8  9 10
## 67 46 24 46 31 67 37 31 30 25 10

  # define a new factor variable
   realestate$store.cat = as.factor((realestate$stores > 0) + (realestate$stores >
4))
  table(realestate$store.cat)
##
##   0   1   2
##  67 147 200
  levels(realestate$store.cat) = c("None", "Several", "Many")
  head(realestate$store.cat)
## [1] Many    Many    Many    Many    Many    Several
## Levels: None Several Many
```

This variable is defined as a factor, which is often used for categorical variables. Since this variable has three different categories, if we include it in the linear regression, it will introduce two additional variables (using the third as the reference):

Hide

```
    lm.fit3 = lm(price ~ age + distance + store.cat, data = realestate)
    summary(lm.fit3)
##
## Call:
## lm(formula = price ~ age + distance + store.cat, data = realestate)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -38.656  -5.360  -0.868   3.913  76.797
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      43.712887   1.751608  24.956  < 2e-16 ***
## age              -0.229882   0.040095  -5.733 1.91e-08 ***
## distance         -0.005404   0.000470 -11.497  < 2e-16 ***
## store.catSeveral  0.838228   1.435773   0.584     0.56
## store.catMany     8.070551   1.646731   4.901 1.38e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.279 on 409 degrees of freedom
## Multiple R-squared:  0.5395, Adjusted R-squared:  0.535
## F-statistic: 119.8 on 4 and 409 DF,  p-value: < 2.2e-16
```

There are usually two types of categorical variables:

- Ordinal: the numbers representing each category are ordered, e.g., how many stores are in the neighborhood. Oftentimes nominal data can be treated as a continuous variable.

- Nominal: they are not ordered and can be represented using either numbers or letters, e.g., ethnic group.

The above example treats `store.cat` as a nominal variable, and the `lm()` function uses dummy variables for each category. However, many other machine learning models and packages could treat them differently.

Interpreting Interactions

Oftentimes, we may be interested in adding interaction terms. This should be relatively simple with continuous variables. However, sometimes confusing for categorical variables. For example, let’s use an iteration term between `distance` and `store.cat`.

Hide

```
    lm.fit4 = lm(price ~ age + distance + store.cat + distance*store.cat, data = re
alestate)
    summary(lm.fit4)
##
## Call:
## lm(formula = price ~ age + distance + store.cat + distance *
##     store.cat, data = realestate)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -37.042  -4.984  -1.089   4.042  76.545
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               40.6792018  2.1334932  19.067  < 2e-16 ***
## age                       -0.2043677  0.0391483  -5.220 2.85e-07 ***
## distance                  -0.0043455  0.0006911  -6.287 8.33e-10 ***
## store.catSeveral           3.9353304  2.3360099   1.685   0.0928 .
## store.catMany             16.9437667  2.4458927   6.927 1.68e-11 ***
## distance:store.catSeveral -0.0014050  0.0009238  -1.521   0.1291
## distance:store.catMany    -0.0209723  0.0039689  -5.284 2.06e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.991 on 407 degrees of freedom
## Multiple R-squared:  0.5697, Adjusted R-squared:  0.5634
## F-statistic: 89.82 on 6 and 407 DF,  p-value: < 2.2e-16
```

There are two additional terms representing the interaction between two categories: `Several` and `Many` with the term `distance`. Again, the `None` category is used as the reference. Hence, when an observation is of the `Several` category, it will “activate” both the `store.catSeveral` parameter and also the `distance:store.catSeveral` parameter.

In many of the later topics in this course, we will not rely on `R` functions to manage interaction or categorical variables for us. Instead, you can create additional columns in the dataset (feature engineering) based on your specific needs. And use them directly in a model fitting. The following code creates the variables used in the example above.

Hide

```
    newdata = cbind("Age" = realestate$age, "Distance" = realestate$distance,
                    "Several" = realestate$store.cat == "Several",
                    "Many" = realestate$store.cat == "Many",
                      "Dist_Several" = realestate$distance*(realestate$store.cat ==
"Several"),
                    "Dist_Many" = realestate$distance*(realestate$store.cat == "Man
y"))
    head(newdata)
##       Age   Distance Several Many Dist_Several Dist_Many
## [1,] 32.0   84.87882       0    1         0.00  84.87882
## [2,] 19.5  306.59470       0    1         0.00 306.59470
## [3,] 13.3  561.98450       0    1         0.00 561.98450
## [4,] 13.3  561.98450       0    1         0.00 561.98450
## [5,]  5.0  390.56840       0    1         0.00 390.56840
## [6,]  7.1 2175.03000       1    0      2175.03   0.00000
```

Prediction

We can use the fitted model to predict future subjects. There are typically two quantities we are interested in:

- In-sample prediction: use the model to predict the outcome of the subjects used in the model fitting. The result is called training error.
- Out-of-sample prediction: use the model to predict future subjects. The result is called testing error.

These two predictions have very different meanings and behavior. The in-sample prediction is usually used to evaluate the model fitting performance. And since you intentionally chose the model to fit them well, the in-sample prediction is often more accurate. The out-of-sample prediction is often difficult. We will return to this topic in later lectures.

For in-sample prediction of linear regression, we can use the `predict()` function without specifying any additional arguments.

Hide

```
    # the fitted values
    fitted = predict(lm.fit)
    head(fitted)
##        1        2        3        4        5        6
## 41.88088 43.17044 42.76180 42.76180 45.91499 32.56633
    # the training error
    mean((fitted - realestate$price)^2)
## [1] 93.97977
```

For out-of-sample prediction we set up a new dataset with the same format as the training data. For example, taking the model with just `age` and `distance` may construct a new dataset containing columns with these names.

Hide

```
  newdata = data.frame("age" = c(0, 10, 20, 30), "distance" = c(100, 500, 1000, 200
0))
  predict(lm.fit, newdata)
##        1        2        3        4
## 49.16472 43.97101 38.05643 28.53755
```