Machine Learning in Public Health

Lecture 2: Linear Regression

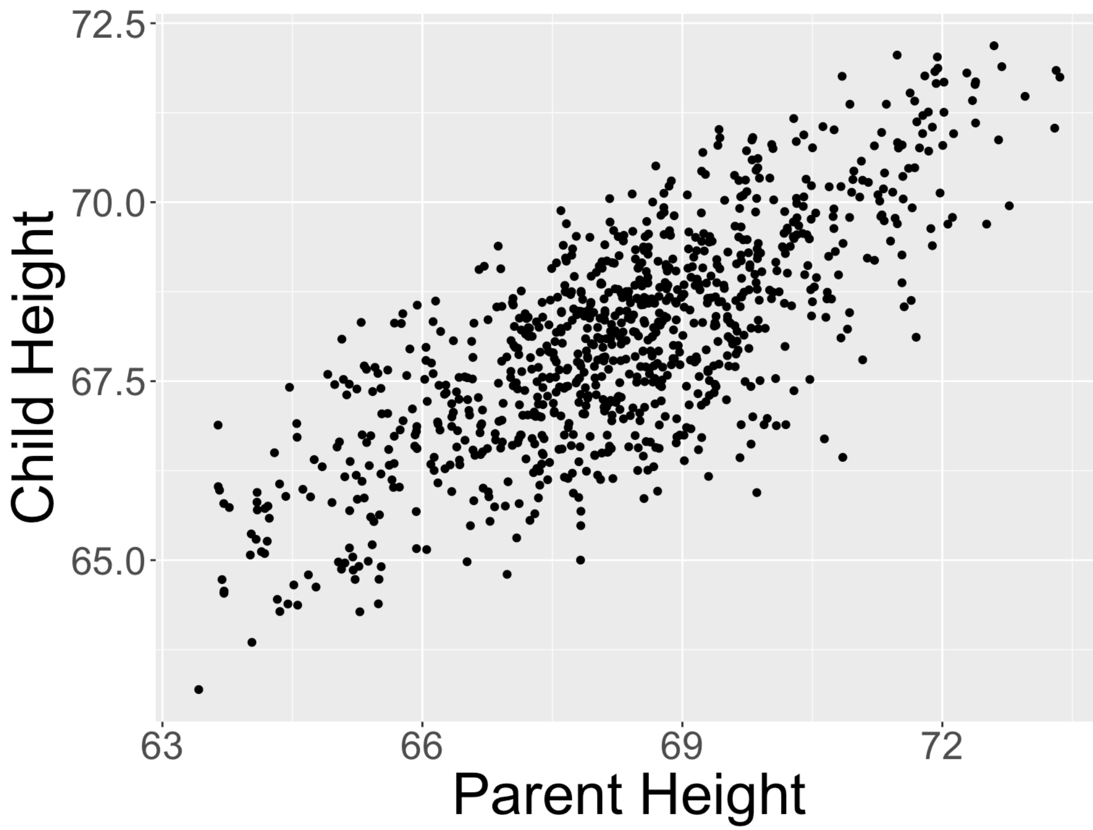
Dr. Yang Feng

Happy New Year

Today's agenda

- Simple Linear Regression
- Multiple Linear Regression
- Qualitative Predictors
- K-Nearest Neighbors for Regression

Linear regression with a single variable

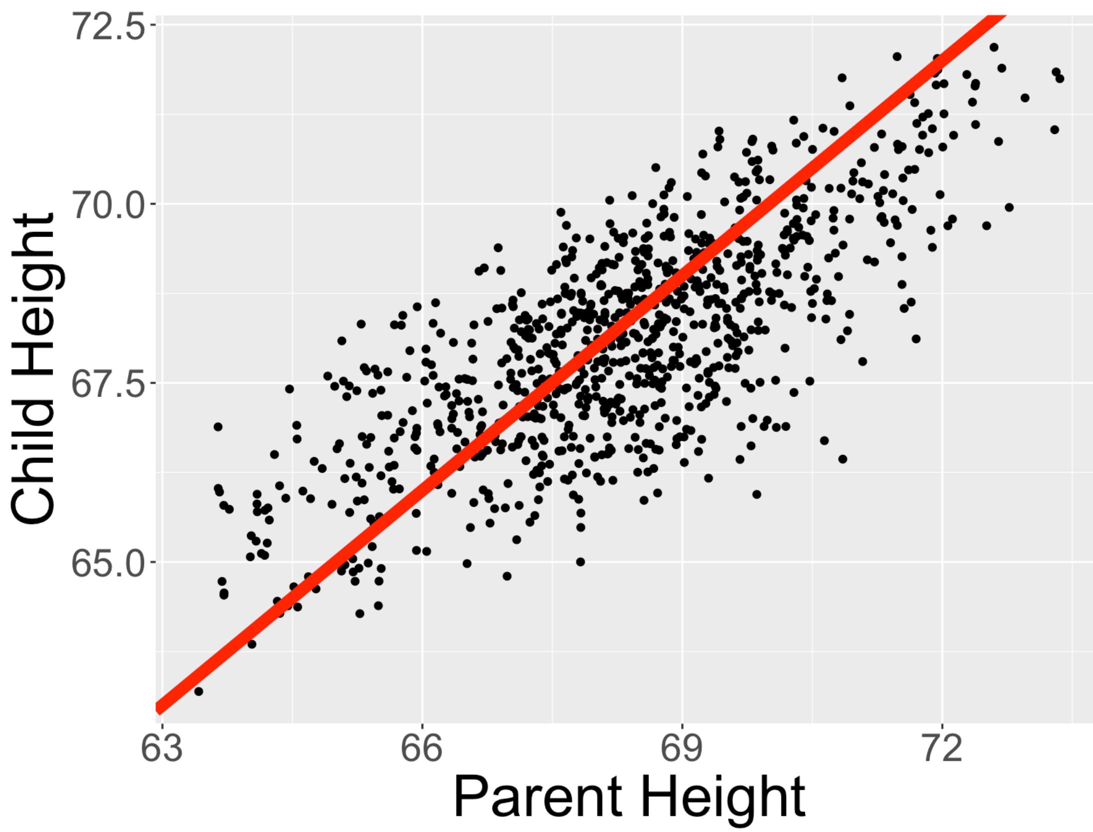


- Goal: predict child's height Y from parent's height X
  - *Call Y the **reponse**, also called **dependent** variable*
  - *Call X the **predictor**, also called **independent** variable*

Predict child's heights using parent's heights

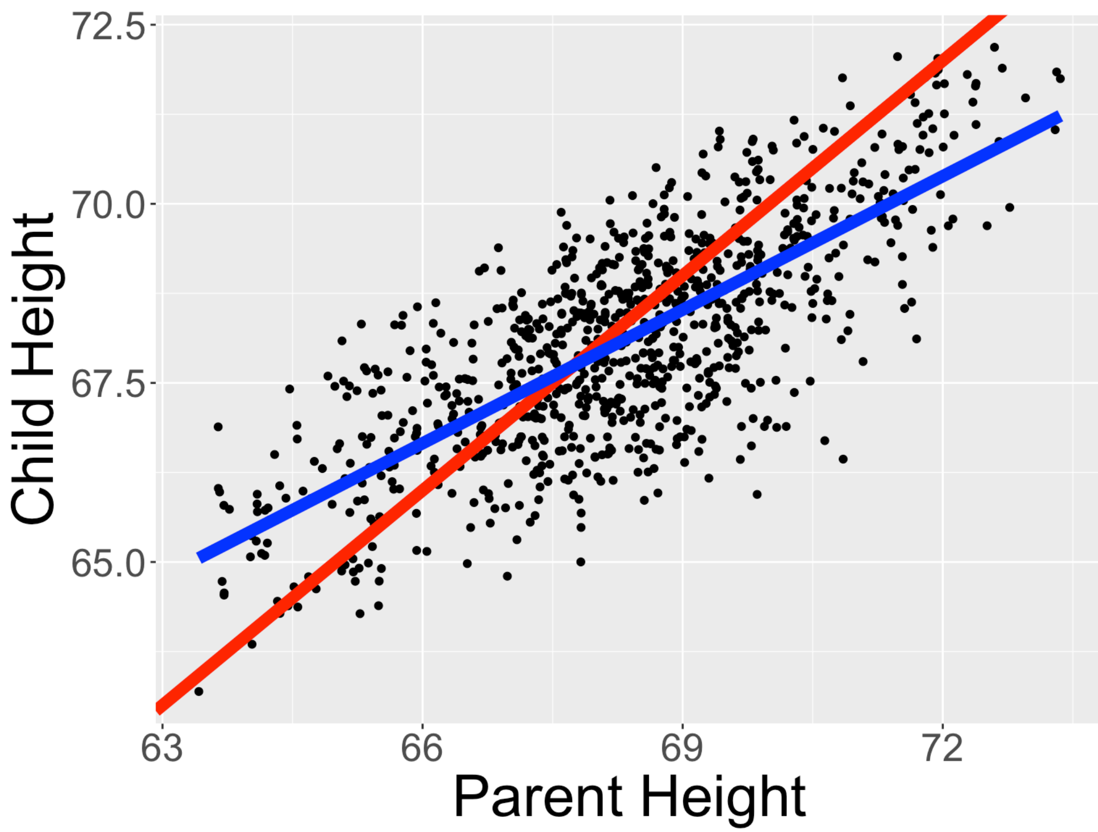
- A naive idea:

$$\widehat{Y} = X$$

- 

Predit child's heights using parent's heights

- Probably a better idea: $\widehat{Y} = \underbrace{\widehat{\beta}_0}_{\text{intercept}} + \underbrace{\widehat{\beta}_1}_{\text{slope}} X$
- The **best** fit line of this form. (what does "best" mean?)
-



Simple Linear Regression Model

George Box:

All models are wrong; some models are useful…

- Predict a quantitative response Y using a single predictor X.
- The simplest of all is a (simple) *linear* regression model; that is, the *response* or *target* Y satisfies

$$Y = \beta_0 + \beta_1 X + \epsilon$$

- β_0 is the *intercept* term
- β_1 is the *slope*
- β_0 and β_1 are *coefficients* or *parameters* of the linear model. They are usually **unknown**.
- ϵ is a noise term, which is usually assumed independent of x and mean zero. It is often assumed to be normally distributed in theoretical analysis.
- Often, we assume the variance of ε is σ^2, which is another **unknown** parameter of the simple linear regression model.

Estimating the Coefficients (1)

- **Data**: Suppose the training data contains $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$, where x_i is the height of parent i and y_i is height of child i. n is the **sample size**.
-

	child	parent
	<dbl>	<dbl>
1	66.43592	70.85107
2	65.94336	69.85889
3	64.27886	65.27814
4	63.85191	64.03263
5	63.19229	63.41899
6	65.00112	67.82480

6 rows

- **Predictions**: For an estimate $\hat{\beta}_0$ and $\hat{\beta}_1$, the prediction is $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ of the height of child if a parent has height x.
- **Loss function**: How do we measure the quality of the fit. First, define the *residual* $e_i = y_i - \hat{y}_i$.

$$Loss(y, \hat{y}) = (y - \hat{y})^2$$

- Why does this loss function make sense? It is the only loss function that makes sense?

Estimating the Coefficients (2)

- **Training Loss**: we can just sum up the loss for all n observations.

$$
\begin{aligned}
L(\hat{\beta}_0, \hat{\beta}_1) &= \sum_{i=1}^{n} Loss(y_i, \hat{y}_i) \\
&= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\
&= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2
\end{aligned}
$$

- Find $\hat{\beta}_0$ and $\hat{\beta}_1$ that **minimizes** the overall training loss.

$$\underset{\beta_0, \beta_1}{\text{minimize}}\ L(\beta_0, \beta_1)$$

- The minimizer is called the **least squares estimate**, denoted as $\hat{\beta}_0$ and $\hat{\beta}_1$.

Residual Sum of Squares and Total Sum of Squares

- Residual Sum of Squares

$$
\begin{aligned}
RSS &= \sum_{i=1}^{n} e_i^2 \\
&= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \\
&= (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + \cdots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2
\end{aligned}
$$

- Here, e_i is the residual for the i-th observation.
- Total Sum of Squares: corresponding to $\hat{\beta}_0 = \bar{y}$ and $\hat{\beta}_1 = 0$.

$$
TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2
$$

R^2: coefficient of determination

$$R^2 = 1 - \frac{RSS}{TSS}$$

- $R^2 \in [0, 1]$ is the percent of the variation in the response explained by the regression model
- R^2 is a common measure for how good a linear fit is.
- Q: is a bigger R^2 always better?

Least Square Estimates

- Exact formula

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \text{ and } \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

  - *Remark: every least squares regression line passes $(\bar{x}, \bar{y})$.*

- Optimization techniques.

$$\frac{\partial L(\beta_0, \beta_1)}{\partial \beta_0} = 0$$
$$\frac{\partial L(\beta_0, \beta_1)}{\partial \beta_1} = 0$$

Multiple Regression: more than one predictors

In addition to scalar input $x \in \mathbb{R}$, we can consider vector input $x \in \mathbb{R}^p$

- p is feature dimension of input (sometimes use d for *dimension*)
- Inputs of form

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}$$

- Call x the *covariates*, *independent variables*, *explanatory variables*, *features*, *attributes* or *predictor variables*
    - *Note that variables are usually NOT independent of one another*

Example: Boston housing data

- Output (response, target, dependent) variable is medv, the median home price in neighborhoods of Suburbs of Boston
- Input variables include (but not limited to)
  - *crim: per capita crime rate by town*
  - *rm: average number of rooms per dwelling*
  - *zn: proportion of large lots (zoned for* $> 25,000$ *feet)*
  - *chas: whether a home is near the Charles river* ($x \in \{0, 1\}$)
  - *ptratio: pupil-teacher ratio by town*
- Let's do a little data exploration

	crim	rm	zn	chas	ptratio	medv
	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>
1	0.00632	6.575	18	0	15.3	24.0
2	0.02731	6.421	0	0	17.8	21.6
3	0.02729	7.185	0	0	17.8	34.7
4	0.03237	6.998	0	0	18.7	33.4
5	0.06905	7.147	0	0	18.7	36.2
6	0.02985	6.430	0	0	18.7	28.7

6 rows

Boston housing: Number of Bedrooms

Idea: let us run a regression on each of these, and see which is best, first, the number of bedrooms `rm`.

```
fit_rm <- lm(medv ~ rm, data = boston)
fit_rm
```

```
##
## Call:
## lm(formula = medv ~ rm, data = boston)
##
## Coefficients:
## (Intercept)           rm
##     -34.671        9.102
```

- To run a regression on a data frame, we tell R to use a formula with the column(s) we care about and the data frame we want.

Measuring the fit

Compute its fitted values $\hat{y}_i$ and residuals

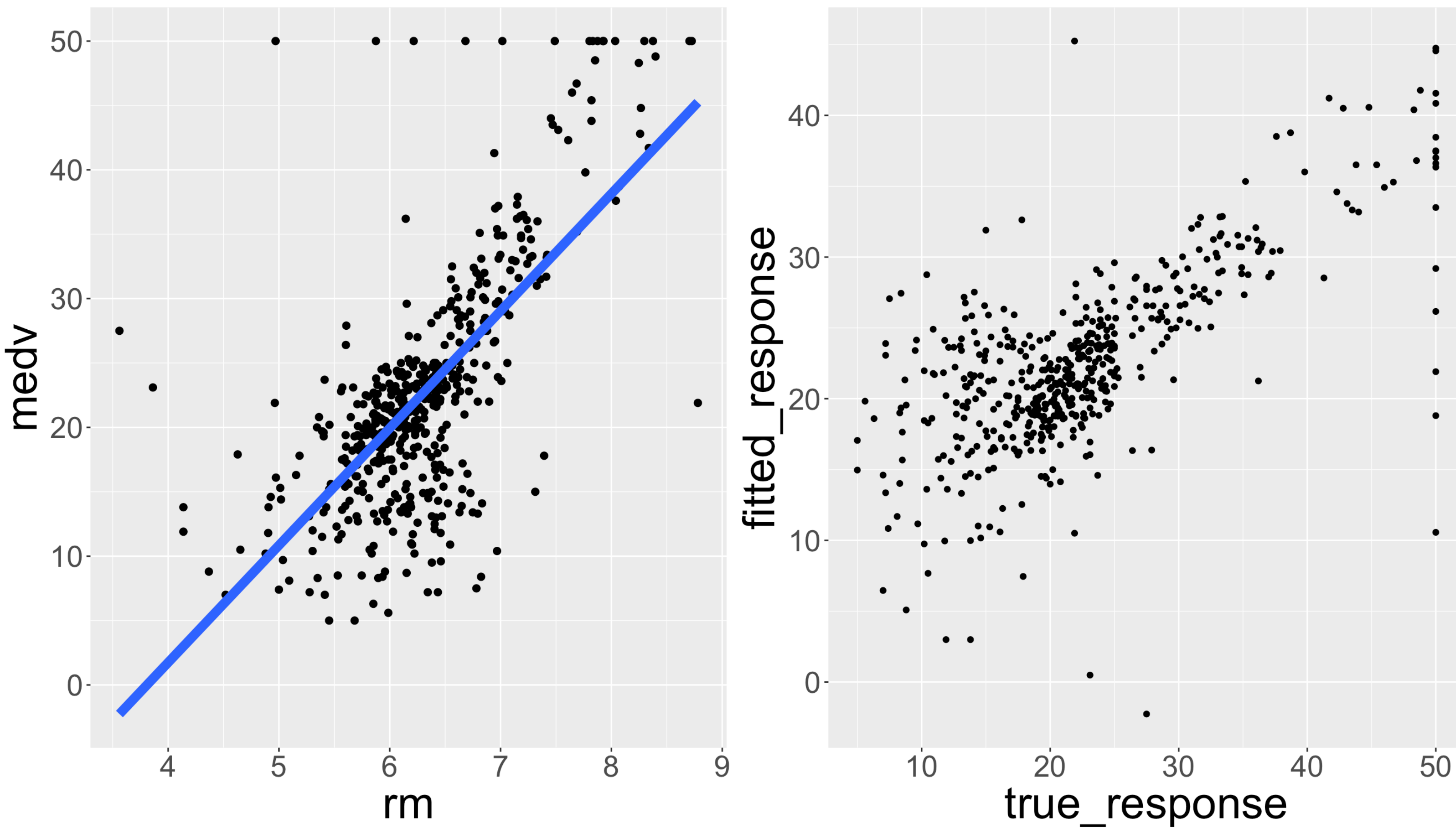
```
df_rm <- data.frame(true_response = boston$medv,
           fitted_response = fit_rm$fitted.values,
           residual = fit_rm$residuals)
head(df_rm)
```

	true_response	fitted_response	residual
	<dbl>	<dbl>	<dbl>
1	24.0	25.17575	-1.175746
2	21.6	23.77402	-2.174021
3	34.7	30.72803	3.971968
4	33.4	29.02594	4.374062
5	36.2	30.38215	5.817848
6	28.7	23.85594	4.844060

6 rows

Its $R^2 = 0.4835255$

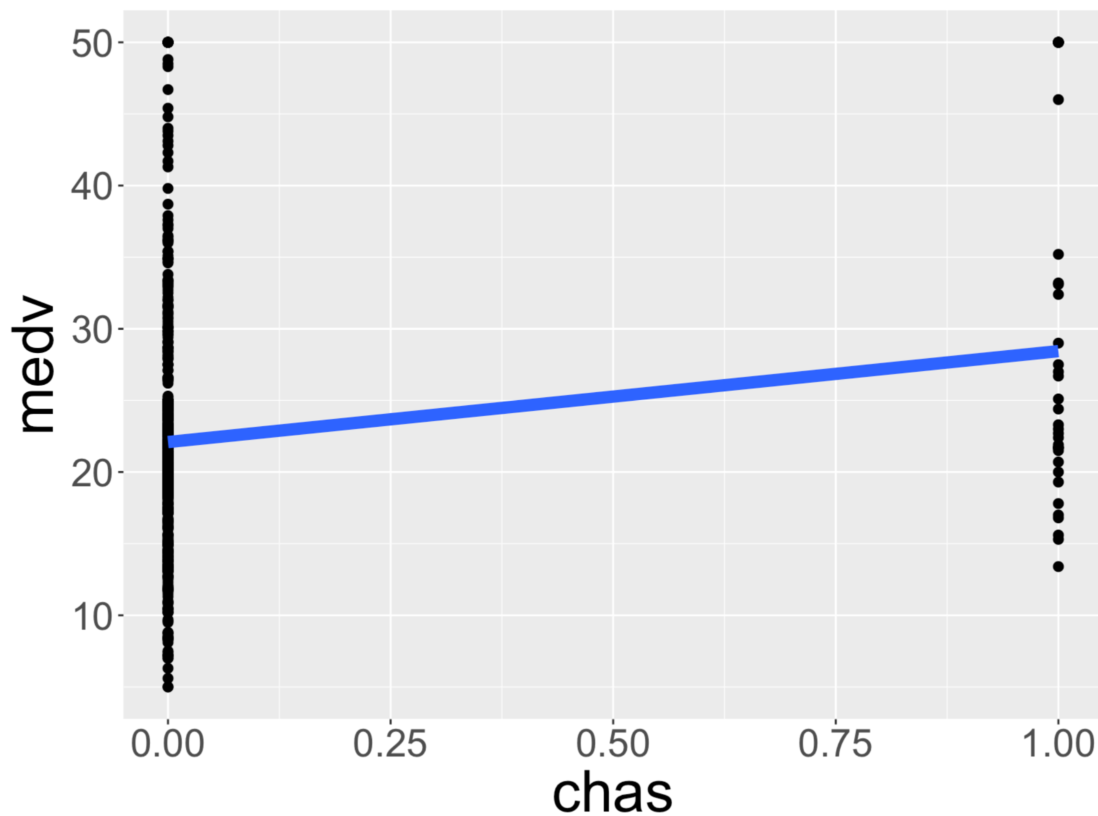
Visualization



Boston housing: Charles River

```
fit_chas <- lm(medv ~ chas, data = boston)
fit_chas
```

```
##
## Call:
## lm(formula = medv ~ chas, data = boston)
##
## Coefficients:
## (Intercept)         chas
##      22.094        6.346
```

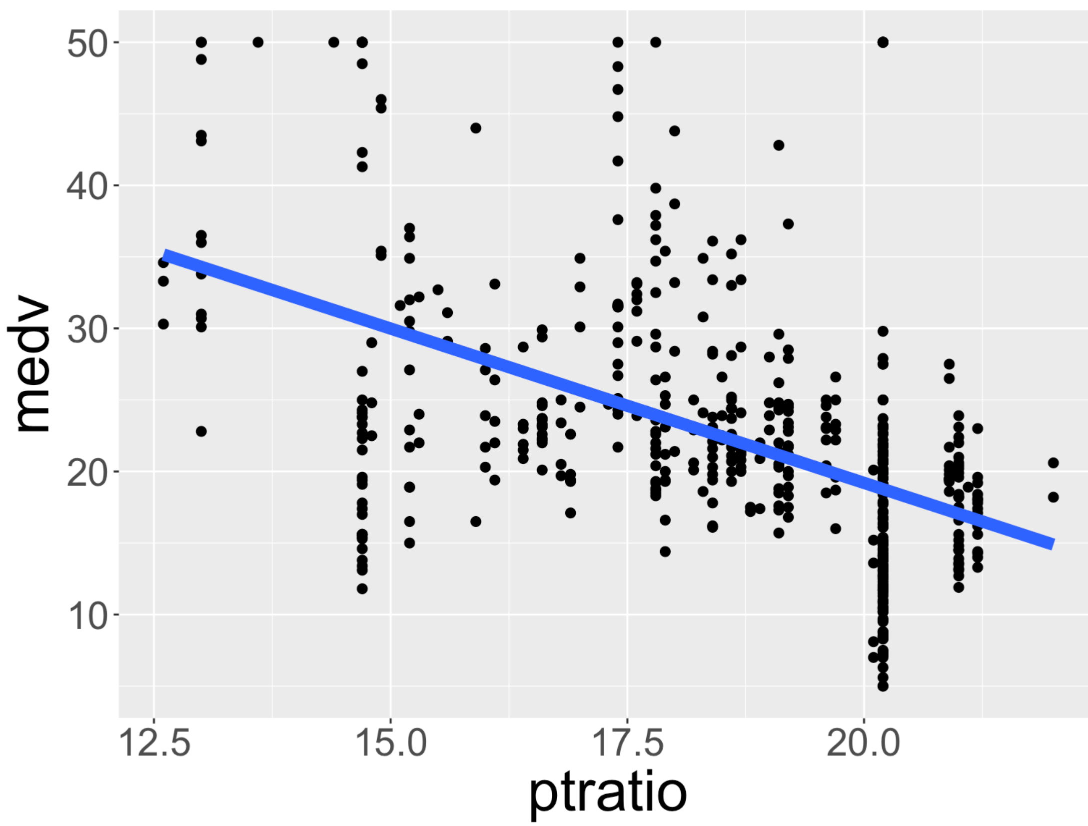


Its $R^2 = 0.0307161$

Boston housing: Student to teacher ratio

```r
fit_ptratio <- lm(medv ~ ptratio, data = boston)
fit_ptratio
```

```
##
## Call:
## lm(formula = medv ~ ptratio, data = boston)
##
## Coefficients:
## (Intercept)      ptratio
##      62.345       -2.157
```



Its $R^2 = 0.2578473$

When we use more than one input variables

- Make predictions

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j$$

- Model the data as

$$y = \beta_0 + \sum_{j=1}^{p} \beta_j x_j + \varepsilon$$

where ε is the random noise.

Fitting a multiple regression

- Making predictions using

$$\hat{y} = \hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_j$$

- Fit as in single variable case. Solve (least squares criterion)

$$\underset{\beta_0 \in \mathbb{R}, \beta \in \mathbb{R}^p}{\text{minimize}} \sum_{i=1}^{n} (y_i - \beta_0 - \beta^T x_i)^2$$

- Since each observation has p values, $x_i = (x_{i1}, \cdots, x_{ip})^T$, where $i = 1, \cdots, n$. In matrix notation, let $y = (y_1 \cdots, y_n)^T$ and

$$X = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{13} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & x_{23} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & x_{n3} & \dots & x_{np} \end{bmatrix}$$

where x_{ij} is the ith observation of the jth variable.

- LS Solution

$$\hat{\beta} = (X^T X)^{-1} X y.$$

Boston housing: number of bedrooms and student to teacher ratio

```
fit_rm_ptratio <- lm(medv ~ rm + ptratio, data = boston)
fit_rm_ptratio
```

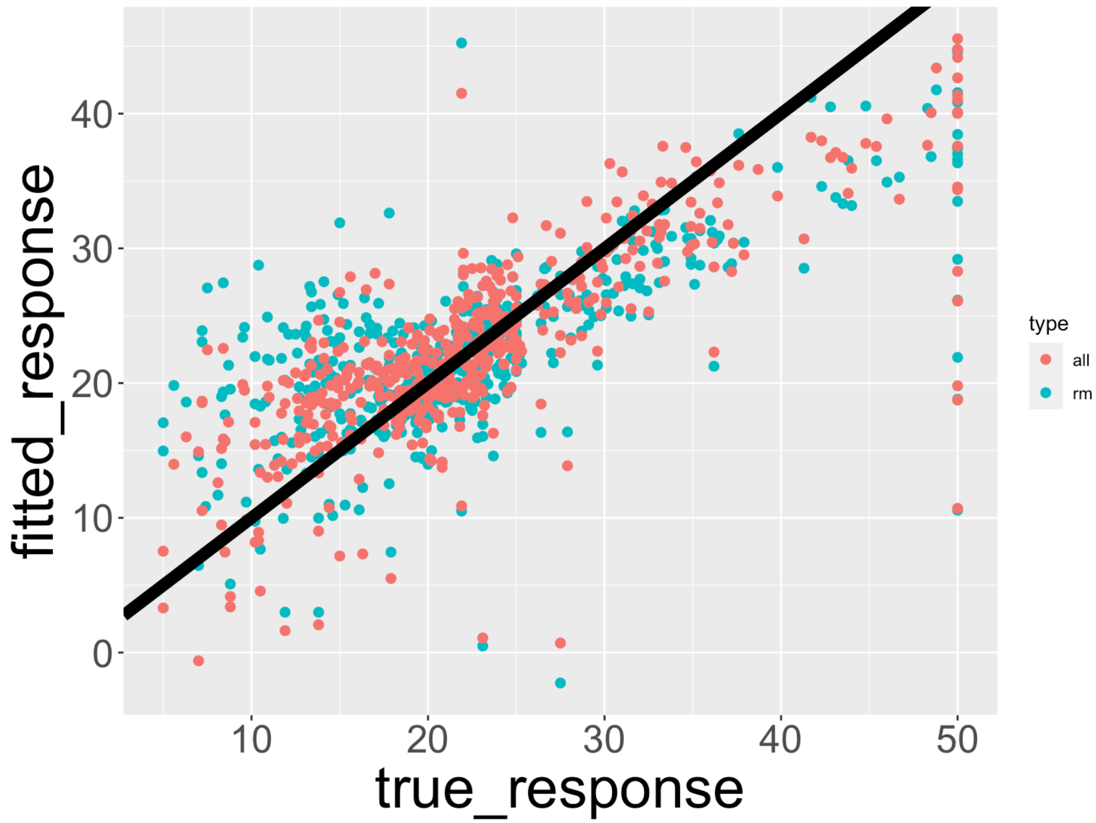
```
##
## Call:
## lm(formula = medv ~ rm + ptratio, data = boston)
##
## Coefficients:
## (Intercept)           rm      ptratio
##      -2.561        7.714       -1.267
```

```
summary(fit_rm_ptratio)$r.squared
```

```
## [1] 0.5612535
```

Housing data: regression on everything

```
fit_all <- lm(formula = medv ~ ., data = boston)
```



Summary of `lm`

```
summary(fit_rm_ptratio)
```

```
##
## Call:
## lm(formula = medv ~ rm + ptratio, data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.672  -2.821   0.102   2.770  39.819
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.5612     4.1889  -0.611    0.541
## rm            7.7141     0.4136  18.650   <2e-16 ***
## ptratio      -1.2672     0.1342  -9.440   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.104 on 503 degrees of freedom
## Multiple R-squared:  0.5613, Adjusted R-squared:  0.5595
## F-statistic: 321.7 on 2 and 503 DF,  p-value: < 2.2e-16
```

Is there a relationship between the reponse and all predictors?

- $H_0 : \beta_1 = \cdots = \beta_p = 0$
- H_a: at least one of β_j is non-zero
- Did we miss β_0?
- This hypothesis test is performed by computing the F-statistic,

$$F = \frac{(TSS - RSS)/p}{RSS/(n-p-1)}$$

- The larger the F-statistic, the strong evidence against null hypothesis

ANOVA

```
anova(fit_rm_ptratio)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
	<int>	<dbl>	<dbl>	<dbl>	<dbl>
rm	1	20654.416	20654.4162	554.3367	3.508210e-83
ptratio	1	3320.252	3320.2525	89.1111	1.388009e-19
Residuals	503	18741.627	37.2597	*NA*	*NA*

3 rows

- The **degrees of freedom** (df) for RSS is $n - p - 1 = 503$.
- The **degrees of freedom** (df) for TSS is $1 + 1 + 503 = 505 = n - 1$.
- What's the sample size of this dataset?

ANOVA: reduce model vs. full model

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	505	42716.30	*NA*	*NA*	*NA*	*NA*
2	503	18741.63	2	23974.67	321.7239	1.039313e-90

2 rows

- Reduced model: `medv ~ 1` (no variable)
- Full model: `medv ~ rm + ptratio`

Confidence Intervals

```r
confint(fit_rm_ptratio, level = 0.95) # 95% CI for the coefficients
```

```
##                   2.5 %    97.5 %
## (Intercept) -10.791000  5.668670
## rm            6.901448  8.526692
## ptratio      -1.530892 -1.003431
```

Prediction using the linear regression model

```
predict(fit_rm_ptratio, newdata = data.frame(rm=1, ptratio=2))
```

```
##        1
## 2.618583
```

```
predict(fit_rm_ptratio, newdata = data.frame(rm=1, ptratio=2),
        level = 0.95, interval = "confidence")
```

```
##        fit       lwr      upr
## 1 2.618583 -4.509618 9.746783
```

```
predict(fit_rm_ptratio, newdata = data.frame(rm=1, ptratio=2),
        level = 0.95, interval = "prediction")
```

```
##        fit       lwr      upr
## 1 2.618583 -11.33255 16.56972
```

Train and testing in the linear regression model

We can split the data into training and testing.

```
boston_tr <- boston[1:400, ]
boston_te <- boston[-(1:400), ]
```

1. Fit the model on the training data.

```
fit_rm <- lm(medv ~ rm, data = boston_tr)
```

2. Compute the fitted value on the training data and compute the training error.

```
pred_rm_train <- predict(fit_rm, newdata = boston_tr)
train_error <- sum((pred_rm_train - boston_tr$medv)^2)
train_error
```

```
## [1] 14521.32
```

3. Compute the predicted value on the test data and compute the test error.

```
pred_rm_test <- predict(fit_rm, newdata = boston_te)
test_error <- sum((pred_rm_test - boston_te$medv)^2)
test_error
```

```
## [1] 8440.338
```

Qualitative Predictors

- Sometimes, **qualitative variables** (also called categorical variables) can be useful in making predictions.

- We usually code qualitative variables by **dummy variables** (variables taking values 0 and 1).

	medv	rm	rm_cate
	<dbl>	<dbl>	<fct>
1	24.0	6.575	Big
2	21.6	6.421	Medium
3	34.7	7.185	Big
4	33.4	6.998	Big
5	36.2	7.147	Big
6	28.7	6.430	Medium

6 rows

- Suppose you have a qualitative variable `rm_cate` with three : Small, Medium, Big. We need to code it with **two** dummy variables.

- Can we code the levels with one variable with values $1, 2$ and 3?

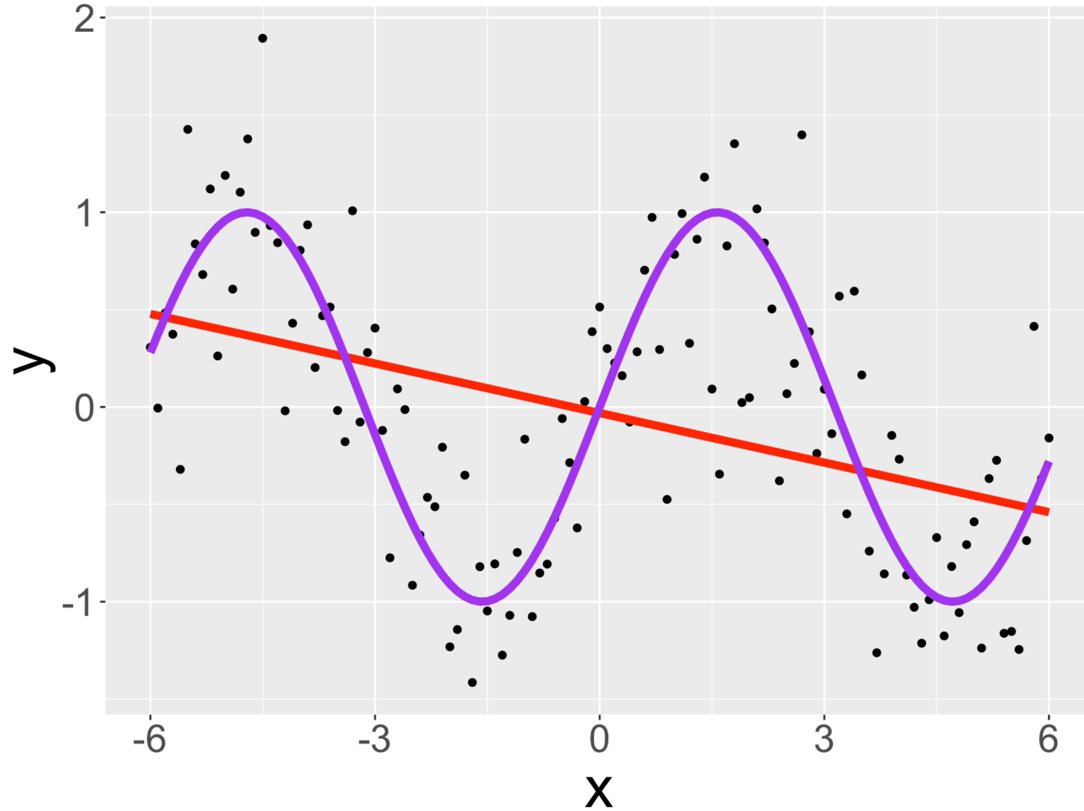
- Always recommended before applying any machine learning algorithms.

- R will automatically do the convertion for us in `lm()`

```
##
## Call:
## lm(formula = medv ~ rm_cate, data = boston)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.417  -4.379   0.466   3.267  32.521
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    17.4793     0.5766  30.315  < 2e-16 ***
## rm_cateMedium   2.7088     0.8166   3.317 0.000976 ***
## rm_cateBig     12.4379     0.8154  15.253  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.496 on 503 degrees of freedom
## Multiple R-squared:  0.3384, Adjusted R-squared:  0.3358
## F-statistic: 128.6 on 2 and 503 DF,  p-value: < 2.2e-16
```

K Nearest Neighbor

- Recall we assume $Y = f(X) + \epsilon$.



x	y	y_true
<dbl>	<dbl>	<dbl>
-6.0	0.306450529	0.27941550
-5.9	-0.005761892	0.37387666
-5.8	0.483365883	0.46460218
-5.7	0.373730825	0.55068554
-5.6	-0.320387037	0.63126664

-5.5	1.425693307	0.70554033
-5.4	0.837394632	0.77276449
-5.3	0.680184558	0.83226744
-5.2	1.119579350	0.88345466
-5.1	0.262042912	0.92581468

KNN

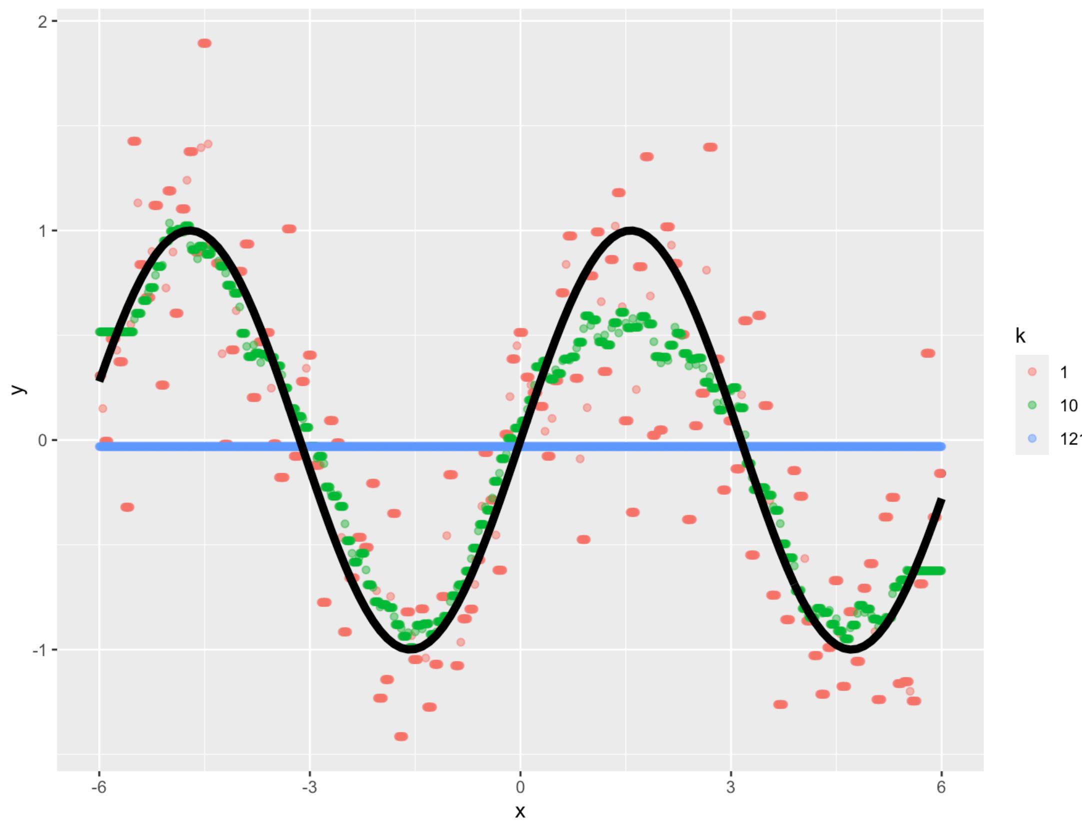
- Main Idea of KNN: given a new observation x_0, find the K **nearest** observations among $\{x_i, i = 1, \cdots, n\}$.
- What is near? Measure by the Euclidean Distance $\|x_i - x_0\|_2^2 = (x_i - x_0)^2$.
- Let's say the smallest are $x_{i_1}, \cdots, x_{i_K}$. Then, we have

$$\hat{y}_0 = \frac{1}{K}\sum_{j=1}^{K} y_{i_j}.$$

Different choice of K

k	train_error	test_error
<dbl>	<dbl>	<dbl>
1	0.00000	581.8208
10	22.81288	364.5425
121	68.58850	964.9834

3 rows



- $K = 1$: perfect fit on training data, too rough, bias low, variance high

- $K = 121$ (sample size): constant fit, just the sample average, too smooth, bias high, variance low (0)
- $K = 10$: a good balance