Prediction problems 1: Overview and Linear Regression

Data Science 101 Team

Agenda

- What is a prediction problem?
- Defining a model
- Fitting a model
- Validating your model (does it work?)

Examples Today

- Linear regression with a single variable
  - Example: predicting heights of children by height of parents
- Linear regression with multiple variables
  - Example: housing data from Boston

Prediction is a key task of data science

- You are given the heights of 1000 children. The task is to predict the height of a child that is randomly chosen from these 1000. The average height of the children is the "best" predictor when using Euclidean distance.
- Now for each child you are also given the average height of the parents. The goal is to predict the height of a child whose parents are on average e.g. 68 in tall. This additional information about the parents should allow us to make a better prediction. Regression does just that.

Applications of Regression

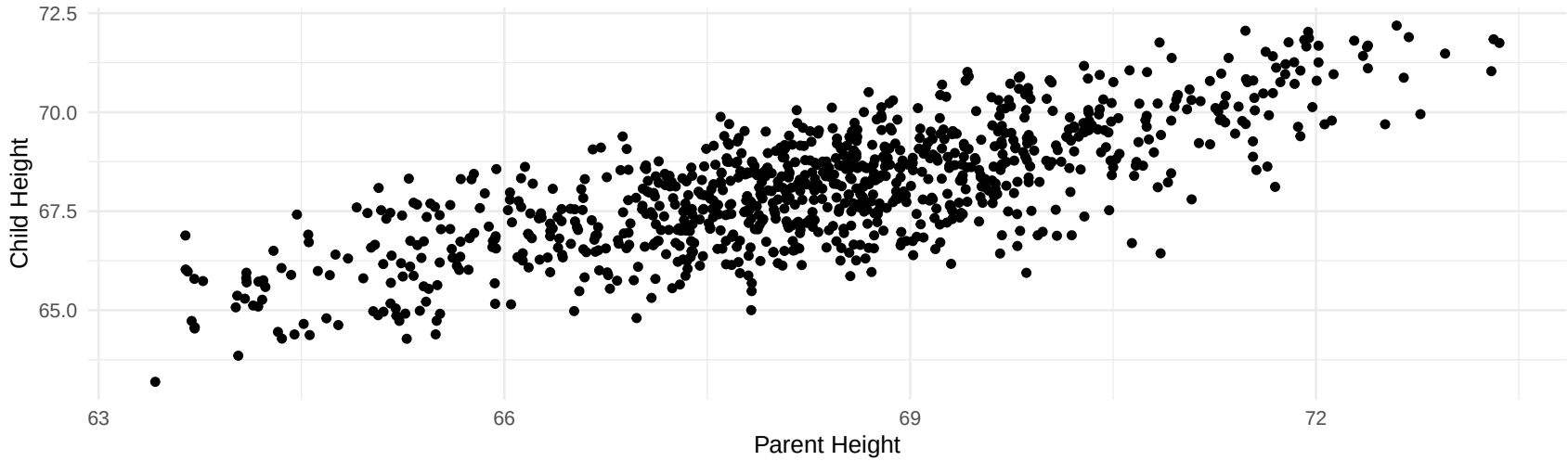
Basic idea: predict a variable y from observations of a (or multiple) variable(s) x

- Disease prediction (person has symptoms x, which disease (y) does he/she have?
- Car insurance: y = number of accidents, x = characteristics of driver (age, gender, tickets)
- Elections: y vote proportion for candidate A or B; x = previous voting results, results of polls, economic conditions, geographic information, demographics

Also known as **"supervised learning"** (in contrast to "unsupervised learning")

Linear regression with a single variable

- Goal: predict child height y from parent's height x
  - Call y the *dependent variable* or *response* or *outcome*
  - Call x the *independent variable* or *predictor*

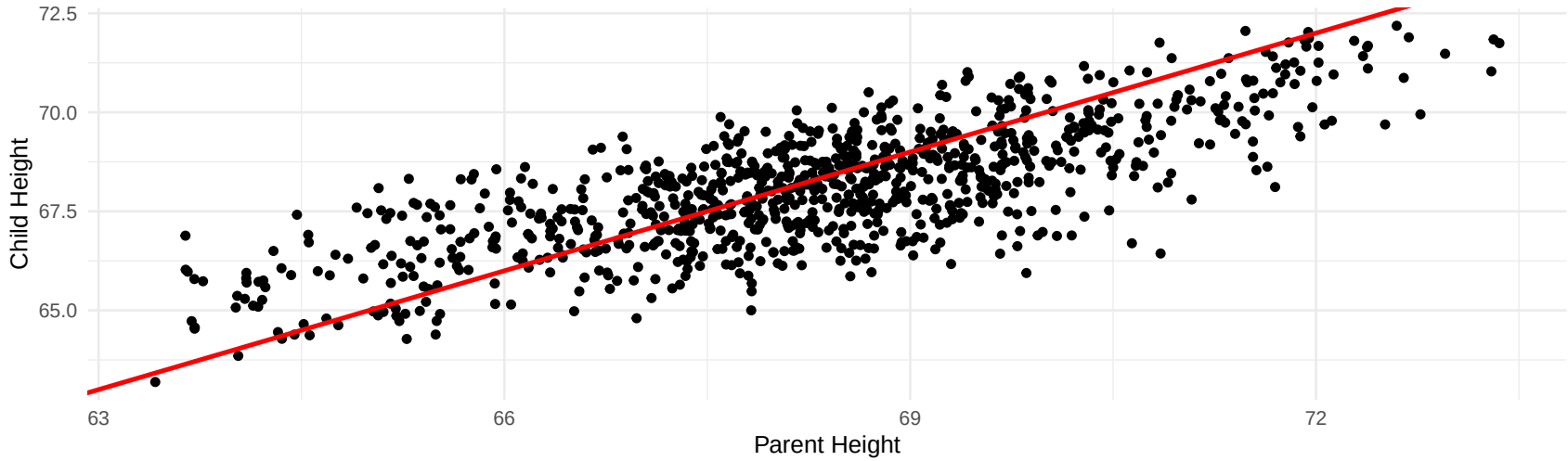



```
heights <- read.csv("data/galton.csv", row.names = 1)
g <- ggplot(heights, aes(parent, child)) + geom_point()
# g stores a ggplot that will be added to throughout
```

Predict child's height from parent's height

- First idea: predict

$$\widehat{y} = x$$

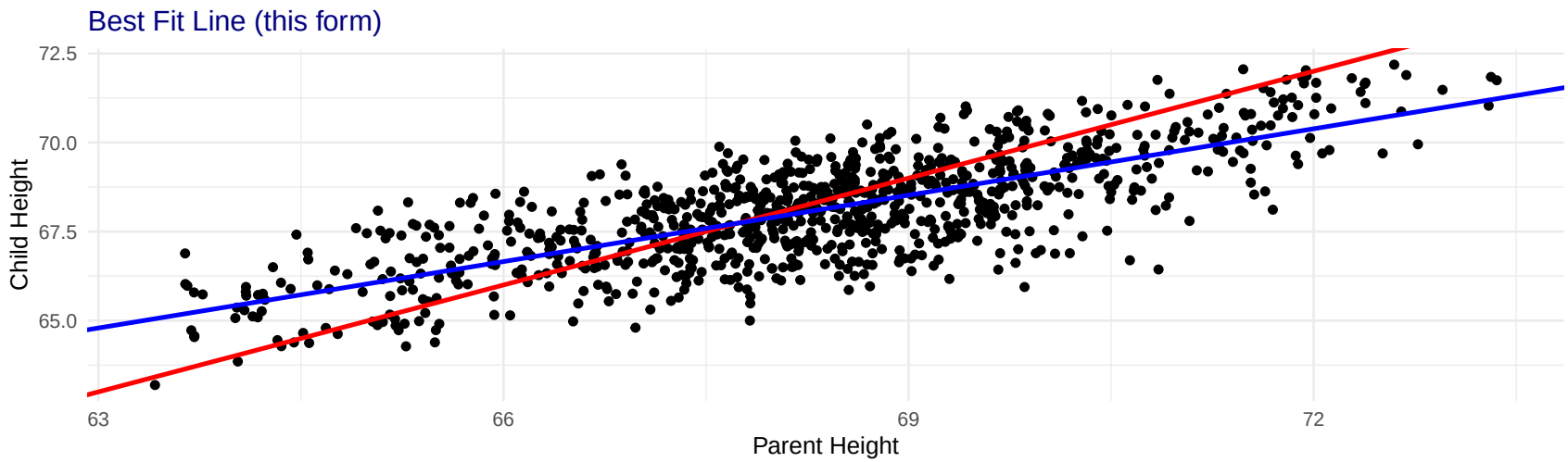



```
g + geom_abline(intercept = 0, slope = 1,
                colour = "red", size = 1)
```

Predict child's height from parent's height

- Expect a better prediction by allowing more flexibility:

$$\widehat{y} = \underbrace{\widehat{\beta}_0}_{\text{intercept}} + \underbrace{\widehat{\beta}_1}_{\text{slope}} x$$




```
linreg <- lm(child ~ parent, data = heights)
b <- coef(linreg)
g + geom_abline(intercept = b[1], slope = b[2],
                colour = "blue", size = 1)
```

The linear regression model

- Often one assumes a *linear* model, that is, that the *response* y satisfies

$$y = \beta_0 + \beta_1 x + \varepsilon$$

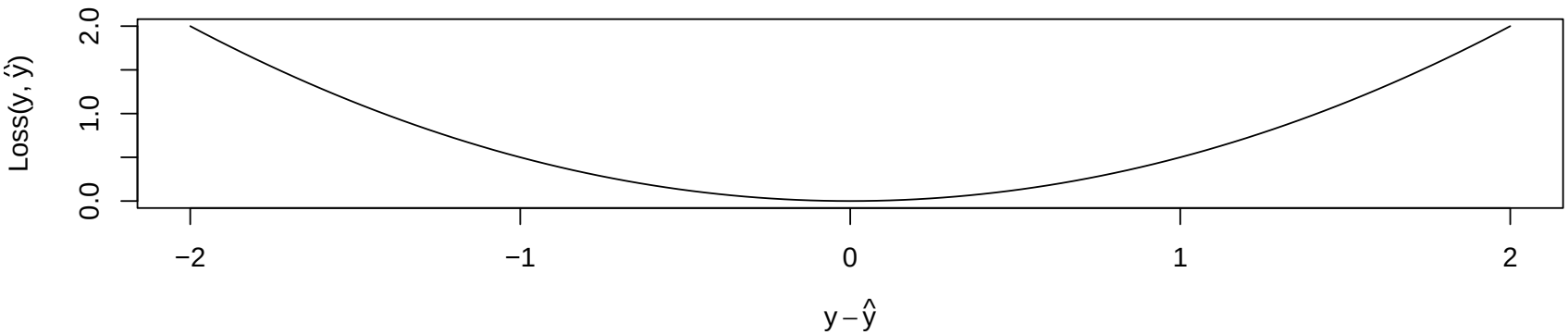
Terms above

- β_0 is the *intercept* term
- β_1 is the *slope* or *coefficient on* x
- ε is a noise term, which is usually assumed independent of x and mean zero

Fitting the regression model

- **Data**: Have n pairs (x_i, y_i), where x_i is the height of parent i and y_i is height of child i
- **Predictions**: $\hat{y}_i$ is our model's prediction of child height i
- **Loss function**: First, define a way to measure *error* or *residual* $\hat{y}_i - y_i$

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



Fitting the regression model

- **Data**: Have n pairs (x_i, y_i), where x_i is the height of parent i and y_i is height of child i
- **Predictions**: $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ is the regression line, the prediction of the model for y_i, the height of child i
- **Loss on data**: Determine β_0 and β_1 with the *method of least squares*, i.e. to solve least squares problem

$$\underset{\beta_0,\beta_1}{\text{minimize}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} (y_i - \beta_0 - x_i\beta_1)^2$$

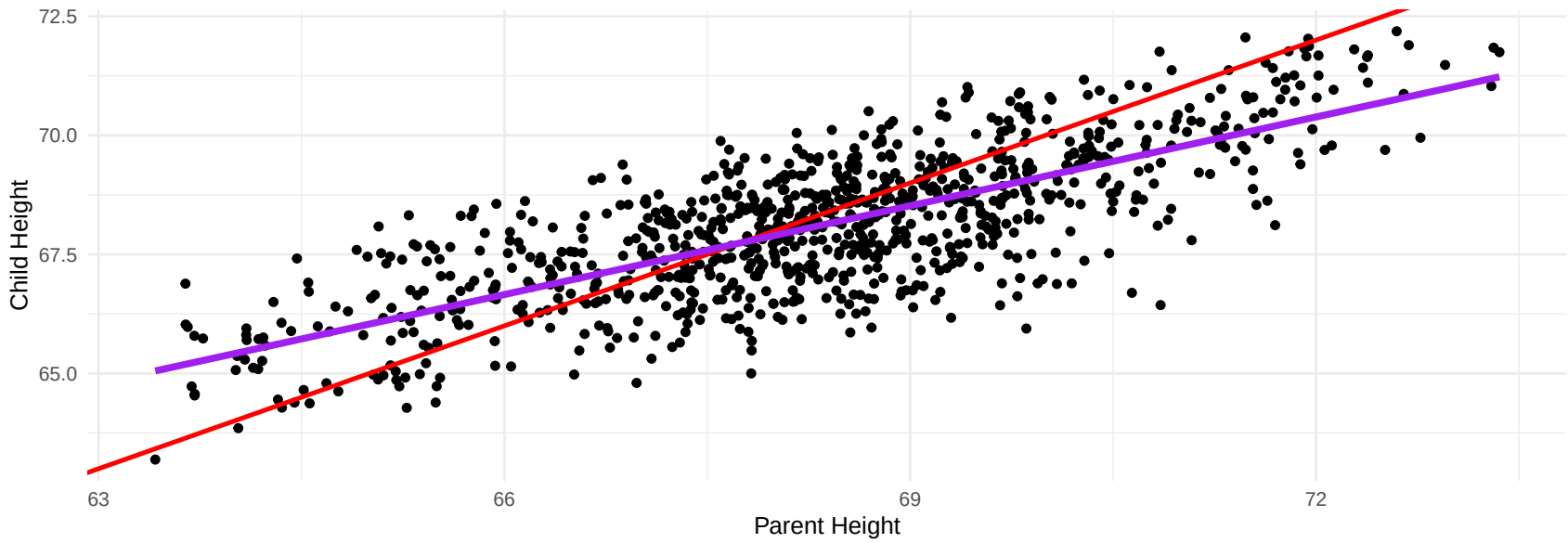
```
linreg <- lm(y ~ x)
```

- `lm` is the code for "linear model", while the ~ symbol means to regress `y` on `x`

Example: height data

```
linreg <- lm(child ~ parent, data = heights)
```

Now that we have performed the regression, we can plot the result. The values that our linear regression predicts are in `linreg$fitted.values`



Residuals of best fit

Compare squared residuals $(y - \hat{y})^2$, to those obtained by predicting with parent's height only.

```
MSE.naive <- mean((heights$child - heights$parent)^2)
MSE.fit <- mean(linreg$residuals^2)
```

Mean squared residuals for *naive* prediction: **1.538**
Mean squared residuals for *fit* prediction: **0.999**

Regression and the correlation coefficient

- It can be shown that the estimate of the intercept is $\hat{\beta}_0 = \overline{y} - \hat{\beta}_1 \cdot \overline{x}$
- It can be shown that the regression line $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$ resulting from the method of least squares results in:

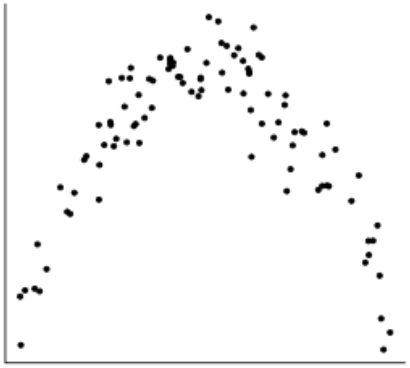
$$\hat{\beta}_1 = r \cdot \frac{\text{SD of y}}{\text{SD of x}}$$

where r is the correlation coefficient

$$r = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{\text{SD of x}} \cdot \frac{y_i - \bar{y}}{\text{SD of y}} \right)$$

Notes on the correlation coefficient

- r is always between -1 and 1. It is a measure of linear association between x and y, with $|r| = 1$ in case of perfect linear association where all the points fall on a line
- Keep in mind that r is only useful for measuring association if the scatter roughly follows a line. The following scatter plot shows a strong nonlinear association, but $r = 0$:



Notes on the regression line

- Remember that correlation does not mean causation: For example, among school children there is a high correlation between shoe size and reading ability. Both are driven e.g. by the lurking variable "age"
- It turns out that the regression line of y on x goes through the point of averages $(\overline{x}, \overline{y})$
- But $\hat{\beta}_1 = r \cdot \frac{\text{SD of y}}{\text{SD of x}}$ means that if x is one SD above $\bar{x}$, then the predicted $\hat{y}$ is only $r \cdot \text{SD}$ above $\bar{y}$.
- Since r is between -1 and 1, the prediction is "towards the mean": $\hat{y}$ is fewer SDs away from $\bar{y}$ than x is from $\bar{x}$. This is called "regression to the mean" or "regression effect"

Another view: Regression to the mean

$$\begin{aligned}\hat{y} &= \hat{\beta}_0 \quad + \quad \hat{\beta}_1 \cdot x \\ &= \hat{\beta}_0 + \hat{\beta}_1 \cdot \overline{x} + \hat{\beta}_1 \cdot (x - \overline{x})\end{aligned}$$

where $\overline{x}$ is the mean of all parent heights

Meaning of these?

Suppose the slope $\hat{\beta}_1 < 1$

- $\hat{\beta}_0 + \hat{\beta}_1 \cdot \overline{x}$ is average child height
- $\hat{\beta}_1 \cdot (x - \overline{x})$ is *regression to mean*, so the children of taller parents shrink towards average

Regression to the mean: Intuition

- For example, in a test-retest situation, the top group on the test will drop down somewhat on the retest, while the bottom group moves up
- A heuristic explanation: To score among the very top in the first test requires excellent preparation as well as some luck. This luck might not be there any more on the retest, and so we expect this top group to fall back a bit
- This effect is simply a consequence of there being a scatter around the line. Erroneously assuming that this occurs due to some action (e.g. "the top scorers on the first test slackened off") is called *regression fallacy*

More than one covariate

Instead of $x \in \mathbb{R}$, input vectors $x \in \mathbb{R}^p$

- p is number of parameters (sometimes d for *dimension* or k in econometrics)
- Call x the *covariates*, *features*, *independent variables*, *inputs*, *exogenous variables*, *explanatory variables*, or *predictor variables*
  - Note that x variables are usually NOT independent of one another

Example: Boston housing data

- Target variable y is median home price in a small area of Boston
- Input variables include
  - Crime rate per capita
  - Average number of bedrooms
  - Proportion of large lots (zoned for $>$ 25,000 feet)
  - Whether a home is near the Charles river ($x \in \{0, 1\}$)

Example: Housing data from Boston

```
library(MASS) # contains Boston data
boston <- filter(Boston, medv != 50) # Remove "$500,000 or more"
glimpse(boston)              # as it is censored data
```

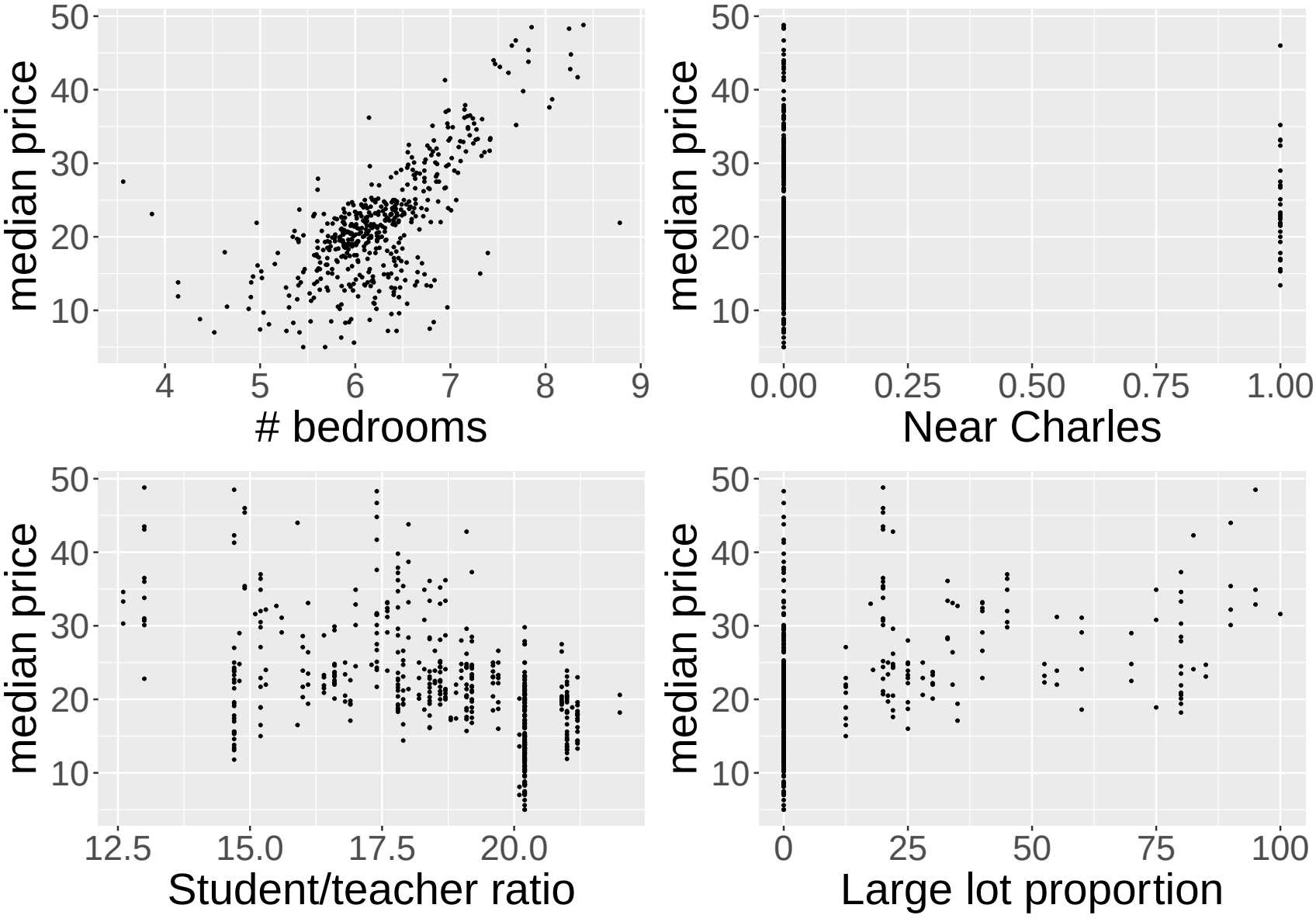
```
Rows: 490
Columns: 14
$ crim    <dbl> 0.00632, 0.02731, 0.02729, 0.03237, 0.06905, 0.02985, 0.08829,~
$ zn      <dbl> 18.0, 0.0, 0.0, 0.0, 0.0, 0.0, 12.5, 12.5, 12.5, 12.5, 12.5, 1~
$ indus   <dbl> 2.31, 7.07, 7.07, 2.18, 2.18, 2.18, 7.87, 7.87, 7.87, 7.87, 7.~
$ chas    <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ nox     <dbl> 0.538, 0.469, 0.469, 0.458, 0.458, 0.458, 0.524, 0.524, 0.524,~
$ rm      <dbl> 6.575, 6.421, 7.185, 6.998, 7.147, 6.430, 6.012, 6.172, 5.631,~
$ age     <dbl> 65.2, 78.9, 61.1, 45.8, 54.2, 58.7, 66.6, 96.1, 100.0, 85.9, 9~
$ dis     <dbl> 4.0900, 4.9671, 4.9671, 6.0622, 6.0622, 6.0622, 5.5605, 5.9505~
$ rad     <int> 1, 2, 2, 3, 3, 3, 5, 5, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 4, 4,~
$ tax     <dbl> 296, 242, 242, 222, 222, 222, 311, 311, 311, 311, 311, 311, 31~
$ ptratio <dbl> 15.3, 17.8, 17.8, 18.7, 18.7, 18.7, 15.2, 15.2, 15.2, 15.2, 15~
$ black   <dbl> 396.90, 396.90, 392.83, 394.63, 396.90, 394.12, 395.60, 396.90~
$ lstat   <dbl> 4.98, 9.14, 4.03, 2.94, 5.33, 5.21, 12.43, 19.15, 29.93, 17.10~
$ medv    <dbl> 24.0, 21.6, 34.7, 33.4, 36.2, 28.7, 22.9, 27.1, 16.5, 18.9, 15~
```

Small multiples

- use the ggpubr package

```
pacman::p_load(ggpubr)
p1 <- ggplot(boston,aes(x=rm,y=medv)) + geom_point(cex = 0.5) +
  xlab('# bedrooms') + ylab('median price') +
  theme(text=element_text(size=24))

p2 <- ggplot(boston,aes(x=chas,y=medv)) + geom_point(cex = 0.5) +
  xlab('Near Charles') + ylab('median price') +
  theme(text=element_text(size=24))

p3 <- ggplot(boston,aes(x=ptratio,y=medv)) + geom_point(cex = 0.5) +
  xlab('Student/teacher ratio') + ylab('median price') +
  theme(text=element_text(size=24))

p4 <- ggplot(boston,aes(x=zn,y=medv)) + geom_point(cex = 0.5) +
  xlab('Large lot proportion') + ylab('median price') +
  theme(text=element_text(size=24))

plt <- ggarrange(p1,p2,p3,p4,ncol=2,nrow=2)
```

Small multiples (ggplot output)



Example (continued): Boston Housing

Idea: let us run a regression on each of these, and see which is best

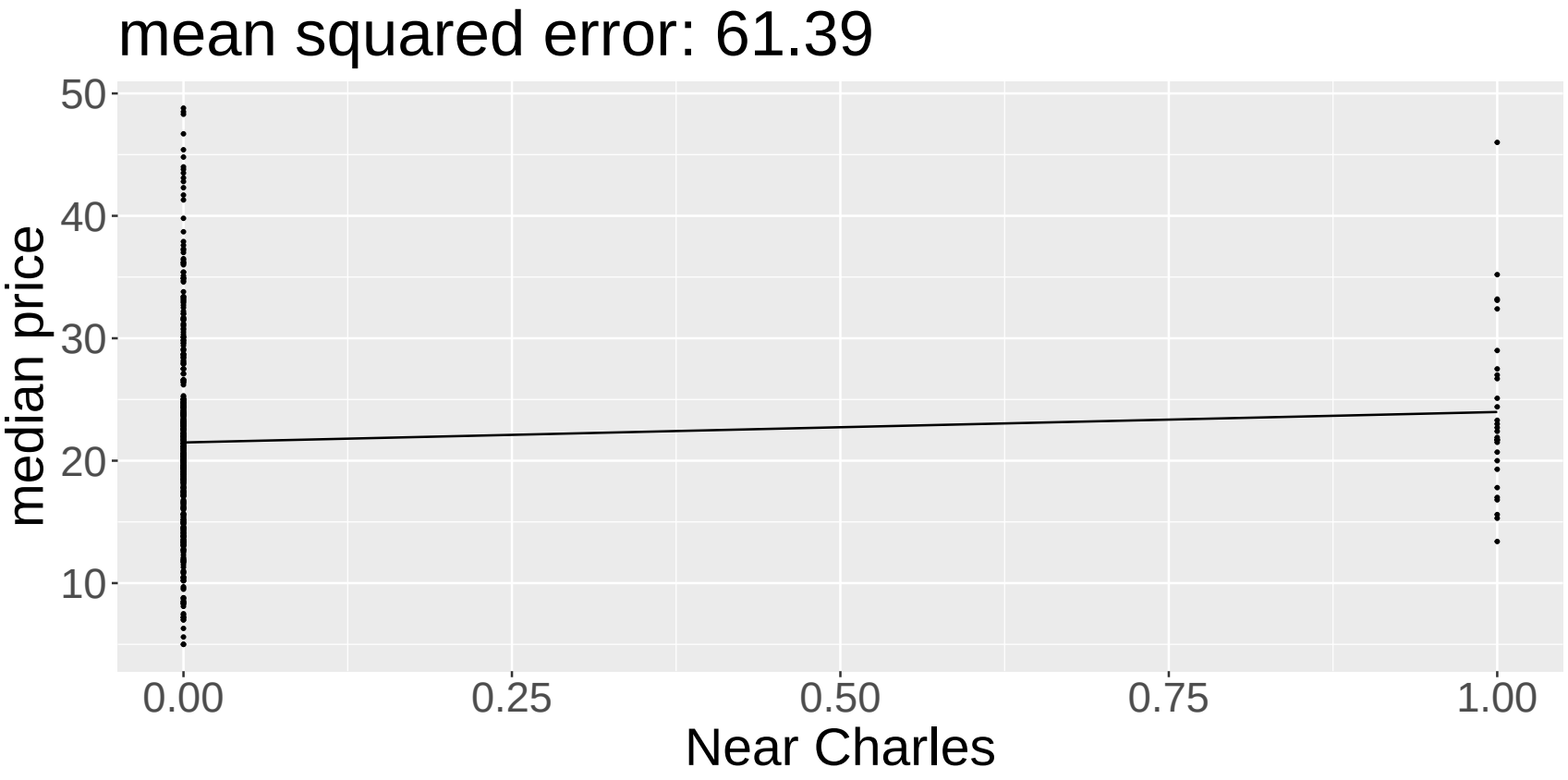
- To run a regression on a dataframe, we tell R to use a formula with the columns we care about and the dataframe we want
- For example, to predict median value `boston$medv` from number of bedrooms `boston$rm`, use

```
lm(formula = medv ~ rm, data = boston)
```

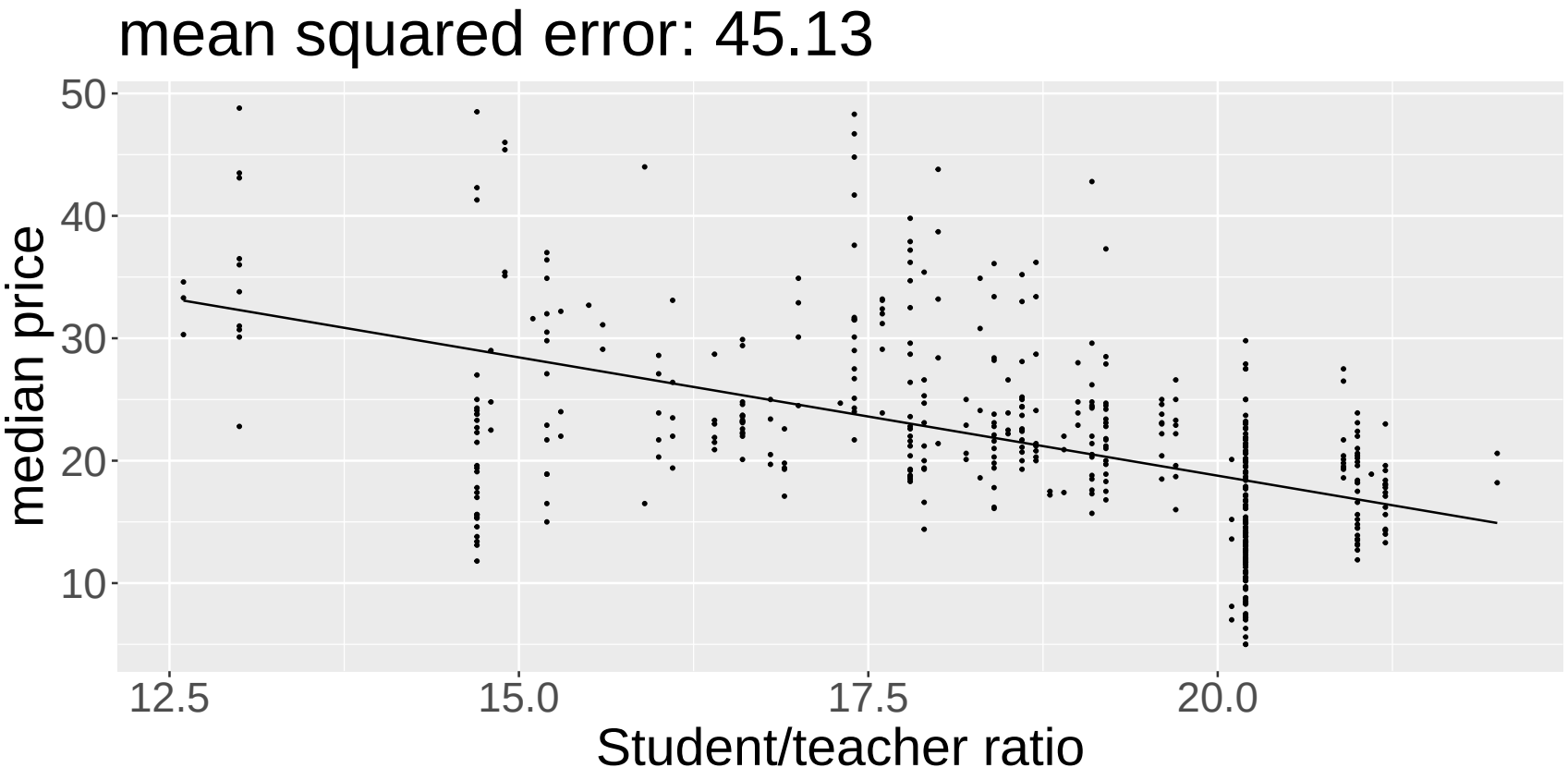
```
Call:
lm(formula = medv ~ rm, data = boston)

Coefficients:
(Intercept)           rm
    -30.005        8.269
```

Boston housing: rooms

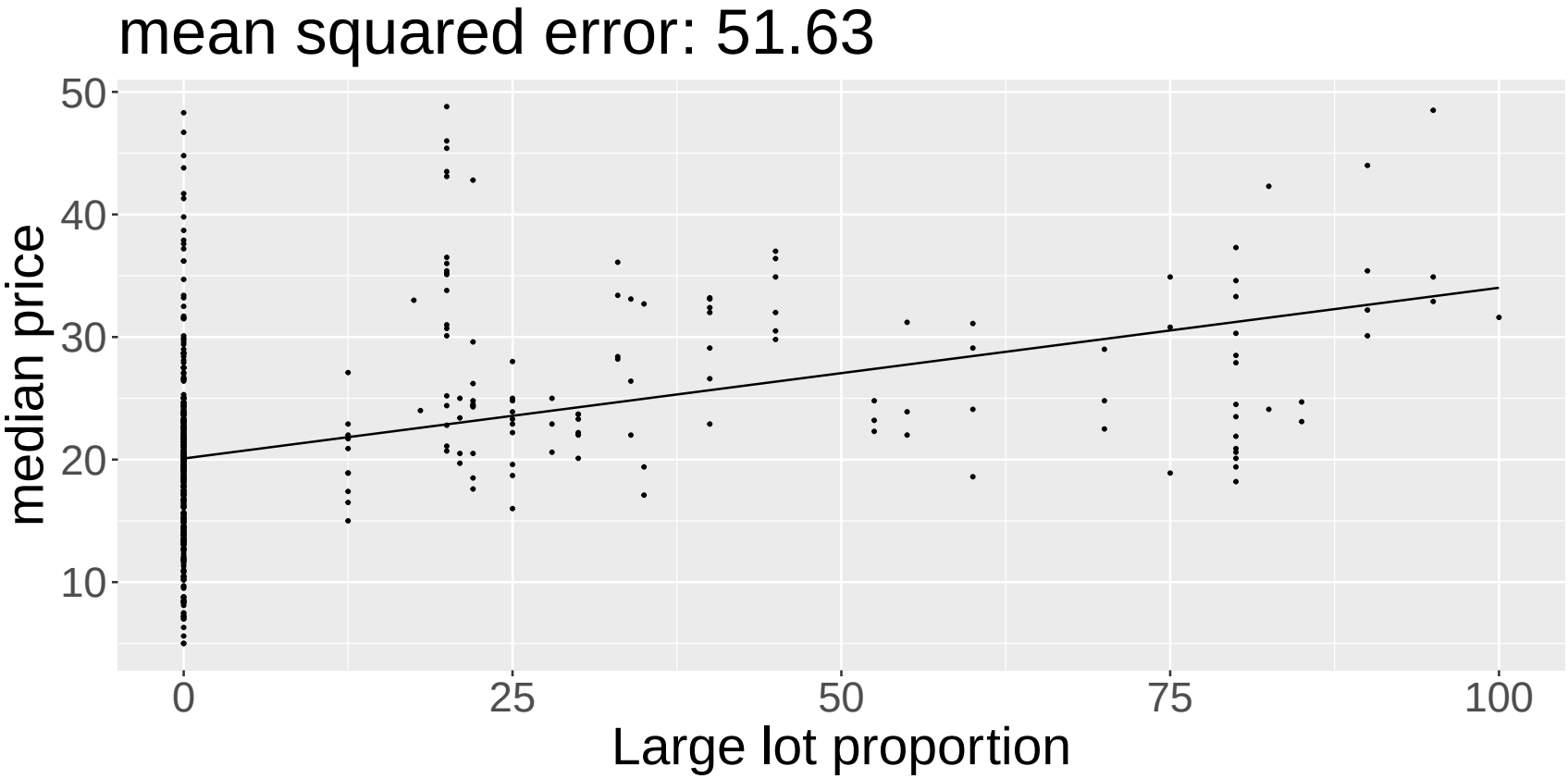
Boston housing: Near Charles River? (A binary variable)



Boston housing: Student-teacher ratio



Boston housing: Big Residential Properties



Dot Product for vectors x and y

For vectors x and v, the dot product is:

$$x \cdot v = x^T v = \sum_{j=1}^{p} x_j v_j$$

```
x <- 1:10
v <- 11:20
sum(x * v)
```

```
[1] 935
```

```
x %*% v
```

```
     [,1]
[1,]  935
```

$x \cdot x$ is often called the "sum of squares."

```
sum(x^2)
```

```
[1] 385
```

```
x %*% x
```

```
     [,1]
[1,]  385
```

Combining input variables

- Make predictions

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

- Model the data as

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

  where ε is noise
- Often, choose noise distribution based on
  - Convenience (assume it is normally distributed)
  - Prior knowledge (rarely)

The linear model is simple but effective

- Despite its simplicity (it's a linear function!), the predictions from this model are often quite good even if the underlying model doesn't apply
- This is because simplicity is helpful for prediction ("Occam's Razor", "Parsimony", see later):

All models are wrong; some models are useful . . . Just as the ability to devise simple but evocative models is the signature of the great scientist so overelaboration and overparameterization is often the mark of mediocrity - George Box

- Also, if the relationship between x and y is not linear, then it may still be possible to obtain a roughly linear relationship between transformations of the x and the y

Fitting a multiple regression

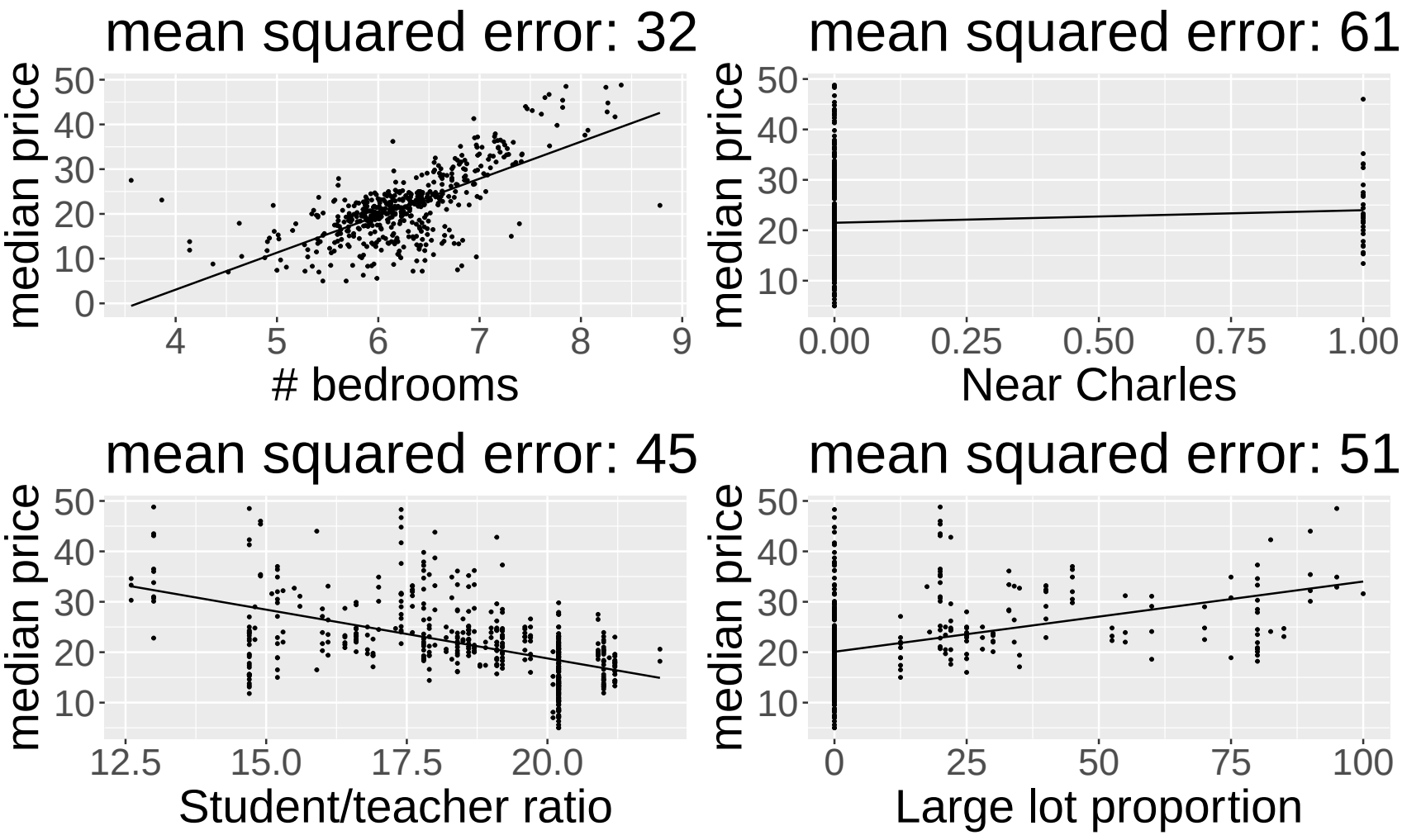
- Making predictions using

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

- Fit as in single variable case. Solve

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

Housing data one at a time (previous four plots)



Housing data regression on two variables

To perform linear regression on multiple variables, in the `lm` command, simply "add" them in the formula

```
lm.both <- lm(formula = medv ~ rm + ptratio,
              data = boston)
```

Mean squared error (just rooms): 32.63
Mean squared error (just student/teacher): 45
Mean squared error (both): *25.83*

Why does MSE go down? Will it always go down?

Housing data: regression on everything

To use all independent variables (except the target y in the regression), use the notation

`lm(y ~ ., data = your.dataset)`

For the Boston housing data,

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

Mean squared error for all: 13.727

Why can't we use the individual regression estimates $\hat{\beta}_j$ for the multiple regression on all predictors?

Why not?

Predictions vs. Actual (using all x variables)

Code for Preceding Plot

```
N <- length(full_reg$fitted.values)

forecasts <- data.frame(medv = rep(boston$medv, 2),
                        fitted = c(full_reg$fitted.values,
                                   lm.rm$fitted.values),
                        model = c(rep("Full Model", N),
                                  rep("Rooms Only", N)))

f <- ggplot(data = forecasts,
            aes(fitted, medv,
                color = model)) + geom_point() +
  xlim(c(0,50)) + ylim(c(0,50))

f + labs(x = "Predicted Price", y = "Actual Price")
```

Mean squared error on the (training) dataset does not tell the whole story

- It tells us how well the model performs on the SAME data that was use to fit it
- By **overfitting** the model (e.g., using lots of predictors), we can make the mean squared error arbitrarily low
- But this model will not accurately predict future datasets
- A good strategy: Divide your dataset at random into say two parts - a training (fitting) part and a validation part
- Fit your model using the training set, and record the mean squared error on the validation part. Overfitting exhibits itself as high mean squared error on the validation set

Summary

- Linear regression
  - Predicted real-valued targets using *linear* predictions
  - Learned to fit model using R
- Things we have *not* covered (but will explore)
  - Choosing variables (variable selection)
  - Encoding variables (is linear form correct?)
  - Interactions between variables