Correlation and Regression

Seth Harrison

Last compiled: Feb 12, 2021

Contents

1 Prerequisites **5**

2 Visualizing two variables **7**

2.1 Scatterplots 7
2.2 Boxplots as discretized/conditioned scatterplots 8
2.3 Creating scatterplots 9
2.4 Transformations 16
2.5 Identifying outliers 18

3 Correlation **21**

3.1 Computing correlation 22
3.2 Exploring Anscombe 23
3.3 Spurious correlation in random data 29

4 Simple linear regression **33**

4.1 The “best fit” line 33
4.2 Uniqueness of least squares regression line 34
4.3 Fitting a linear model “by hand” 36
4.4 Regression to the mean 37

5 Interpreting regression models **41**

5.1 Fitting simple linear models 42
5.2 The lm summary output 44
5.3 Fitted values and residuals 45
5.4 Tidying your linear model 46
5.5 Making predictions 47
5.6 Adding a regression line to a plot manually 48

6 Model Fit **51**

6.1 Standard error of residuals 51
6.2 Assessing simple linear model fit 53
6.3 Linear vs. average 55
6.4 Leverage 56
6.5 Influence 57

6.6 Removing outliers 58
6.7 High leverage points 60

Chapter 1

Prerequisites

This material is from the DataCamp course Correlation and Regression by Ben Baumer. Before using this material, the reader should have completed and be comfortable with the material in the DataCamp modules Introduction to R.

Reminder to self: each `*.Rmd` file contains one and only one chapter, and a chapter is defined by the first-level heading `#`.

Chapter 2

Visualizing two variables

In this chapter, you will learn techniques for exploring bivariate relationships.

2.1 Scatterplots

Scatterplots are the most common and effective tools for visualizing the relationship between two numeric variables.

The `ncbirths` dataset is a random sample of 1,000 cases taken from a larger dataset collected in 2004. Each case describes the birth of a single child born in North Carolina, along with various characteristics of the child (e.g. birth weight, length of gestation, etc.), the child's mother (e.g. age, weight gained during pregnancy, smoking habits, etc.) and the child's father (e.g. age). You can view the help file for these data by running `?ncbirths` in the console.

```r
library(openintro)
DT::datatable(ncbirths)
```

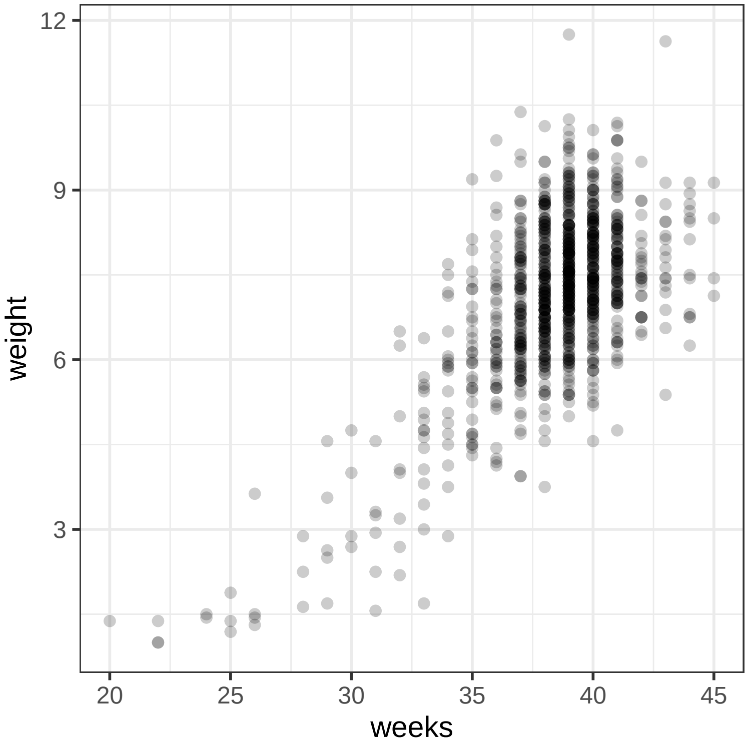
```
PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pl
```

Exercise

Using the `ncbirths` dataset, make a scatterplot using `ggplot()` to illustrate how the birth weight of these babies varies according to the number of weeks of gestation.

```
# Scatterplot of weight vs. weeks
library(ggplot2)
ggplot(data = ncbirths, aes(y = weight, x = weeks)) +
  geom_point(alpha = 0.2) +
  theme_bw()
```

```
Warning: Removed 2 rows containing missing values (geom_point).
```



2.2 Boxplots as discretized/conditioned scatterplots

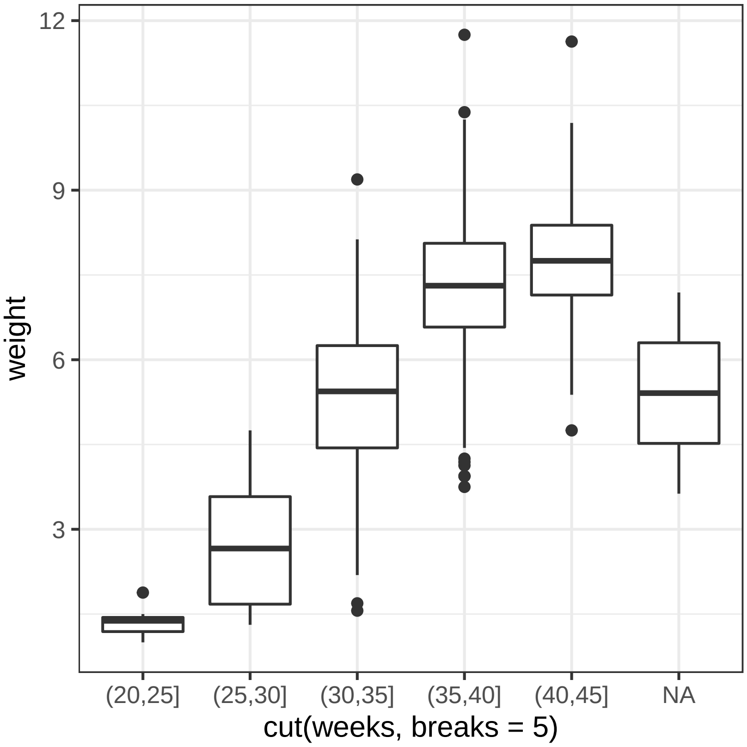
If it is helpful, you can think of boxplots as scatterplots for which the variable on the x-axis has been discretized.

The `cut()` function takes two arguments: the continuous variable you want to discretize and the number of `breaks` that you want to make in that continuous variable in order to discretize it.

Exercise

Using the ncbirths dataset again, make a boxplot illustrating how the birth weight of these babies varies according to the number of weeks of gestation. This time, use the cut() function to discretize the x-variable into six intervals (i.e. five breaks).

```
# Boxplot of weight vs. weeks
ggplot(data = ncbirths,
       aes(x = cut(weeks, breaks = 5), y = weight)) +
       geom_boxplot() +
       theme_bw()
```



2.3 Creating scatterplots

Creating scatterplots is simple and they are so useful that is it worthwhile to expose yourself to many examples. Over time, you will gain familiarity with the types of patterns that you see. You will begin to recognize how scatterplots can reveal the nature of the relationship between two variables.

In this exercise, and throughout this chapter, we will be using several datasets listed below. These data are available through the `openintro` package. Briefly:

- The `mammals` dataset contains information about 39 different species of mammals, including their body weight, brain weight, gestation time, and a few other variables.

- The `mlbbat10` dataset contains batting statistics for 1,199 Major League Baseball players during the 2010 season.

- The `bdims` dataset contains body girth and skeletal diameter measurements for 507 physically active individuals.

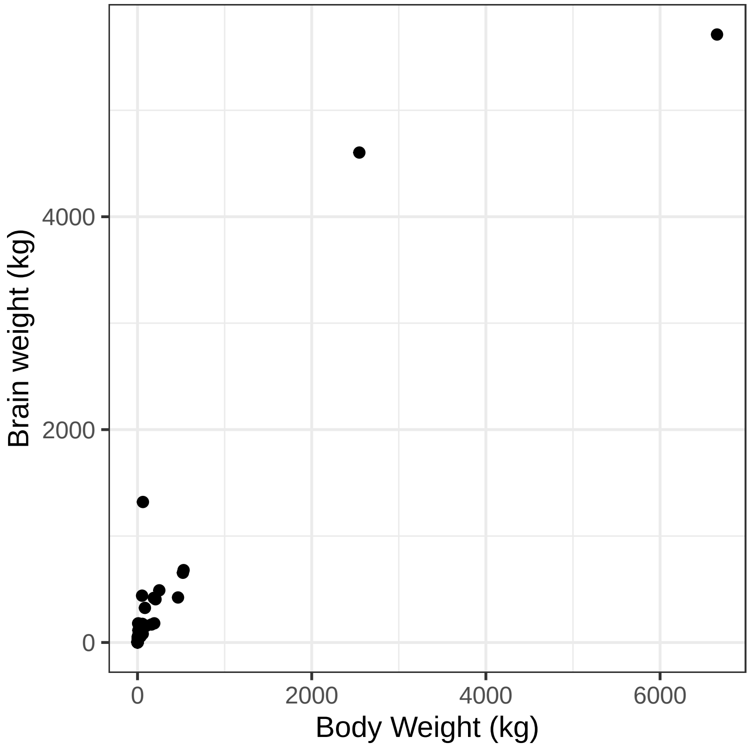
- The `smoking` dataset contains information on the smoking habits of 1,691 citizens of the United Kingdom.

To see more thorough documentation, use the `?` or `help()` functions.

Exercise

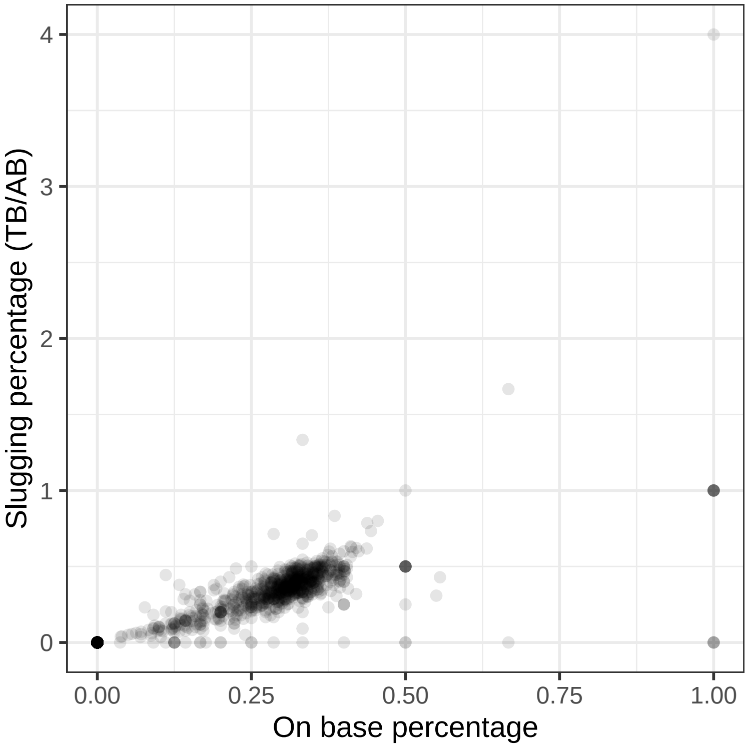
- Using the `mammals` dataset, create a scatterplot illustrating how the brain weight of a mammal varies as a function of its body weight.

```r
library(openintro)
# Mammals scatterplot
ggplot(data = mammals, aes(y = brain_wt, x = body_wt)) +
  geom_point() +
  theme_bw() +
  labs(x = "Body Weight (kg)", y = "Brain weight (kg)")
```



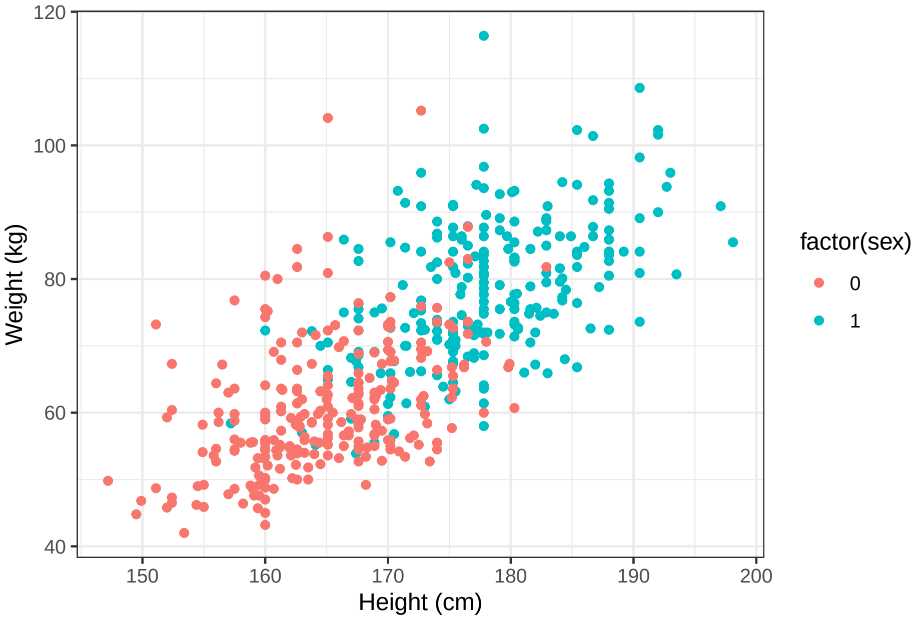
- Using the `mlbbat10` dataset, create a scatterplot illustrating how the slugging percentage (slg) of a player varies as a function of his on-base percentage (obp).

```
# Baseball player scatterplot
p1 <- ggplot(data = mlbbat10, aes(y = slg, x = obp)) +
  geom_point(alpha = 0.10) +
  theme_bw() +
  labs(y = "Slugging percentage (TB/AB)", x = "On base percentage" )
p1
```



- Using the `bdims` dataset, create a scatterplot illustrating how a person's weight varies as a function of their height. Use color to separate by sex, which you'll need to coerce to a factor with `factor()`.

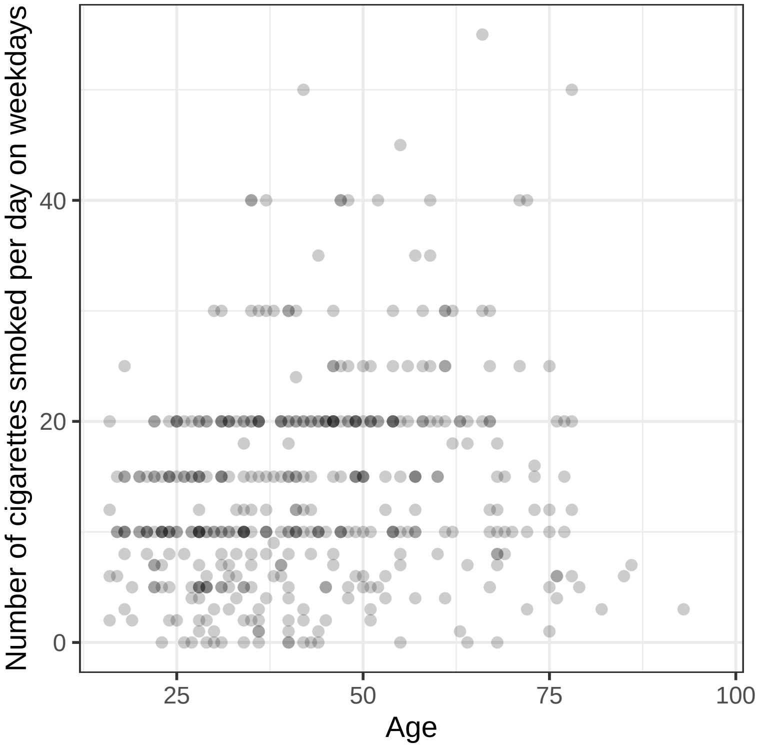
```
# Body dimensions scatterplot
p3 <- ggplot(data = bdims, aes(y = wgt, x= hgt, color = factor(sex))) +
  geom_point() +
  theme_bw() +
  labs(y = "Weight (kg)", x = "Height (cm)")
p3
```



- Using the smoking dataset, create a scatterplot illustrating how the amount that a person smokes on weekdays varies as a function of their age.

```
# Smoking scatterplot
ggplot(data = smoking, aes(y = amt_weekdays, x = age)) +
  geom_point(alpha = 0.2) +
  theme_bw() +
  labs(x = "Age", y = "Number of cigarettes smoked per day on weekdays")
```

```
Warning: Removed 1270 rows containing missing values (geom_point).
```



Characterizing scatterplots

Figure 2.1 shows the relationship between the poverty rates and high school graduation rates of counties in the United States.

```
-- Column specification --------------------------------------------------------
cols(
  .default = col_double()
)
i Use `spec()` for the full column specifications.
```

Describe the form, direction, and strength of this relationship.

- Linear, positive, strong
- Linear, negative, weak
- **Linear, negative, moderately strong**
- Non-linear, negative, strong

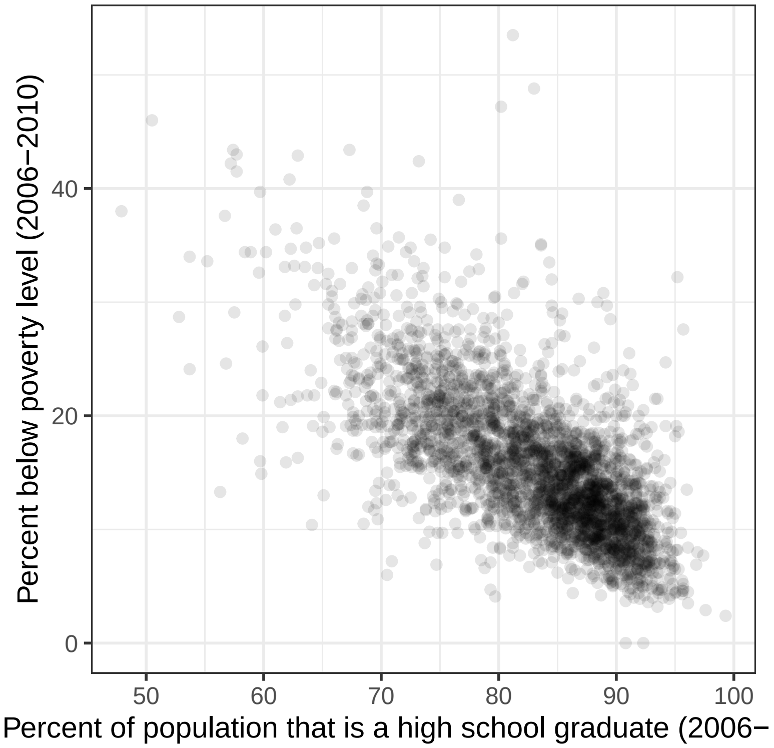


Figure 2.1: Poverty versus high school graduation rate

2.4 Transformations

The relationship between two variables may not be linear. In these cases we can sometimes see strange and even inscrutable patterns in a scatterplot of the data. Sometimes there really is no meaningful relationship between the two variables. Other times, a careful transformation of one or both of the variables can reveal a clear relationship.

Recall the bizarre pattern that you saw in the scatterplot between brain weight and body weight among mammals in a previous exercise. Can we use transformations to clarify this relationship?

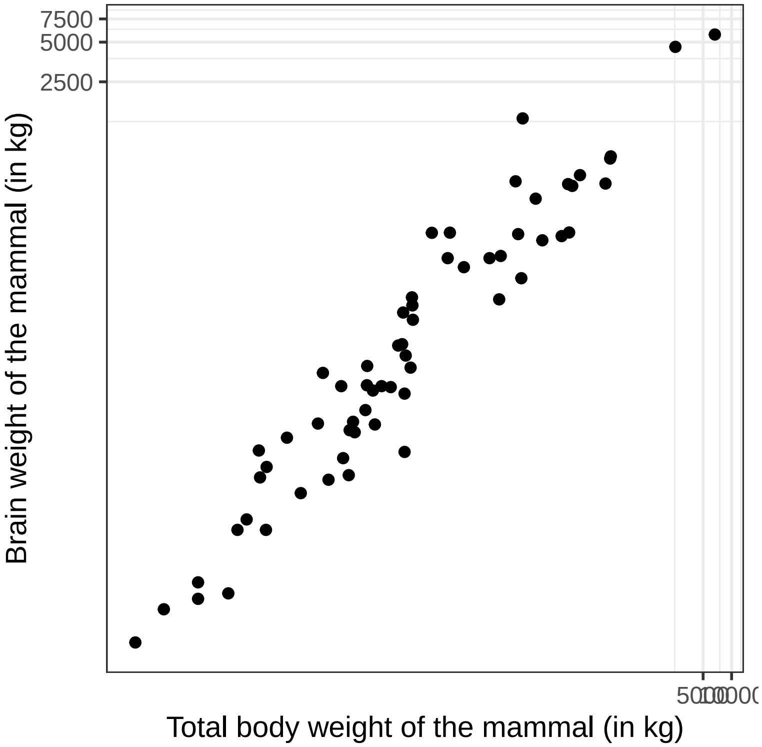
`ggplot2` provides several different mechanisms for viewing transformed relationships. The `coord_trans()` function transforms the coordinates of the plot. Alternatively, the `scale_x_log10()` and `scale_y_log10()` functions perform a base-10 log transformation of each axis. Note the differences in the appearance of the axes.

Exercise

The `mammals` dataset is available in your workspace.

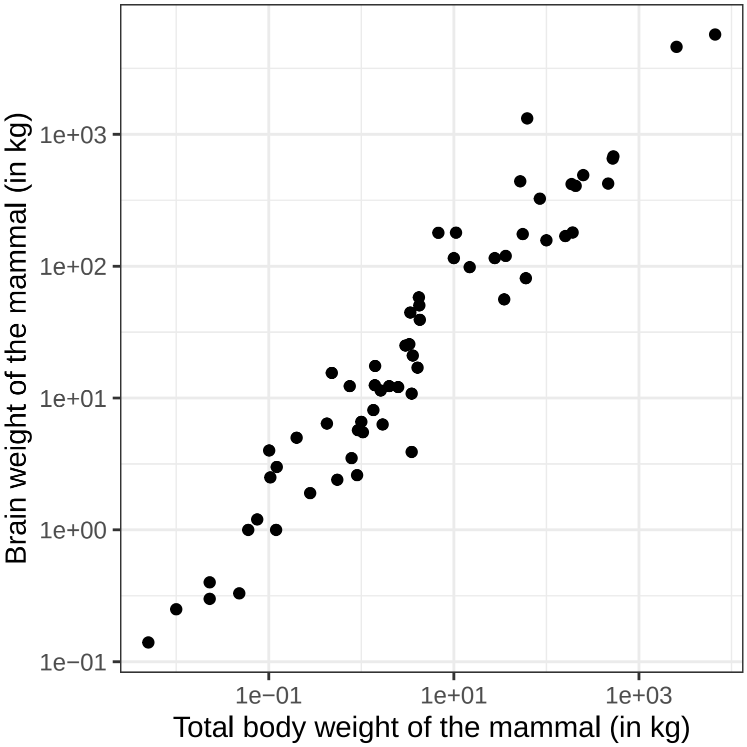
- Use `coord_trans()` to create a scatterplot showing how a mammal's brain weight varies as a function of its body weight, where both the x and y axes are on a `"log10"` scale.

```r
# Scatterplot with coord_trans()
ggplot(data = mammals, aes(y = brain_wt, x = body_wt)) +
  geom_point() +
  coord_trans(x = "log10", y = "log10") +
  theme_bw() +
  labs(x = "Total body weight of the mammal (in kg)",
       y = "Brain weight of the mammal (in kg)")
```



- Use `scale_x_log10()` and `scale_y_log10()` to achieve the same effect but with different axis labels and grid lines.

```
# Scatterplot with scale_x_log10() and scale_y_log10()
p4 <- ggplot(data = mammals, aes(x = body_wt, y = brain_wt)) +
  geom_point() +
  scale_x_log10() +
  scale_y_log10() +
  theme_bw() +
  labs(x = "Total body weight of the mammal (in kg)",
       y = "Brain weight of the mammal (in kg)")
p4
```



2.5 Identifying outliers

In Chapter ??, we will discuss how outliers can affect the results of a linear regression model and how we can deal with them. For now, it is enough to simply identify them and note how the relationship between two variables may change as a result of removing outliers.

Recall that in the baseball example earlier in the chapter, most of the points were clustered in the lower left corner of the plot, making it difficult to see the general pattern of the majority of the data. This difficulty was caused by a few outlying players whose on-base percentages (OBPs) were exceptionally high. These values are present in our dataset only because these players had very few batting opportunities.

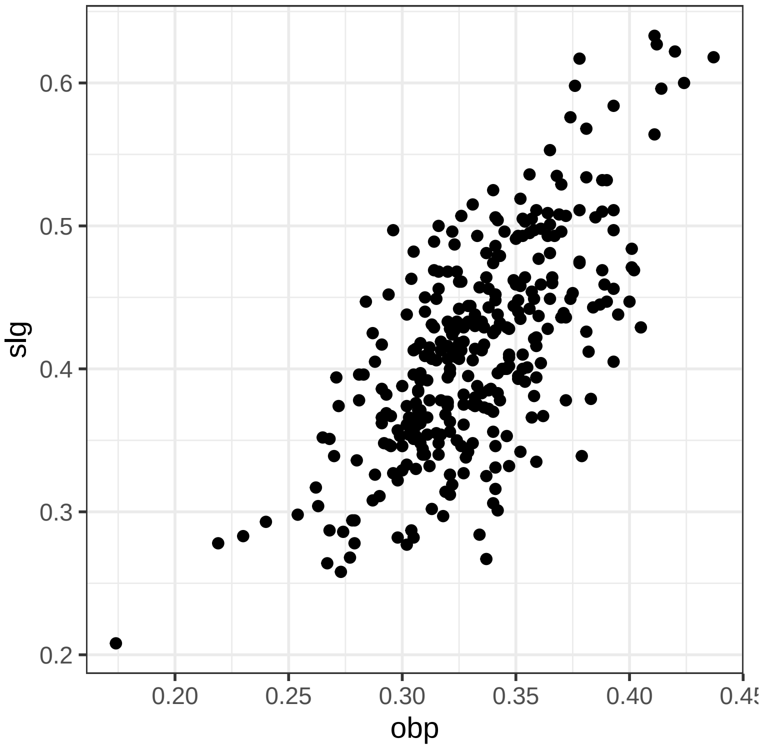
Both OBP and SLG are known as rate statistics, since they measure the frequency of certain events (as opposed to their count). In order to compare these rates sensibly, it makes sense to include only players with a reasonable number of opportunities, so that these observed rates have the chance to approach their long-run frequencies.

In Major League Baseball, batters qualify for the batting title only if they have 3.1 plate appearances per game. This translates into roughly 502 plate appearances in a 162-game season. The mlbbat10 dataset does not include plate appearances as a variable, but we can use at-bats (at_bat) – which constitute a subset of plate appearances – as a proxy.

Exercise

- Use filter() to create a scatterplot for slg as a function of obp among players who had at least 200 at-bats.

```
library(dplyr)
# Scatterplot of slg vs. obp
ntib <- mlbbat10 %>%
  filter(at_bat >= 200)
p2 <- ggplot(data = ntib, aes(y = slg, x = obp)) +
  geom_point() +
  theme_bw()
p2
```



- Find the row of mlbbat10 corresponding to the one player with at least

200 at-bats whose obp was below 0.200.

```
# Identify the outlying player
mlbbat10 %>%
  filter(at_bat >=200, obp < 0.2)
```

```
# A tibble: 1 x 19
  name  team  position  game at_bat   run   hit double triple home_run   rbi
  <fct> <fct> <fct>    <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl>    <dbl> <dbl>
1 B Wo~ LAA   3B          81    226    20    33      2      0        4    14
# ... with 8 more variables: total_base <dbl>, walk <dbl>, strike_out <dbl>,
#   stolen_base <dbl>, caught_stealing <dbl>, obp <dbl>, slg <dbl>,
#   bat_avg <dbl>
```

Chapter 3

Correlation

This chapter introduces correlation as a means of quantifying bivariate relationships.

Understanding correlation scale

In a scientific paper, three correlations are reported with the following values:

1. -0.395
2. 1.827
3. 0.738

Choose the correct interpretation of these findings.

1. is invalid.
2. **is invalid.**
3. is invalid.

Understanding correlation sign

In a scientific paper, three correlations are reported with the following values:

1. 0.582
2. 0.134
3. -0.795

Which of these values represents the strongest correlation?

Possible Answers

- 0.582

- 0.134
- **-0.795**
- Can’t tell!

3.1 Computing correlation

The `cor(x, y)` function will compute the Pearson product-moment correlation between variables, `x` and `y`. Since this quantity is symmetric with respect to `x` and `y`, it doesn’t matter in which order you put the variables.

At the same time, the `cor()` function is very conservative when it encounters missing data (e.g. `NA`s). The `use` argument allows you to override the default behavior of returning `NA` whenever any of the values encountered is `NA`. Setting the `use` argument to `"pairwise.complete.obs"` allows `cor()` to compute the correlation coefficient for those observations where the values of `x` and `y` are both not missing.

Exercise

- Use `cor()` to compute the correlation between the birthweight of babies in the `ncbirths` dataset and their mother’s age. There is no missing data in either variable.

```r
library(openintro)
DT::datatable(ncbirths)
```

```r
# Compute correlation
ncbirths %>%
  summarize(N = n(), r = cor(weight, mage))
```

```
# A tibble: 1 x 2
      N      r
  <int>  <dbl>
1  1000 0.0551
```

- Compute the correlation between the birthweight and the number of weeks of gestation for all non-missing pairs.

```r
# Compute correlation for all non-missing pairs
ncbirths %>%
  summarize(N = n(), r = cor(weight, weeks,
                             use = "pairwise.complete.obs"))
```

```
# A tibble: 1 x 2
      N     r
  <int> <dbl>
1  1000 0.670
```

3.2 Exploring Anscombe

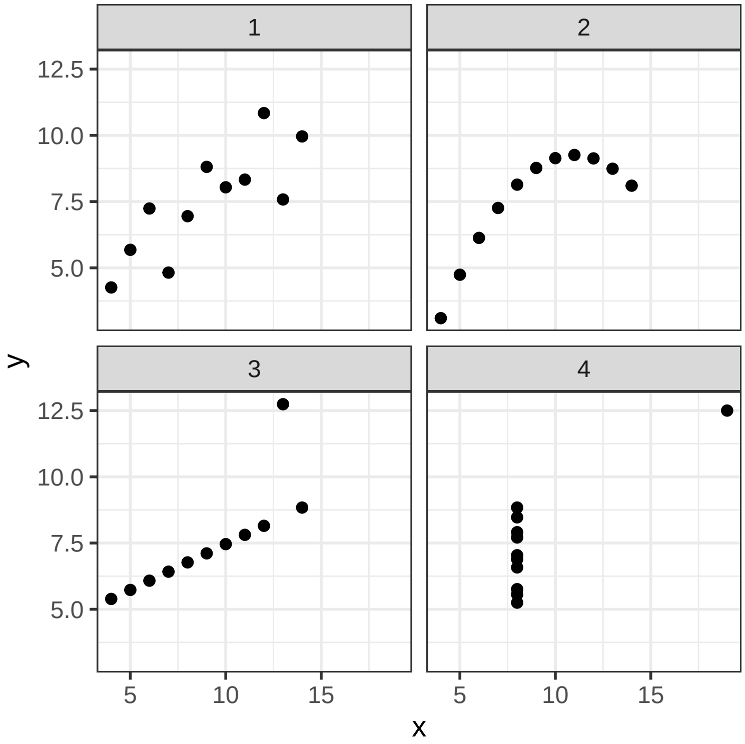
In 1973, Francis Anscombe famously created four datasets with remarkably similar numerical properties, but obviously different graphic relationships. The `Anscombe` dataset contains the `x` and `y` coordinates for these four datasets, along with a grouping variable, `set`, that distinguishes the quartet.

It may be helpful to remind yourself of the graphic relationship by viewing the four scatterplots:

```
dat <- datasets::anscombe
Anscombe <- data.frame(
    set  = rep(1:4, each = 11),
    x = unlist(dat[ ,c(1:4)]),
    y = unlist(dat[ ,c(5:8)])
    )
rownames(Anscombe) <- NULL
head(Anscombe)
```

```
  set  x    y
1   1 10 8.04
2   1  8 6.95
3   1 13 7.58
4   1  9 8.81
5   1 11 8.33
6   1 14 9.96
```

```
#
ggplot(data = Anscombe, aes(x = x, y = y)) +
  geom_point() +
  facet_wrap(vars(set)) +
  theme_bw()
```



Exercise

For each of the four sets of data points in the `Anscombe` dataset, compute the following in the order specified. Don't worry about naming any of the variables other than the first in your call to `summarize()`.

- Number of observations, `N`
- Mean of `x`
- Standard deviation of `x`
- Mean of `y`
- Standard deviation of `y`
- Correlation coefficient between `x` and `y`

```
# Compute properties of Anscombe
Anscombe %>%
  group_by(set) %>%
  summarize(N = n(), mean(x), sd(x), mean(y), sd(y), cor(x, y))
```

```
# A tibble: 4 x 7
    set     N `mean(x)` `sd(x)` `mean(y)` `sd(y)` `cor(x, y)`
* <int> <int>     <dbl>   <dbl>     <dbl>   <dbl>       <dbl>
1     1    11         9    3.32      7.50    2.03       0.816
2     2    11         9    3.32      7.50    2.03       0.816
3     3    11         9    3.32      7.5     2.03       0.816
4     4    11         9    3.32      7.50    2.03       0.817
```

Perception of correlation

Recall Figure 2.1 which displays the poverty rate of counties in the United States and the high school graduation rate in those counties from the previous chapter. Which of the following values is the correct correlation between poverty rate and high school graduation rate?

- -0.861
- **-0.681**
- -0.186
- 0.186
- 0.681
- 0.861

```
library(openintro)
cc %>%
  summarize(r = cor(poverty, hs_grad)) %>%
  round(3)
```

```
# A tibble: 1 x 1
       r
   <dbl>
1 -0.685
```

Perception of correlation (2)

Estimating the value of the correlation coefficient between two quantities from their scatterplot can be tricky. Statisticians have shown that people's perception of the strength of these relationships can be influenced by design choices like the x and y scales.

Nevertheless, with some practice your perception of correlation will improve. Study the four scatterplots in Figure 3.1, each of which you've seen in a previous exercise.

```r
library(gridExtra)
grid.arrange(p1, p2, p3, p4)
```

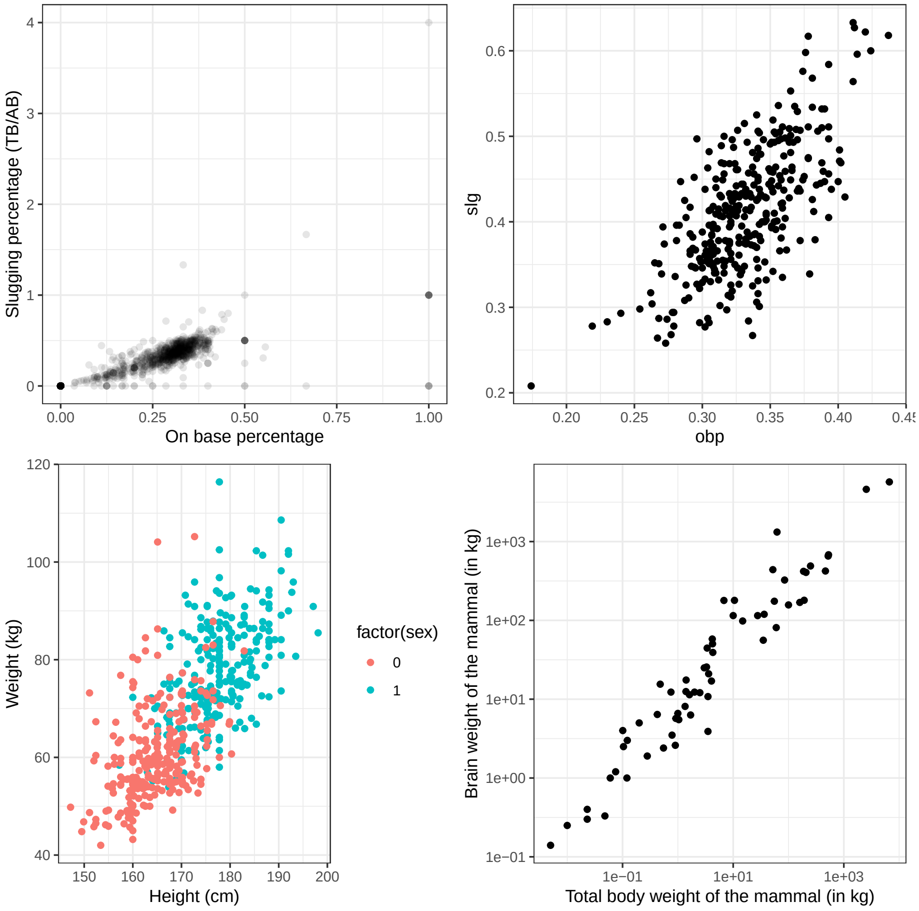


Figure 3.1: Four scatterplots

Jot down your best estimate of the value of the correlation coefficient between each pair of variables. Then, compare these values to the actual values you compute in this exercise.

Exercise

Each graph in the plotting window corresponds to an instruction below. Compute the correlation between...

- obp and slg for all players in the mlbbat10 dataset.

```
# Correlation for all baseball players
mlbbat10 %>%
  summarize(r = cor(obp, slg))
```

```
# A tibble: 1 x 1
      r
  <dbl>
1 0.815
```

- obp and slg for all players in the mlbbat10 dataset with at least 200 at-bats.

```
# Correlation for all players with at least 200 ABs
mlbbat10 %>%
  filter(at_bat >= 200) %>%
  summarize(r = cor(obp, slg))
```

```
# A tibble: 1 x 1
      r
  <dbl>
1 0.686
```

- Height and weight for each sex in the bdims dataset.

```
# Correlation of body dimensions
bdims %>%
  group_by(sex) %>%
  summarize(N = n(), r = cor(hgt, wgt))
```

```
# A tibble: 2 x 3
    sex     N     r
* <int> <int> <dbl>
1     0   260 0.431
2     1   247 0.535
```

- Body weight and brain weight for all species of mammals. Alongside this computation, compute the correlation between the same two quantities after taking their natural logarithms.

```
# Correlation among mammals, with and without log
mammals %>%
  summarize(N = n(),
            r = cor(brain_wt, body_wt),
            r_log = cor(log(brain_wt), log(body_wt)))
```

```
# A tibble: 1 x 3
      N     r r_log
  <int> <dbl> <dbl>
```

```
1    62 0.934 0.960
```

Interpreting correlation in context

Recall Figure 2.1 where you previously determined the value of the correlation coefficient between the poverty rate of counties in the United States and the high school graduation rate in those counties was -0.681. Choose the correct interpretation of this value.

- People who graduate from high school are less likely to be poor.
- Counties with lower high school graduation rates are likely to have lower poverty rates.
- **Counties with lower high school graduation rates are likely to have higher poverty rates.**
- Because the correlation is negative, there is no relationship between poverty rates and high school graduate rates.
- Having a higher percentage of high school graduates in a county results in that county having lower poverty rates.

Correlation and causation

In the San Francisco Bay Area from 1960-1967, the correlation between the birthweight of 1,236 babies and the length of their gestational period was 0.408. Which of the following conclusions is **not** a valid statistical interpretation of these results.

- We observed that babies with longer gestational periods tended to be heavier at birth.
- It may be that a longer gestational period contributes to a heavier birthweight among babies, but a randomized, controlled experiment is needed to confirm this observation.
- **Staying in the womb longer causes babies to be heavier when they are born.**
- These data suggest that babies with longer gestational periods tend to be heavier at birth, but there are many potential confounding factors that were not taken into account.

3.3 Spurious correlation in random data

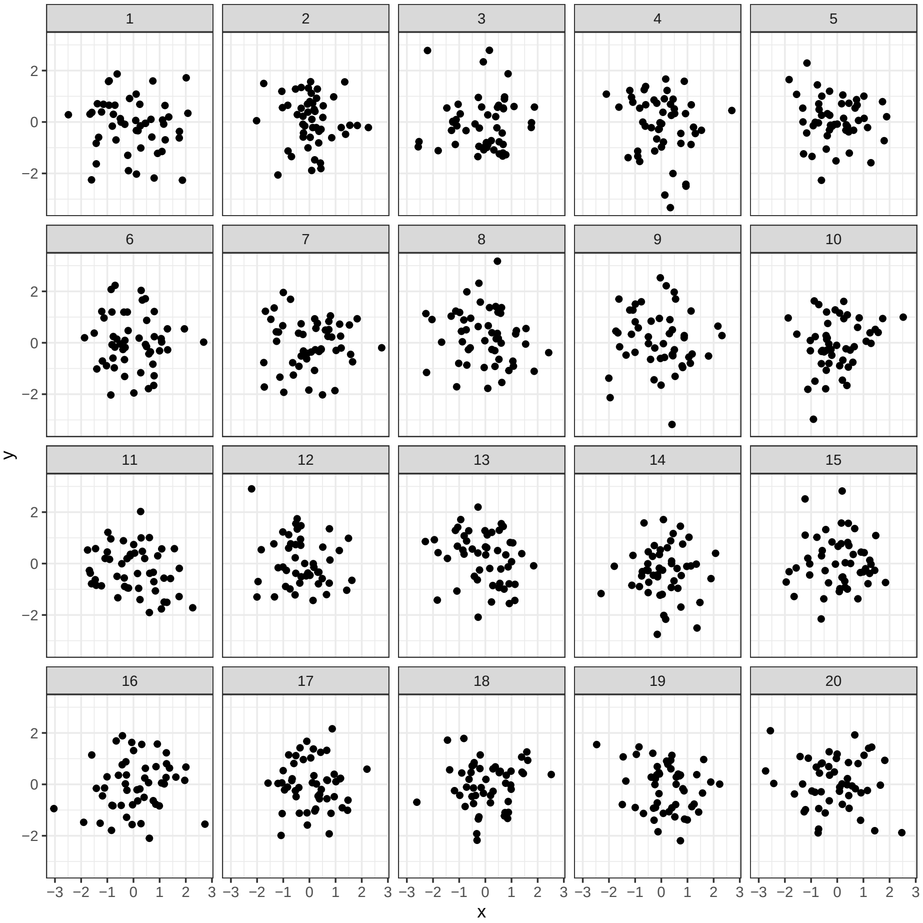
Statisticians must always be skeptical of potentially spurious correlations. Human beings are very good at seeing patterns in data, sometimes when the patterns themselves are actually just random noise. To illustrate how easy it can be to fall into this trap, we will look for patterns in truly random data.

The noise dataset contains 20 sets of `x` and `y` variables drawn at random from a standard normal distribution. Each set, denoted as `z`, has 50 observations of `x`, `y` pairs. Do you see any pairs of variables that might be meaningfully correlated? Are all of the correlation coefficients close to zero?

```
# Create noise
set.seed(9)
noise <- data.frame(x = rnorm(1000), y = rnorm(1000), z = rep(1:20, 50))
```

- Create a faceted scatterplot that shows the relationship between each of the 20 sets of pairs of random variables `x` and `y`. You will need the `facet_wrap()` function for this.

```
# Create faceted scatterplot
ggplot(dat = noise, aes(x= x, y = y)) +
  geom_point() +
  facet_wrap(vars(z)) +
  theme_bw()
```



- Compute the actual correlation between each of the 20 sets of pairs of x and y.

```
# Compute correlations for each dataset
noise_summary <- noise %>%
  group_by(z) %>%
  summarize(N = n(), spurious_cor = cor(x, y))
noise_summary
```

```
# A tibble: 20 x 3
       z     N spurious_cor
 * <int> <int>        <dbl>
 1     1    50     -0.104
 2     2    50     -0.0704
 3     3    50      0.0185
 4     4    50     -0.161
 5     5    50     -0.131
 6     6    50    -0.0356
 7     7    50    -0.00713
 8     8    50    -0.141
 9     9    50    -0.0564
10    10    50     0.216
11    11    50    -0.246
12    12    50    -0.175
13    13    50    -0.223
14    14    50     0.0175
15    15    50    -0.000720
16    16    50     0.149
17    17    50    -0.0179
18    18    50     0.0839
19    19    50    -0.221
20    20    50    -0.0969
```

- Identify the datasets that show non-trivial correlation of greater than 0.2 in absolute value.

```
# Isolate sets with correlations above 0.2 in absolute strength
noise_summary %>%
  filter(abs(spurious_cor) >= 0.2)
```

```
# A tibble: 4 x 3
      z     N spurious_cor
  <int> <int>        <dbl>
1    10    50        0.216
2    11    50       -0.246
3    13    50       -0.223
4    19    50       -0.221
```

Chapter 4

Simple linear regression

With the notion of correlation under your belt, we'll now turn our attention to simple linear models in this chapter.

4.1 The "best fit" line

The simple linear regression model for a numeric response as a function of a numeric explanatory variable can be visualized on the corresponding scatterplot by a straight line. This is a "best fit" line that cuts through the data in a way that minimizes the distance between the line and the data points.

We might consider linear regression to be a specific example of a larger class of smooth models. The `geom_smooth()` function allows you to draw such models over a scatterplot of the data itself. This technique is known as visualizing the model in the data space. The method argument to `geom_smooth()` allows you to specify what class of smooth model you want to see. Since we are exploring linear models, we'll set this argument to the value `"lm"`.

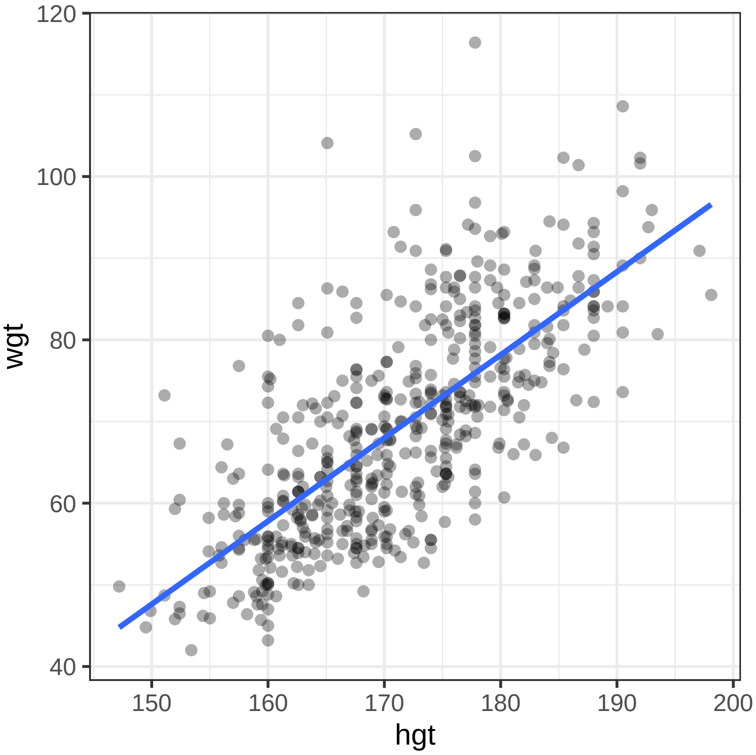
Note that `geom_smooth()` also takes an `se` argument that controls the standard error, which we will ignore for now.

Exercise

Create a scatterplot of body weight as a function of height for all individuals in the `bdims` dataset with a simple linear model plotted over the data.

```r
library(openintro)
# Scatterplot with regression line
```

```
ggplot(data = bdims, aes(x = hgt, y = wgt)) +
  geom_point(alpha = 0.33) +
  geom_smooth(method = "lm", se = FALSE) +
  theme_bw()
```



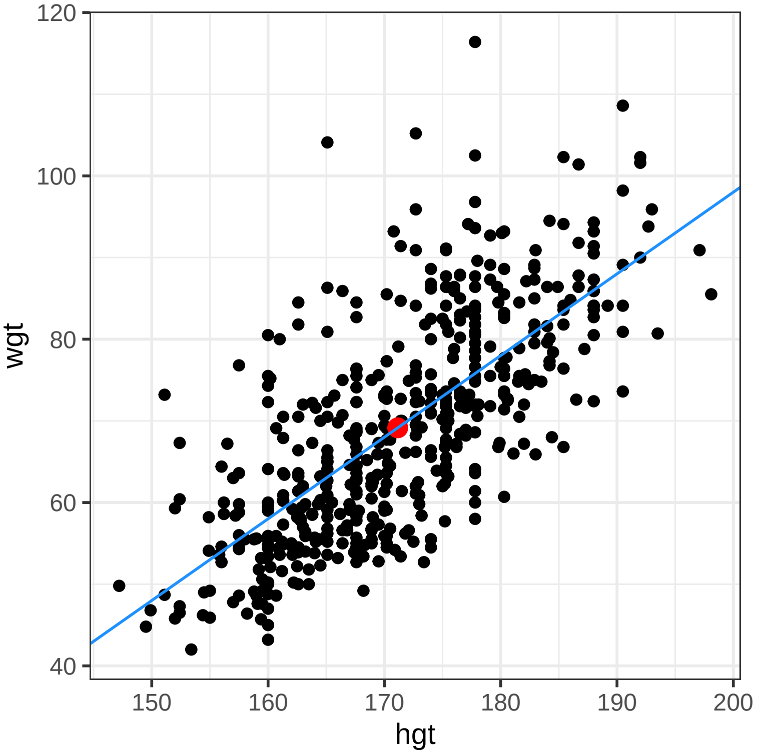
4.2 Uniqueness of least squares regression line

The least squares criterion implies that the slope of the regression line is unique. In practice, the slope is computed by R. In this exercise, you will experiment with trying to find the optimal value for the regression slope for weight as a function of height in the `bdims` dataset via trial-and-error.

To help, we've built a custom function for you called `add_line()`, which takes a single argument: the proposed slope coefficient.

The `bdims` dataset is available in your workspace. Experiment with different values (to the nearest integer) of the `my_slope` parameter until you find one that you think fits best.

```
# Estimate optimal value of my_slope
add_line(my_slope = 1.0)
```



Regression model terminology

Consider a linear regression model of the form:

$Y = \beta_0 + \beta_1 \cdot X + \varepsilon$, where $\varepsilon \sim (0, \sigma_\varepsilon)$

The slope coefficient is:

- Y
- β_0
- **1**
- ε

4.2.1 Regression model output terminology

The fitted model for the poverty rate of U.S. counties as a function of high school graduation rate is:

$\widehat{poverty} = 64.594 - 0.591 \cdot hs\_grad$

In Hampshire County in western Massachusetts, the high school graduation rate is 92.4%. These two facts imply that the poverty rate in Hampshire County is _____.

```
64.594 - 0.591 * 92.4
```

```
[1] 9.9856
```

- exactly 11.7%
- exactly 10.0%
- **expected to be about 10.0%**
- expected to be about 11.7%

4.3 Fitting a linear model "by hand"

Recall the simple linear regression model:

$$Y = b_0 + b_1 \cdot X$$

Two facts enable you to compute the slope b_1 and intercept b_0 of a simple linear regression model from some basic summary statistics.

First, the slope can be defined as:

$$b_1 = r_{X,Y} \cdot \frac{s_Y}{s_X}$$

where $r_{X,Y}$ represents the correlation (`cor()`) of `X` and `Y` and s_X and s_Y represent the standard deviation (`sd()`) of `X` and `Y`, respectively.

Second, the point $(\bar{x}, \bar{y})$ is always on the least squares regression line, where $\bar{x}$ and $\bar{y}$ denote the average of x and y, respectively.

The `bdims_summary` data frame contains all of the information you need to compute the slope and intercept of the least squares regression line for body weight (`Y`) as a function of height (`X`). You might need to do some algebra to solve for b_0!

```
bdims_summary <- bdims %>%
    summarize(N = n(), r = cor(wgt, hgt), mean_hgt = mean(hgt),
    sd_hgt = sd(hgt), mean_wgt = mean(wgt), sd_wgt = sd(wgt))
bdims_summary
```

```
# A tibble: 1 x 6
      N     r mean_hgt sd_hgt mean_wgt sd_wgt
  <int> <dbl>    <dbl>  <dbl>    <dbl>  <dbl>
1   507 0.717     171.   9.41     69.1   13.3
```

```
# Add slope and intercept
bdims_summary %>%
  mutate(slope = r*sd_wgt/sd_hgt,
         intercept = mean_wgt - slope*mean_hgt)
```

```
# A tibble: 1 x 8
      N     r mean_hgt sd_hgt mean_wgt sd_wgt slope intercept
  <int> <dbl>    <dbl>  <dbl>    <dbl>  <dbl> <dbl>     <dbl>
1   507 0.717     171.   9.41     69.1   13.3  1.02     -105.
```

4.4 Regression to the mean

Regression to the mean is a concept attributed to Sir Francis Galton. The basic idea is that extreme random observations will tend to be less extreme upon a second trial. This is simply due to chance alone. While “regression to the mean” and “linear regression” are not the same thing, we will examine them together in this exercise.

One way to see the effects of regression to the mean is to compare the heights of parents to their children’s heights. While it is true that tall mothers and fathers tend to have tall children, those children tend to be less tall than their parents, relative to average. That is, fathers who are 3 inches taller than the average father tend to have children who may be taller than average, but by less than 3 inches.

The `Galton_men` and `Galton_women` datasets contain data originally collected by Galton himself in the 1880s on the heights of men and women, respectively, along with their parents’ heights.

```
download.file("http://www.randomservices.org/random/data/Galton.txt", destfile = "./Data/Galton.t
Galton <- read.table("./Data/Galton.txt", header = TRUE)
Galton_men <- Galton %>%
  filter(Gender == "M")
```

```
Galton_women <- Galton %>%
  filter(Gender == "F")
```

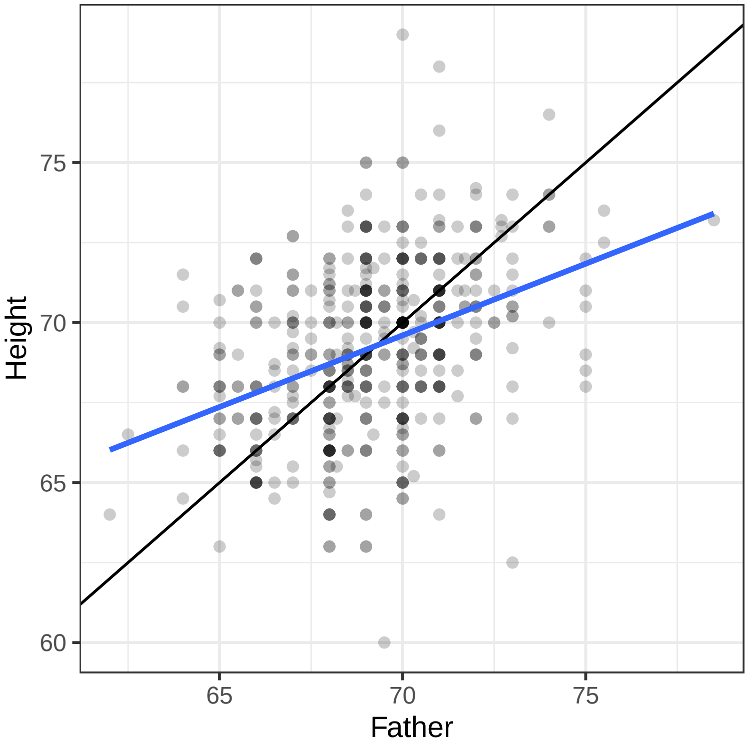
Compare the slope of the regression line to the slope of the diagonal line. What does this tell you?

Exercise

- Create a scatterplot of the height of men as a function of their father's height. Add the simple linear regression line and a diagonal line (with slope equal to 1 and intercept equal to 0) to the plot.

```
# Height of children vs. height of father
ggplot(data = Galton_men, aes(x = Father, y = Height)) +
  geom_point(alpha = 0.2) +
  geom_abline(slope = 1, intercept = 0) +
  geom_smooth(method = "lm", se = FALSE) +
  theme_bw()
```

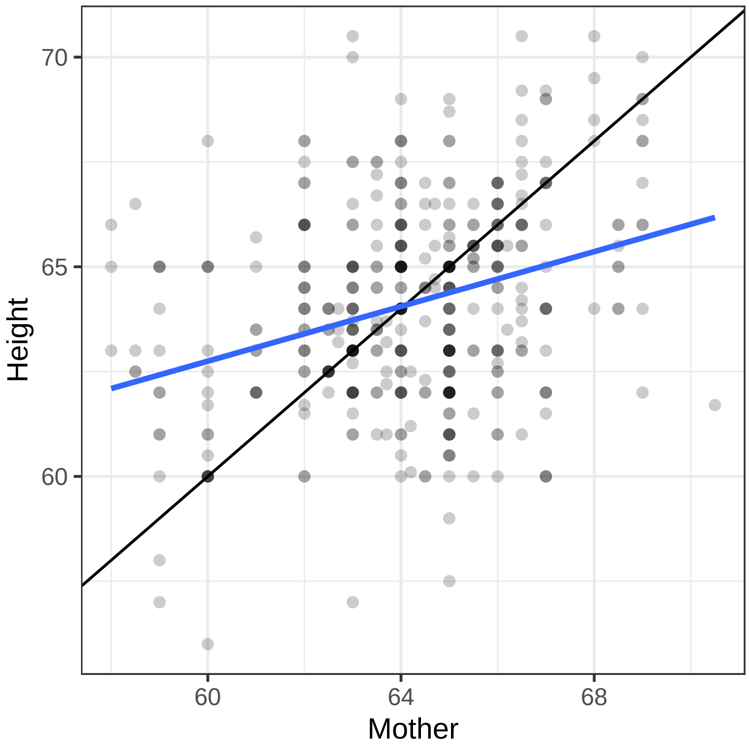
```
`geom_smooth()` using formula 'y ~ x'
```



- Create a scatterplot of the height of women as a function of their mother's height. Add the simple linear regression line and a diagonal line to the plot.

```
# Height of children vs. height of mother
ggplot(data = Galton_women, aes(x = Mother, y = Height)) +
  geom_point(alpha = 0.2) +
  geom_abline(slope = 1, intercept = 0) +
  geom_smooth(method = "lm", se = FALSE) +
  theme_bw()
```

```
`geom_smooth()` using formula 'y ~ x'
```



“Regression” in the parlance of our time

In an opinion piece about nepotism published in The New York Times in 2015, economist Seth Stephens-Davidowitz wrote that:

> “Regression to the mean is so powerful that once-in-a-generation talent basically never sires once-in-a-generation talent. It explains why Michael Jordan’s sons were middling college basketball players and

> Jakob Dylan wrote two good songs. It is why there are no American parent-child pairs among Hall of Fame players in any major professional sports league."

The author is arguing that...

- Because of regression to the mean, an outstanding basketball player is likely to have sons that are as good at basketball as him.
- Because of regression to the mean, an outstanding basketball player is likely to have sons that are not good at basketball.
- **Because of regression to the mean, an outstanding basketball player is likely to have sons that are good at basketball, but not as good as him.**
- Linear regression is incapable of evaluating musical or athletic talent.

Chapter 5

Interpreting regression models

This chapter looks at how to interpret the coefficients in a regression model.

Interpretation of coefficients

Recall that the fitted model for the poverty rare of U.S. countries as a function of high school graduation rate is:

$$\widehat{poverty} = 64.594 - 0.591 \cdot hs\_grad \tag{5.1}$$

Which of the following equations is the correct interpretation of the slope coefficient?

- Among U.S. counties, each additional percentage point increase in the poverty rate is associated with about a 0.591 percentage point decrease in the high school graduation rate.
- **Among U.S. counties, each additional percentage point increase in the high school graduation rate is associated with about a 0.591 percentage point decrease in the poverty rate.**
- Among U.S. counties, each additional percentage point increase in the high school graduation rate is associated with about a 0.591 percentage point increase in the poverty rate.
- Among U.S. counties, a 1% increase in the high school graduation rate is associated with about a 0.591% decrease in the poverty rate.

Interpretation in context

A politician interpreting the relationship between poverty rates and high school graduation rates implores his constituents:

If we can lower the poverty rate by 59%, we'll double the high school graduate rate in our county (i.e. raise it by 100%).

Which of the following mistakes in interpretation has the politician made?

- Implying that the regression model establishes a cause-and-effect relationship.
- Switching the role of the response and explanatory variables.
- Confusing percentage change with percentage point change.
- **All of the above.**
- None of the above.

5.1 Fitting simple linear models

While the `geom_smooth(method = "lm")` function is useful for drawing linear models on a scatterplot, it doesn't actually return the characteristics of the model. As suggested by that syntax, however, the function that creates linear models is `lm()`. This function generally takes two arguments:

- A `formula` that specifies the model
- A `data` argument for the data frame that contains the data you want to fit to the model

The `lm()` function return a model object having class `"lm"`. This object contains lots of information about your regression model, including the data used to fit the model, the specification of the model, the fitted values and residuals, etc.

Excercise

- Using the `bdims` dataset, create a linear model for the weight of people as a function of their height.

```r
library(openintro)
# Linear model for weight as a function of height
lm(wgt ~ hgt, data = bdims)
```

```

Call:
lm(formula = wgt ~ hgt, data = bdims)

Coefficients:
(Intercept)          hgt
   -105.011        1.018
```

- Using the `mlbBat10` dataset, create a linear model for `SLG` as a function of `OBP`.

```r
# Linear model for SLG as a function of OBP
lm(slg ~ obp, data = mlbbat10)
```

```

Call:
lm(formula = slg ~ obp, data = mlbbat10)

Coefficients:
(Intercept)          obp
   0.009407     1.110323
```

*Using the `mammals` dataset, create a linear model for the body weight of mammals as a function of their brain weight, after taking the natural log of both variables.

```r
# Log-linear model for body weight as a function of brain weight
lm(log(body_wt) ~ log(brain_wt), data = mammals)
```

```

Call:
lm(formula = log(body_wt) ~ log(brain_wt), data = mammals)

Coefficients:
  (Intercept)  log(brain_wt)
       -2.509          1.225
```

Units and scale

In the previous examples, we fit two regression models:

$$\widehat{wgt} = -105.011 + 1.018 \cdot hgt \tag{5.2}$$

and

$$\widehat{SLG} = 0.009 + 1.110 \cdot OBP \tag{5.3}$$

Which of the following statements is **incorrect**?

- A person who is 170 cm tall is expected to weigh about 68 kg.
- **Because the slope coefficient for OBP is larger (1.110) than the slope coefficient for hgt (1.018), we can conclude that the association between OBP and SLG is stronger than the association between height and weight.**
- None of the above.

5.2 The lm summary output

An `lm` object contains a host of information about the regression model that you fit. There are various ways of extracting different pieces of information.

The `coef()` function displays only the values of the coefficients. Conversely, the `summary()` function displays not only that information, but a bunch of other information, including the associated standard error and p-value for each coefficient, the, R^2 adjusted R^2, and the residual standard error. The summary of an `"lm"` object in R is very similar to the output you would see in other statistical computing environments (e.g. Stata, SPSS, etc.)

Excercise

We have already created the mod object, a linear model for the weight of individuals as a function of their height, using the `bdims` dataset and the code

```
mod <- lm(wgt ~ hgt, data = bdims)
```

Now, you will:

- Use `coef()` to display the coefficients of `mod`.

```
mod <- lm(wgt ~ hgt, data = bdims)
# Show the coefficients
coef(mod)
```

```
(Intercept)         hgt
-105.011254    1.017617
```

- Use `summary()` to display the full regression output of `mod()`.

```
summary(mod)
```

```
Call:
lm(formula = wgt ~ hgt, data = bdims)

Residuals:
    Min      1Q  Median      3Q     Max 
-18.743  -6.402  -1.231   5.059  41.103 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -105.01125    7.53941  -13.93   <2e-16 ***
hgt            1.01762    0.04399   23.14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.308 on 505 degrees of freedom
Multiple R-squared:  0.5145,    Adjusted R-squared:  0.5136 
F-statistic: 535.2 on 1 and 505 DF,  p-value: < 2.2e-16
```

5.3 Fitted values and residuals

Once you have fit a regression model, you are often interested in the fitted values ($\hat{y}_i$) and the residuals (e_i), where i indexes the observations. Recall that:

$$e_i = y_i - \hat{y}_i \tag{5.4}$$

The least squares fitting procedure guarantees that the mean of the residuals is zero (n.b., numerical instability may result in the computed values not being exactly zero). At the same time, the mean of the fitted values must equal the mean of the response variable.

In this exercise, we will confirm these two mathematical facts by accessing the fitted values and residuals with the `fitted.values()` and `residuals()` functions, respectively, for the following model:

```
mod <- lm(wgt ~ hgt, data = bdims)
```

Exercise

- Confirm that the mean of the body weights equals the mean of the fitted values of `mod`.

```r
# Mean of weights equal to mean of fitted values?
mean(bdims$wgt) == mean(fitted.values(mod))
```

```
[1] TRUE
```

- Compute the mean of the residuals of `mod`.

```r
mean(resid(mod))
```

```
[1] -3.665467e-16
```

5.4 Tidying your linear model

As you fit a regression model, there are some quantities (e.g. R^2) hat apply to the model as a whole, while others apply to each observation (e.g. $\hat{y}_i$). If there are several of these per-observation quantities, it is sometimes convenient to attach them to the original data as new variables.

The `augment()` function from the `broom` package does exactly this. It takes a model object as an argument and returns a data frame that contains the data on which the model was fit, along with several quantities specific to the regression model, including the fitted values, residuals, leverage scores, and standardized residuals.

Excercise

The same linear model from the last exercise, `mod`, is available in your workspace.

- Load the `broom` package.

```r
# Load broom
library(broom)
```

- Create a new data frame called `bdims_tidy` that is the augmentation of the `mod` linear model.

```r
# Create bdims_tidy
bdims_tidy <- augment(mod)
```

- View the `bdims_tidy` data frame using `glimpse()`.

```
# Glimpse the resulting data frame
glimpse(bdims_tidy)
```

```
Rows: 507
Columns: 8
$ wgt        <dbl> 65.6, 71.8, 80.7, 72.6, 78.8, 74.8, 86.4, 78.4, 62.0, 81...
$ hgt        <dbl> 174.0, 175.3, 193.5, 186.5, 187.2, 181.5, 184.0, 184.5, ...
$ .fitted    <dbl> 72.05406, 73.37697, 91.89759, 84.77427, 85.48661, 79.686...
$ .resid     <dbl> -6.4540648, -1.5769666, -11.1975919, -12.1742745, -6.686...
$ .hat       <dbl> 0.002154570, 0.002358152, 0.013133942, 0.007238576, 0.00...
$ .sigma     <dbl> 9.312824, 9.317005, 9.303732, 9.301360, 9.312471, 9.3147...
$ .cooksd    <dbl> 5.201807e-04, 3.400330e-05, 9.758463e-03, 6.282074e-03, ...
$ .std.resid <dbl> -0.69413418, -0.16961994, -1.21098084, -1.31269063, -0.7...
```

5.5 Making predictions

The `fitted.values()` function or the `augment()`-ed data frame provides us with the fitted values for the observations that were in the original data. However, once we have fit the model, we may want to compute expected values for observations that were not present in the data on which the model was fit. These types of predictions are called out-of-sample.

The `ben` data frame contains a height and weight observation for one person.

```
ben <- data.frame(wgt = 74.8, hgt = 182.8)
```

The `mod` object contains the fitted model for weight as a function of height for the observations in the `bdims` dataset. We can use the `predict()` function to generate expected values for the weight of new individuals. We must pass the data frame of new observations through the `newdata` argument.

Excercise

The same linear model, `mod`, is defined in your workspace.

- Print `ben` to the console.

```
# Print ben
ben
```

```
   wgt   hgt
1 74.8 182.8
```

- Use predict() with the newdata argument to compute the expected weight of the individual in the ben data frame.

```
# Predict the weight of ben
predict(mod, newdata = ben)
```

```
       1
81.00909
```

Note that the data frame ben has variables with the exact same names as those in the fitted model.

5.6 Adding a regression line to a plot manually

The geom_smooth() function makes it easy to add a simple linear regression line to a scatterplot of the corresponding variables. And in fact, there are more complicated regression models that can be visualized in the data space with geom_smooth(). However, there may still be times when we will want to add regression lines to our scatterplot manually. To do this, we will use the geom_abline() function, which takes slope and intercept arguments. Naturally, we have to compute those values ahead of time, but we already saw how to do this (e.g. using coef()).

The coefs data frame contains the model estimates retrieved from coef(). Passing this to geom_abline() as the data argument will enable you to draw a straight line on your scatterplot.

Excercise

Use geom_abline() to add a line defined in the coefs data frame to a scatterplot of weight vs. height for individuals in the bdims dataset.

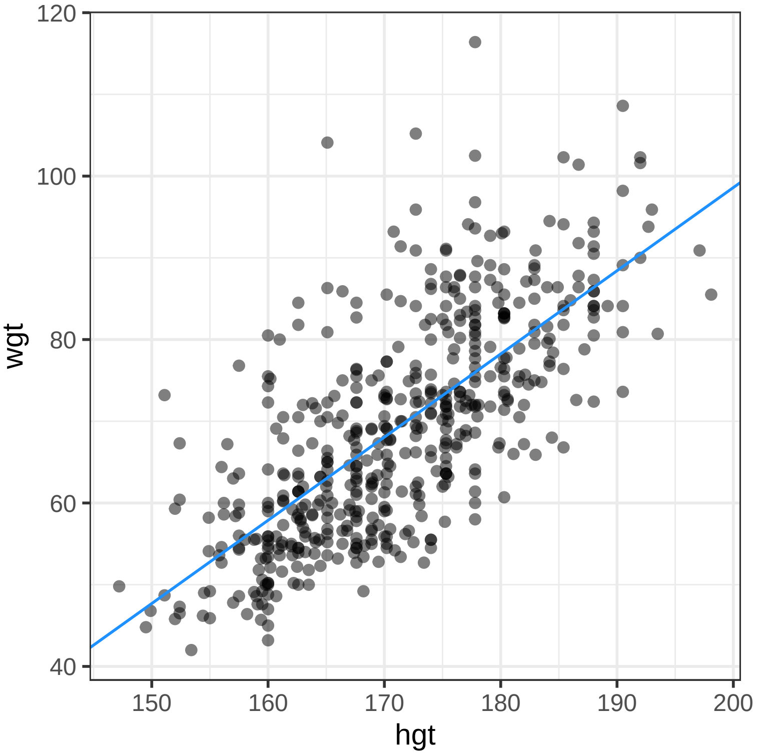
```
coef(mod)
```

```
(Intercept)         hgt
-105.011254    1.017617
```

```
coefs <- data.frame(intercept = -105, slope = 1.018)
coefs
```

```
  intercept slope
1      -105 1.018
```

```
# Add the line to the scatterplot
ggplot(data = bdims, aes(x = hgt, y = wgt)) +
  geom_point(alpha = 0.5) +
  theme_bw() +
  geom_abline(data = coefs,
              aes(intercept = intercept, slope = slope),
              color = "dodgerblue")
```



Chapter 6

Model Fit

In this final chapter, you'll learn how to assess the "fit" of a simple linear regression model.

RMSE

The residual standard error reported for the regression model for poverty rate of U.S. counties in terms of high school graduation rate is 4.67. What does this mean?

- **The typical difference between the observed poverty rate and the poverty rate predicted by the model is about 4.67 percentage points.**
- The typical difference between the observed poverty rate and the poverty rate predicted by the model is about 4.67%.

*The model explains about 4.67% of the variability in poverty rate among counties.

- The model correctly predicted the poverty rate of 4.67% of the counties.

6.1 Standard error of residuals

One way to assess strength of fit is to consider how far off the model is for a typical case. That is, for some observations, the fitted value will be very close to the actual value, while for others it will not. The magnitude of a typical residual can give us a sense of generally how close our estimates are.

However, recall that some of the residuals are positive, while others are negative. In fact, it is guaranteed by the least squares fitting procedure that the mean of the residuals is zero. Thus, it makes more sense to compute the square root of the mean squared residual, or root mean squared error (RMSE). R calls this quantity the *residual standard error*.

To make this estimate unbiased, you have to divide the sum of the squared residuals by the degrees of freedom in the model. Thus,

$$RMSE = \sqrt{\frac{\sum_i e_i^2}{d.f.}} = \sqrt{\frac{SSE}{d.f.}} \tag{6.1}$$

You can recover the residuals from `mod` with `residuals()`, and the degrees of freedom with `df.residual()`.

Exercise

*View a `summary()` of `mod`.

```r
# View summary of model
summary(mod)
```

```

Call:
lm(formula = wgt ~ hgt, data = bdims)

Residuals:
    Min      1Q  Median      3Q     Max 
-18.743  -6.402  -1.231   5.059  41.103 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -105.01125    7.53941  -13.93   <2e-16 ***
hgt            1.01762    0.04399   23.14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.308 on 505 degrees of freedom
Multiple R-squared:  0.5145,	Adjusted R-squared:  0.5136 
F-statistic: 535.2 on 1 and 505 DF,  p-value: < 2.2e-16
```

- Compute the mean of the `residuals()` and verify that it is approximately zero.

```
# Compute the mean of the residuals
mean(resid(mod))
```

```
[1] -3.665467e-16
```

- Use `residuals()` and `df.residual()` to compute the root mean squared error (RMSE), a.k.a. *residual standard error*.

```
# Compute RMSE
sqrt(sum(residuals(mod)^2) / df.residual(mod))
```

```
[1] 9.30804
```

6.2 Assessing simple linear model fit

Recall that the coefficient of determination (R^2), can be computed as

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{Var(e)}{Var(y)}, \tag{6.2}$$

where e is the vector of residuals and y is the response variable. This gives us the interpretation of R^2 as the percentage of the variability in the response that is explained by the model, since the residuals are the part of that variability that remains unexplained by the model.

The `bdims_tidy` data frame is the result of `augment()`-ing the `bdims` data frame with the mod for wgt as a function of `hgt`.

```
bdims_tidy <- augment(mod)
```

- Use the `summary()` function to view the full results of mod.

```
# View model summary
summary(mod)
```

```

Call:
lm(formula = wgt ~ hgt, data = bdims)

Residuals:
    Min      1Q  Median      3Q     Max 
-18.743  -6.402  -1.231   5.059  41.103 

Coefficients:
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -105.01125    7.53941  -13.93   <2e-16 ***
hgt            1.01762    0.04399   23.14   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.308 on 505 degrees of freedom
Multiple R-squared:  0.5145,	Adjusted R-squared:  0.5136 
F-statistic: 535.2 on 1 and 505 DF,  p-value: < 2.2e-16
```

- Use the `bdims_tidy` data frame to compute the R^2 of `mod` manually using the formula above, by computing the ratio of the variance of the residuals to the variance of the response variable.

```r
bdims_tidy %>%
  summarize(var_y = var(wgt), var_e = var(.resid)) %>%
  mutate(R_squared = 1 - var_e/var_y)
```

```
# A tibble: 1 x 3
  var_y var_e R_squared
  <dbl> <dbl>     <dbl>
1  178.  86.5     0.515
```

This means that 51.4% of the variability in weight is explained by height.

Interpretation of R^2

The R^2 reported for the regression model for poverty rate of U.S. counties in terms of high school graduation rate is 0.464.

```r
lm(formula = poverty ~ hs_grad, data = countyComplete) %>%
  summary()
```

How should this result be interpreted?

- 46.4% of the variability in high school graduate rate among U.S. counties can be explained by poverty rate.
- **46.4% of the variability in poverty rate among U.S. counties can be explained by high school graduation rate.**
- This model is 46.4% effective.
- The correlation between poverty rate and high school graduation rate is 0.464.

6.3 Linear vs. average

The R^2 gives us a numerical measurement of the strength of fit relative to a null model based on the average of the response variable:

$$\hat{y} = \bar{y} \tag{6.3}$$

This model has an R^2 of zero because $SSE = SST$. That is, since the fitted values ($\hat{y}_{null}$) are all equal to the average ($\bar{y}$), the residual for each observation is the distance between that observation and the mean of the response. Since we can always fit the null model, it serves as a baseline against which all other models will be compared.

In Figure 6.1, we visualize the residuals for the null model (`mod_null` at left) vs. the simple linear regression model (`mod_hgt` at right) with height as a single explanatory variable. Try to convince yourself that, if you squared the lengths of the grey arrows on the left and summed them up, you would get a larger value than if you performed the same operation on the grey arrows on the right.

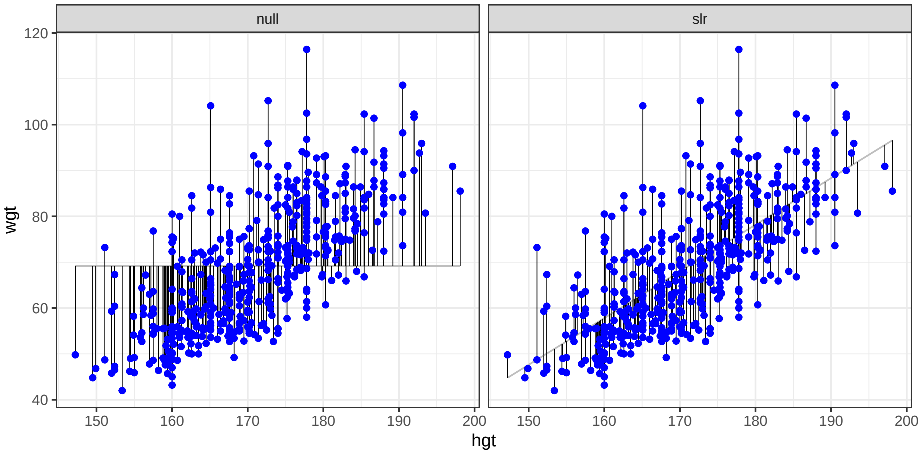


Figure 6.1: At left, the model based on overall average weight. At right, the simple linear regression model.

It may be useful to preview these `augment()`-ed data frames with `glimpse()`:

```r
glimpse(mod_null)
glimpse(mod_hgt)
```

Excercise

- Compute the sum of the squared residuals (SSE) for the null model `mod_null`.

```
# Compute SSE for null model
mod_null %>%
  summarize(SSE = var(wgt))
```

```
# A tibble: 1 x 1
    SSE
  <dbl>
1  178.
```

- Compute the sum of the squared residuals (SSE) for the regression model `mod_hgt`.

```
# Compute SSE for regression model
mod_hgt %>%
  summarize(SSE = var(.resid))
```

```
# A tibble: 1 x 1
    SSE
  <dbl>
1  86.5
```

6.4 Leverage

The leverage of an observation in a regression model is defined entirely in terms of the distance of that observation from the mean of the explanatory variable. That is, observations close to the mean of the explanatory variable have low leverage, while observations far from the mean of the explanatory variable have high leverage. Points of high leverage may or may not be influential.

The `augment()` function from the `broom` package will add the leverage scores (`.hat`) to a model data frame.

Excercise

- Use `augment()` to list the top 6 observations by their leverage scores, in descending order.

```
mod <- lm(slg ~ obp, data = mlbbat10)
# Rank points of high leverage
mod %>%
  augment() %>%
  arrange(desc(.hat)) %>%
  head()
```

```
# A tibble: 6 x 8
    slg   obp .fitted .resid    .hat .sigma .cooksd .std.resid
  <dbl> <dbl>   <dbl>  <dbl>   <dbl>  <dbl>   <dbl>      <dbl>
1     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
2     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
3     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
4     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
5     4     1    1.12  2.88  0.0184  0.108 4.22        21.2
6     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
```

6.5 Influence

As noted previously, observations of high leverage may or may not be influential. The influence of an observation depends not only on its leverage, but also on the magnitude of its residual. Recall that while leverage only takes into account the explanatory variable ($\hat{x}$), the residual depends on the response variable ($\hat{y}$) and the fitted value ($\hat{y}$).

Influential points are likely to have high leverage and deviate from the general relationship between the two variables. We measure influence using Cook's distance, which incorporates both the leverage and residual of each observation.

- Use `augment()` to list the top 6 observations by their Cook's distance (`.cooksd`), in descending order.

```
# Rank influential points
mod %>%
 augment() %>%
 arrange(desc(.cooksd)) %>%
 head()
```

```
# A tibble: 6 x 8
    slg   obp .fitted .resid    .hat .sigma .cooksd .std.resid
  <dbl> <dbl>   <dbl>  <dbl>   <dbl>  <dbl>   <dbl>      <dbl>
1   4     1       1.12   2.88  0.0184   0.108   4.22       21.2
2   0     1       1.12  -1.12  0.0184   0.133   0.638      -8.25
3   0     1       1.12  -1.12  0.0184   0.133   0.638      -8.25
```

```
4  0    1       1.12  -1.12  0.0184   0.133   0.638       -8.25
5  0    1       1.12  -1.12  0.0184   0.133   0.638       -8.25
6  1.67 0.667   0.750  0.917 0.00676  0.134   0.154        6.72
```

6.6 Removing outliers

Observations can be outliers for a number of different reasons. Statisticians must always be careful—and more importantly, transparent—when dealing with outliers. Sometimes, a better model fit can be achieved by simply removing outliers and re-fitting the model. However, one must have strong justification for doing this. A desire to have a higher $R^2$is not a good enough reason!

In the `mlbBat10` data, the outlier with an OBP of 0.550 is Bobby Scales, an infielder who had four hits in 13 at-bats for the Chicago Cubs. Scales also walked seven times, resulting in his unusually high OBP. The justification for removing Scales here is weak. While his performance was unusual, there is nothing to suggest that it is not a valid data point, nor is there a good reason to think that somehow we will learn more about Major League Baseball players by excluding him.

Nevertheless, we can demonstrate how removing him will affect our model.

- Use `filter()` to create a subset of `mlbBat10` called `nontrivial_players` consisting of only those players with at least 10 at-bats and `OBP` of below 0.500.

```
# Create nontrivial_players
nontrivial_players <- mlbbat10 %>%
 filter(at_bat >= 10, obp < 0.500)
```

- Fit the linear model for `SLG` as a function of `OBP` for the `nontrivial_players`. Save the result as `mod_cleaner`.

```
# Fit model to new data
mod_cleaner <- lm(slg ~ obp, data = nontrivial_players)
```

- View the `summary()` of the new model and compare the slope and R^2 to those of `mod`, the original model fit to the data on all players.

```
# View model summary
summary(mod_cleaner)
```

```

Call:
lm(formula = slg ~ obp, data = nontrivial_players)
```

```
Residuals:
     Min       1Q   Median       3Q      Max 
-0.31383 -0.04165 -0.00261  0.03992  0.35819 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.043326   0.009823  -4.411 1.18e-05 ***
obp          1.345816   0.033012  40.768  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.07011 on 734 degrees of freedom
Multiple R-squared:  0.6937,	Adjusted R-squared:  0.6932 
F-statistic:  1662 on 1 and 734 DF,  p-value: < 2.2e-16
```

```r
summary(mod_cleaner)$r.square
```

```
[1] 0.6936567
```

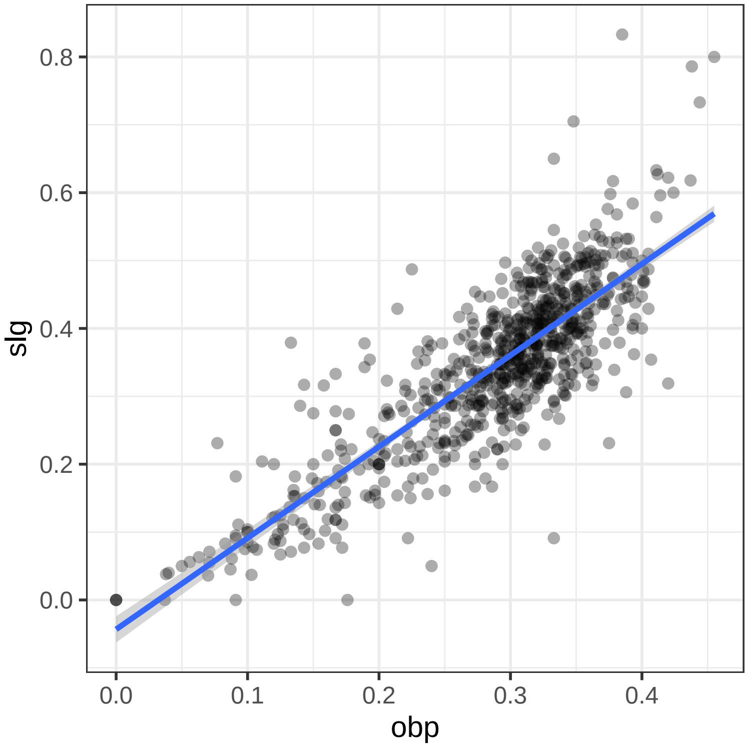
```r
# Original with all players
summary(mod)$r.square
```

```
[1] 0.6635126
```

- Visualize the new model with `ggplot()` and the appropriate `geom_*()` functions.

```r
ggplot(data = nontrivial_players, aes(x = obp, y = slg)) +
  geom_point(alpha = 0.33) +
  geom_smooth(method = "lm") +
  theme_bw()
```

```
`geom_smooth()` using formula 'y ~ x'
```



6.7 High leverage points

Not all points of high leverage are influential. While the high leverage observation corresponding to Bobby Scales in the previous exercise is influential, the three observations for players with OBP and SLG values of 0 are not influential.

This is because they happen to lie right near the regression anyway. Thus, while their extremely low OBP gives them the power to exert influence over the slope of the regression line, their low SLG prevents them from using it.

- The linear model, `mod`, is available in your workspace. Use a combination of `augment()`, `arrange()` with two arguments, and `head()` to find the top 6 observations with the highest leverage but the lowest Cook's distance.

```
# Rank high leverage points
mod %>%
 augment() %>%
arrange(desc(.hat), .cooksd) %>%
head()
```

```
# A tibble: 6 x 8
    slg   obp .fitted .resid   .hat .sigma .cooksd .std.resid
  <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>      <dbl>
1     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
2     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
3     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
4     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
5     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
6     1     1    1.12 -0.120 0.0184  0.137 0.00729     -0.882
```