Correlation and Regression in R

Devin Otto

March 11, 2021

Contents

Introduction

The content within this document is taken from the Correlation and Regression in R course on DataCamp. This course's instructor, Ben Baumer, is an Assistant Professor in the Statistical & Data Sciences Program at Smith College. The solutions to the course problems have been completed by Devin Otto and can be found in the material below. DataCamp utilizes some custom datasets and functions that cannot be found within the course, leading to the necessity of constructing this custom material within this R Markdown.

Libraries

```
library(dplyr)
library(ggplot2)
library(openintro)
library(tidyverse)
```

1 Chapter: Visualizing Two Variables

1.1 Scatterplots

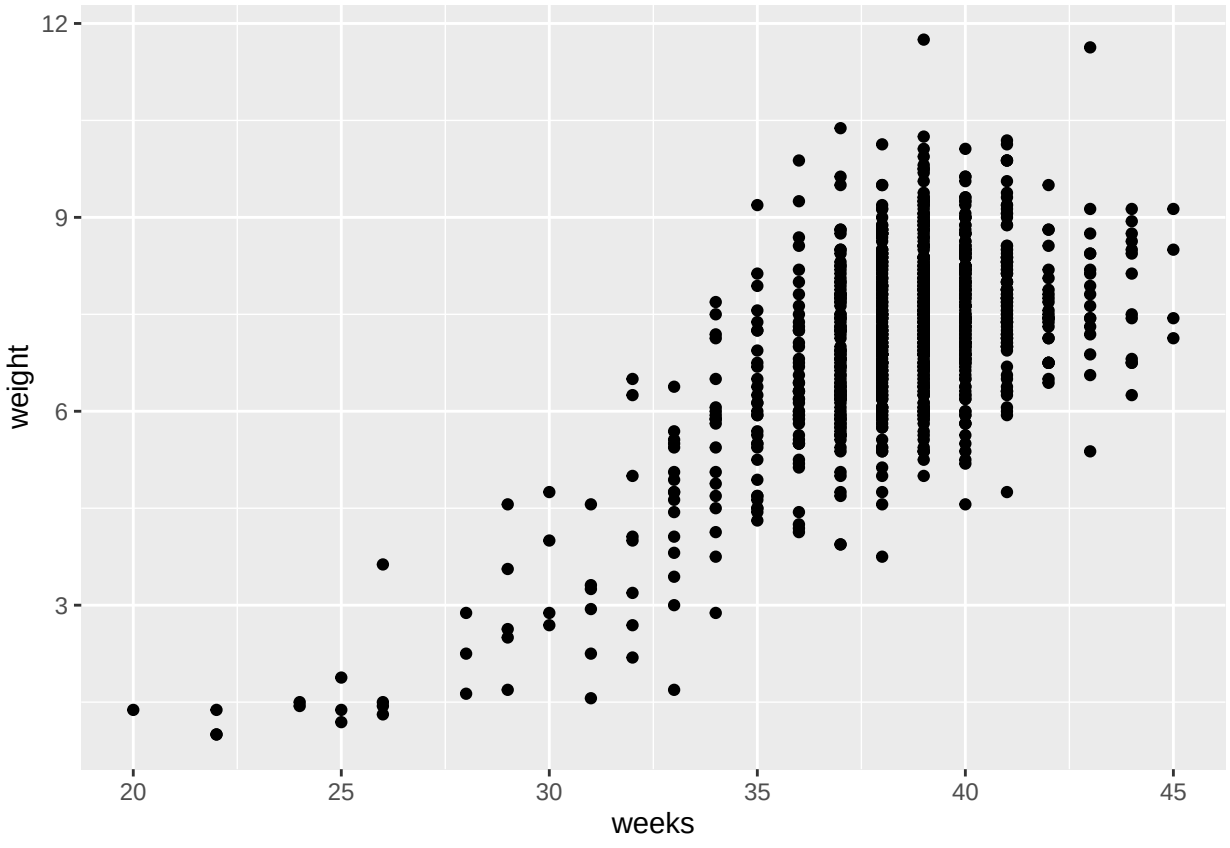
Scatterplots are the most common and effective tools for visualizing the relationship between two numeric variables.

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.

- **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
ggplot(data = ncbirths, aes(y = weight, x = weeks)) +
  geom_point()
```



1.2 Boxplots as discretized/conditioned scatterplots

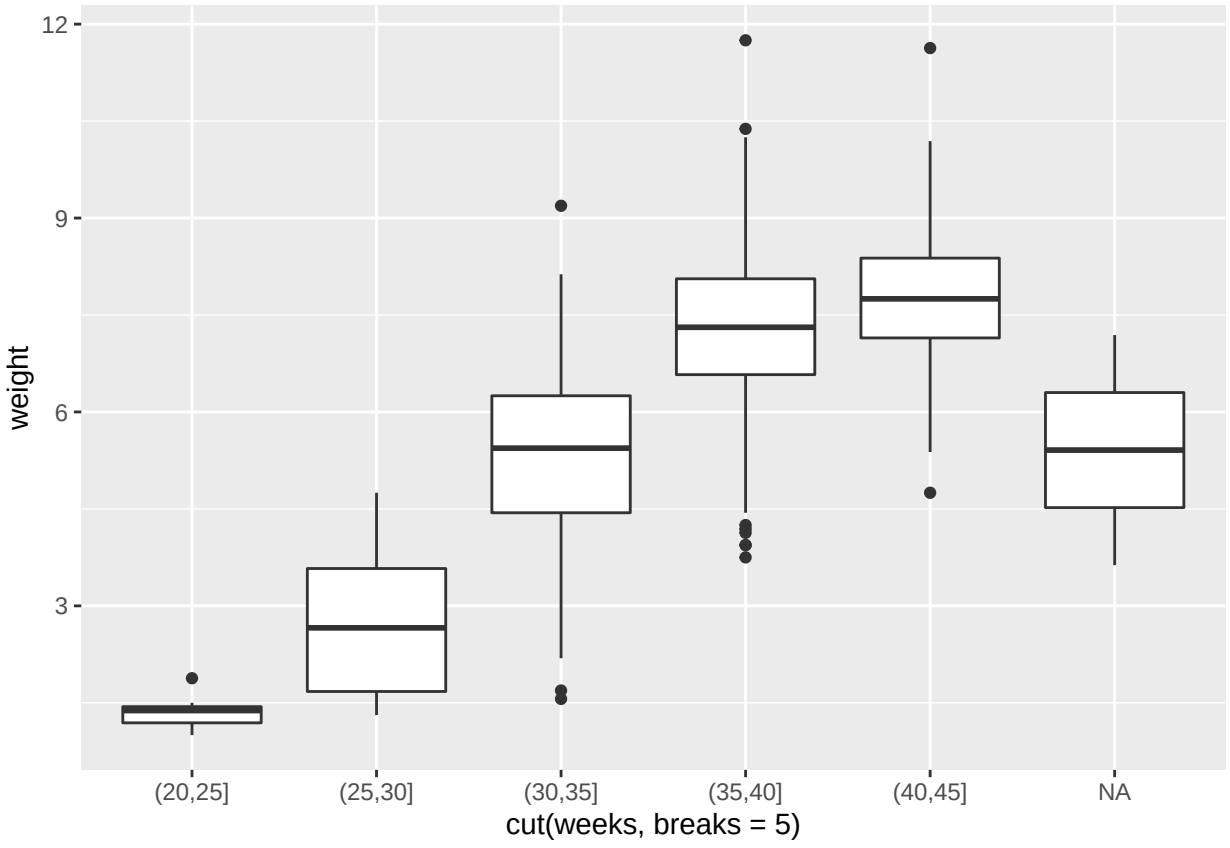
If it is helpful, you can think of boxplots as scatterplots for which the variable on the x-axis has been discretized.

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()
```



1.3 Creating Scatterplots

Creating scatterplots is simple and they are so useful that it is 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.

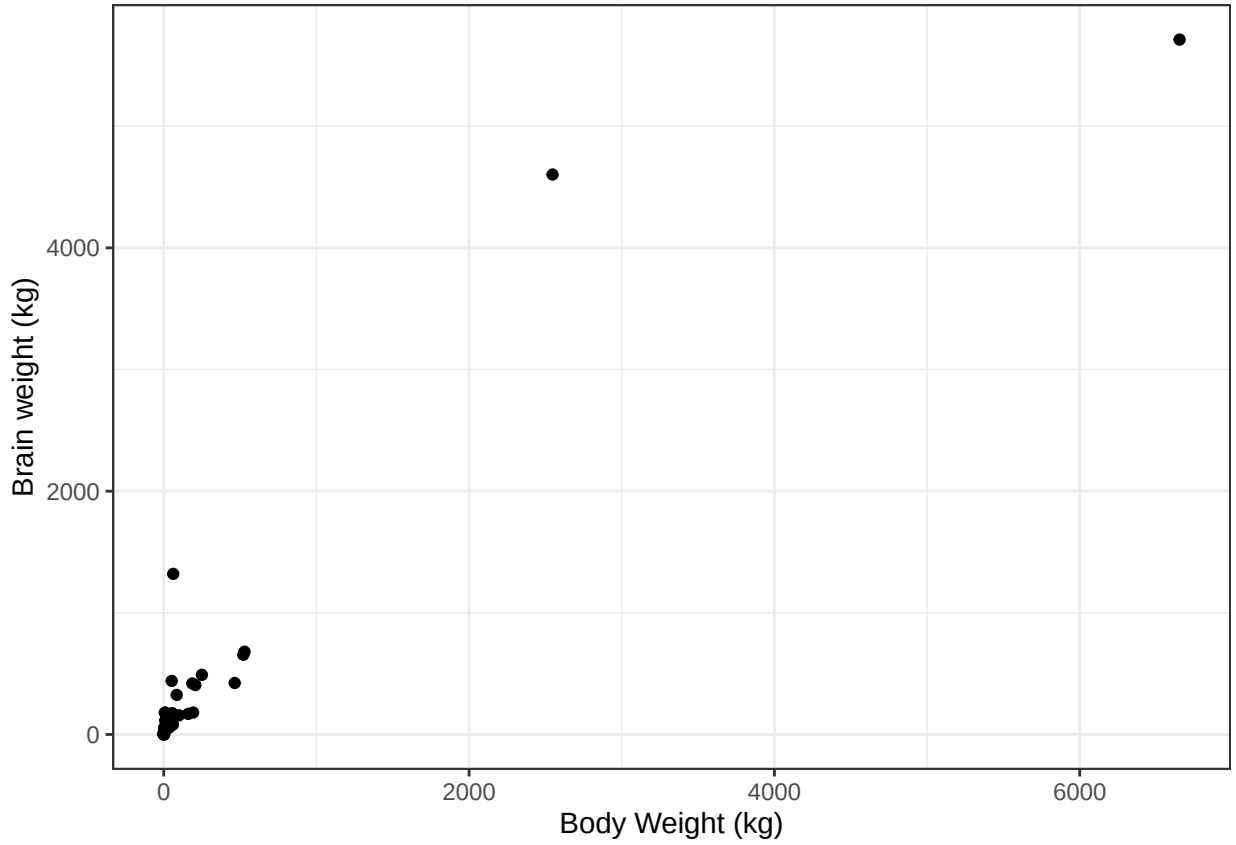
In this exercise, and throughout this chapter, we will be using several datasets listed below. These data are available through the `openintro` package. Briefly:

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.

- **Exercise**

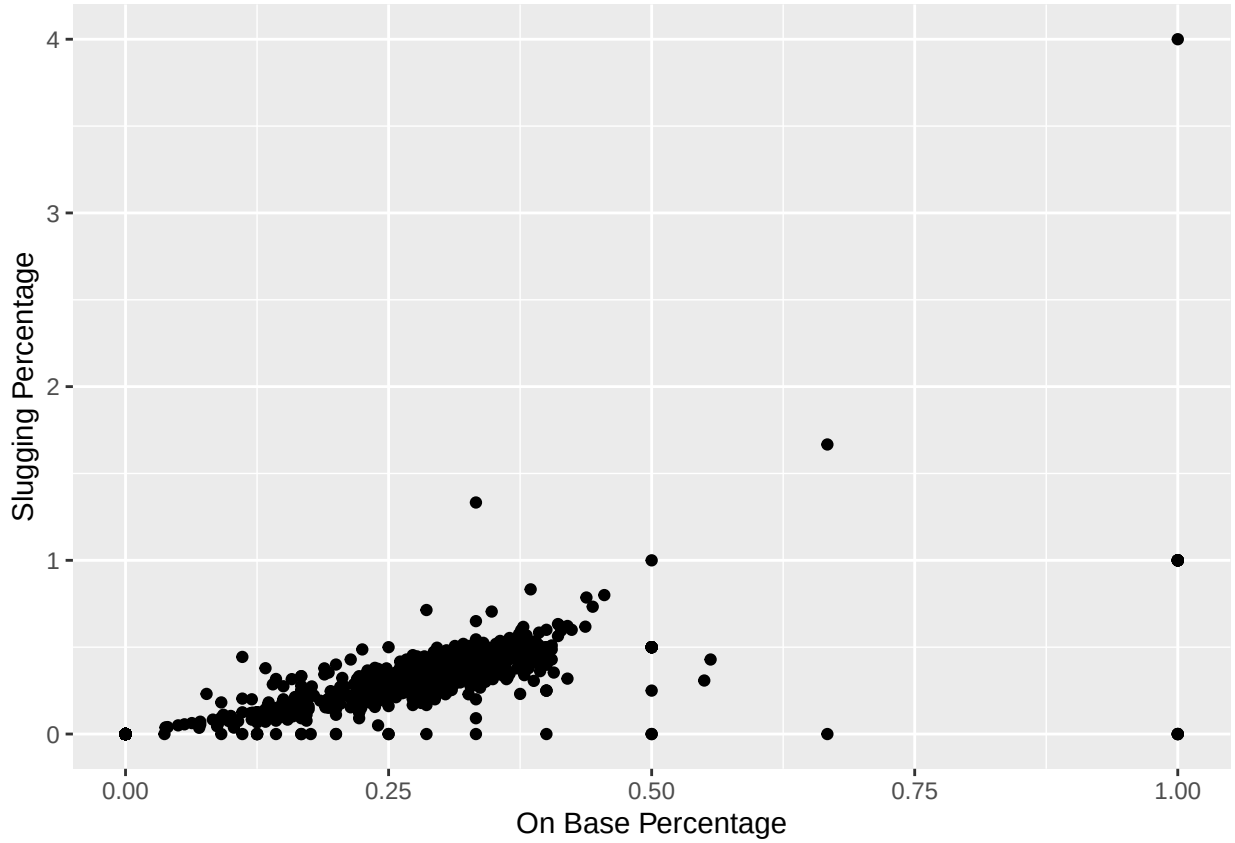
Using the `mammals` dataset, create a scatterplot illustrating how the brain weight of a mammal varies as a function of its body weight.

```
# 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)")
```



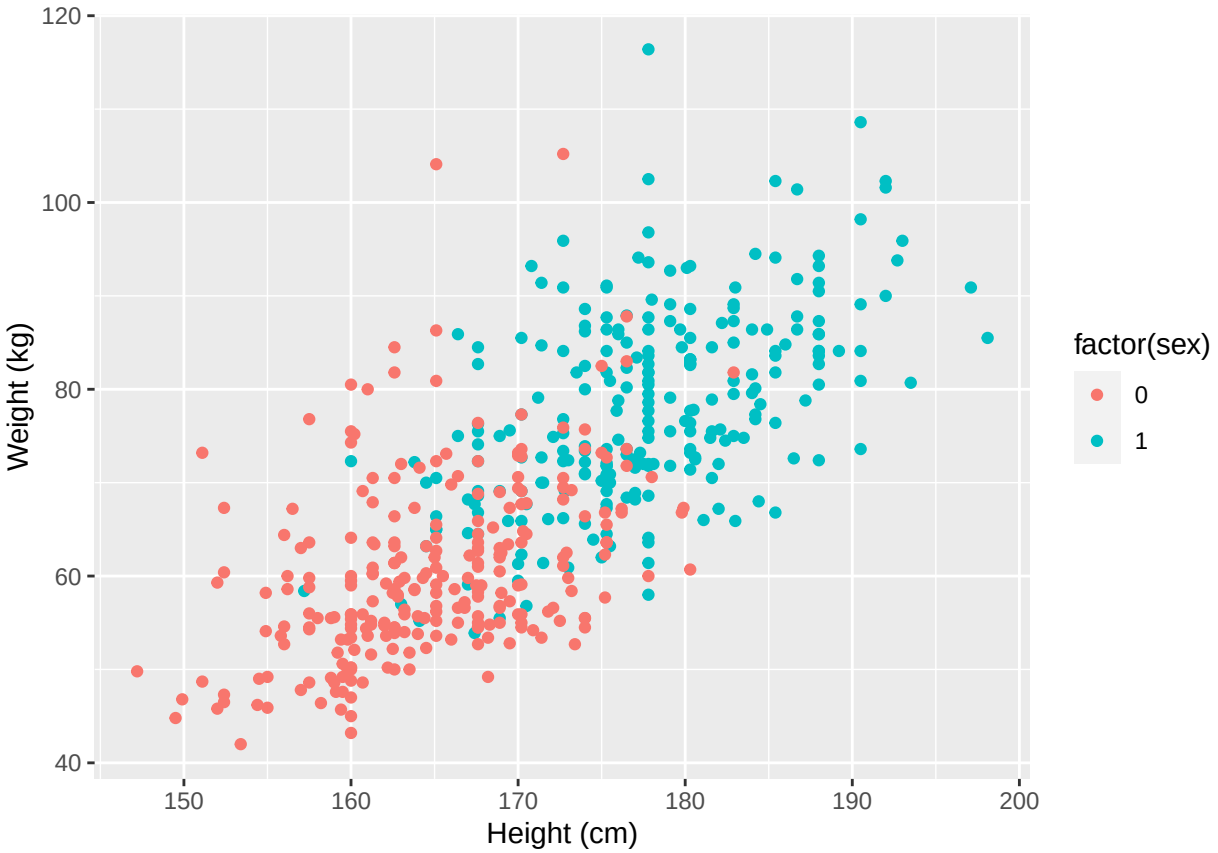
Using the `mlbBat10` dataset, create a scatterplot illustrating how the slugging percentage (SLG) of a player varies as a function of his on-base percentage (OBP).

```
# Baseball player scatterplot
ggplot(data = mlbbat10, aes(y = slg, x = obp)) +
geom_point() +
labs(x = "On Base Percentage", y = "Slugging Percentage")
```



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
ggplot(data = bdims, aes(x = hgt, y = wgt, color = factor(sex))) +
geom_point() +
labs(x = "Height (cm)", y = "Weight (kg)")
```



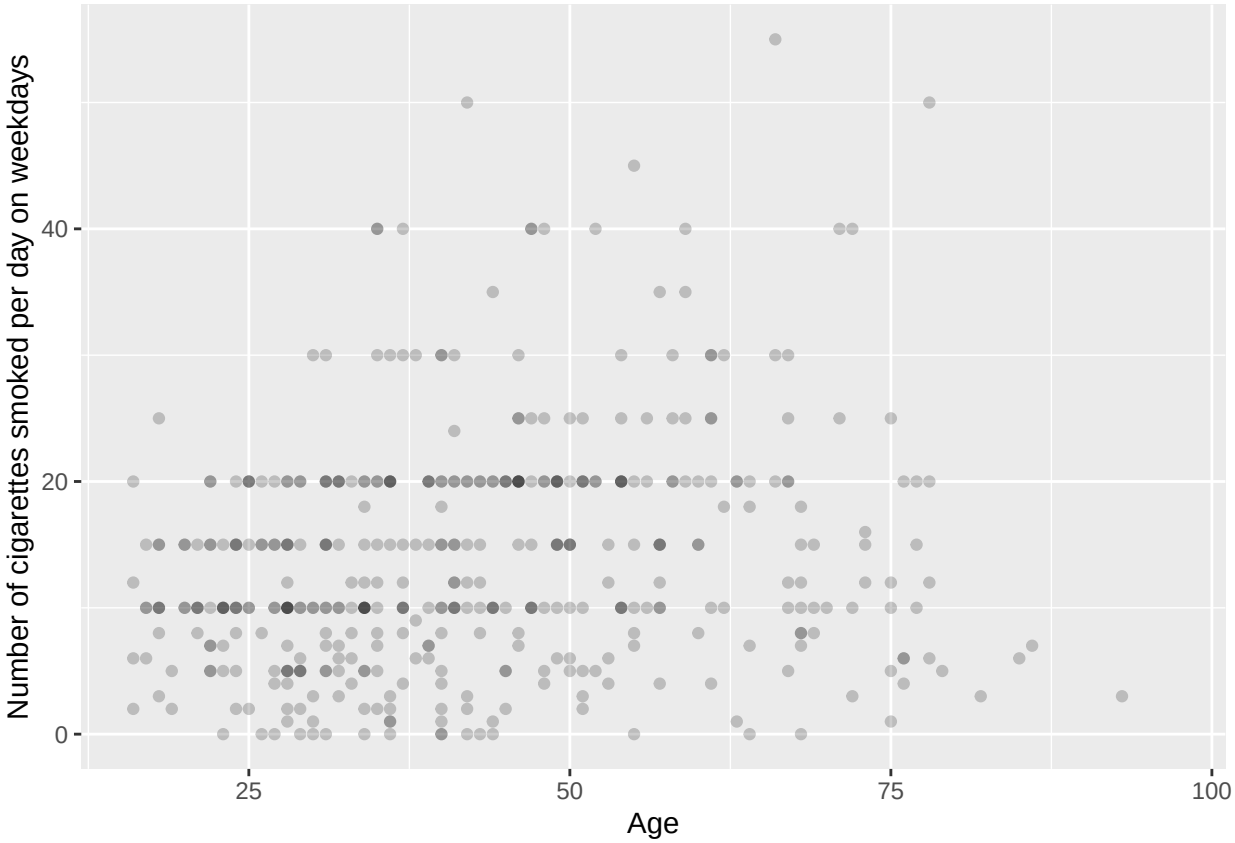
Using the `smoking` dataset, create a scatterplot illustrating how the amount that a person smokes on weekdays varies as a function of their age.

```
# Smoking scatterplot
head(smoking)
```

```
## # A tibble: 6 x 12
##   gender   age marital_status highest_qualification nationality ethnicity
##   <fct>  <int> <fct>          <fct>                  <fct>       <fct>
## 1 Male      38 Divorced       No Qualification       British     White
## 2 Female    42 Single         No Qualification       British     White
## 3 Male      40 Married        Degree                 English     White
## 4 Female    40 Married        Degree                 English     White
## 5 Female    39 Married        GCSE/O Level           British     White
## 6 Female    37 Married        GCSE/O Level           British     White
## # ... with 6 more variables: gross_income <fct>, region <fct>, smoke <fct>,
## #   amt_weekends <int>, amt_weekdays <int>, type <fct>
```

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

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


Number of cigarettes smoked per day on weekdays
0
20
40
25
50
75
100
Age

1.4 Characterizing scatterplots

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

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

- **Possible Answers:**
  - Linear, positive, strong
  - Linear, negative, weak
  - **Linear, negative, moderately strong**
  - Non-linear, negative, strong

1.5 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?

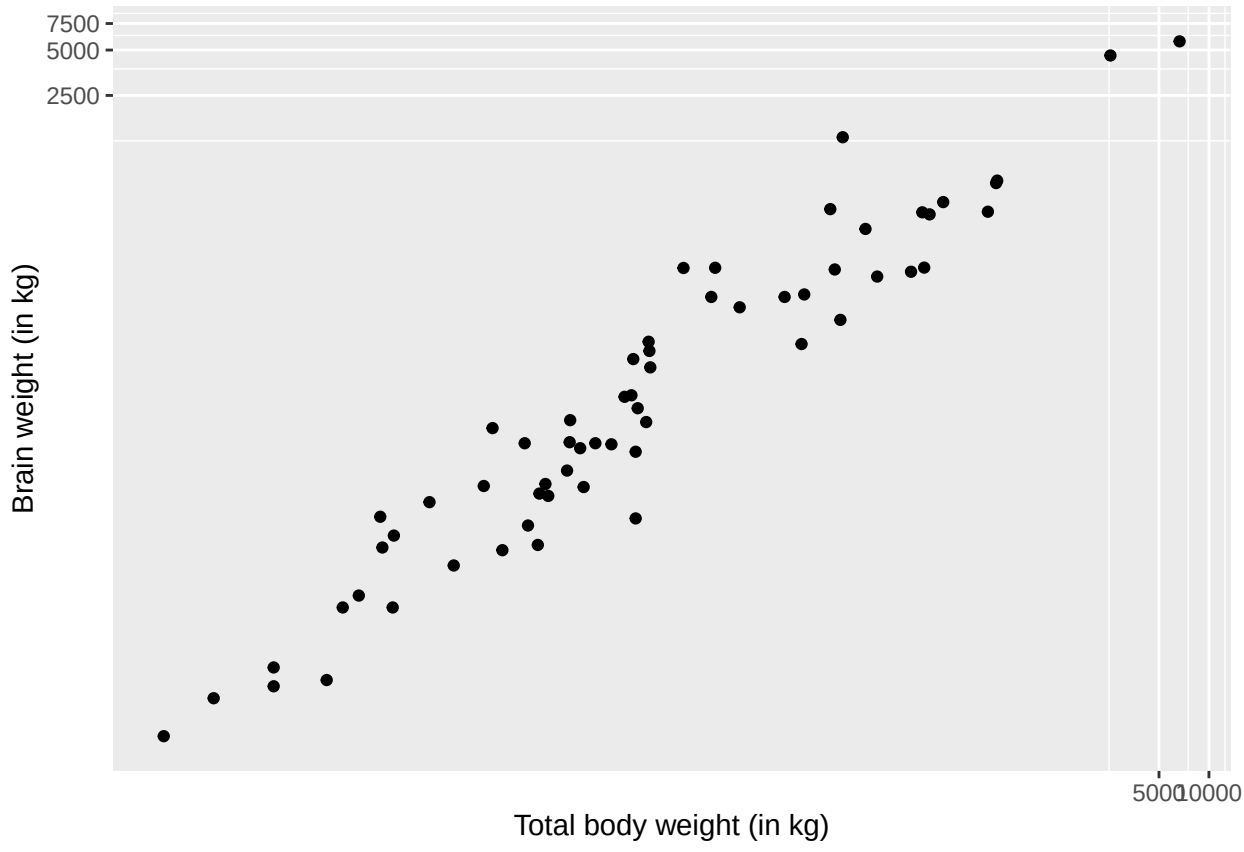
`ggplot2` provides several different mechanisms for viewing transformed relationships. The `coord_trans()` function transforms the coordinates of the plot. Alternatively, the `scale_x_log10()` and `scale_y_log10()` functions perform a base-10 log transformation of each axis. Note the differences in the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

The `mammals` dataset is available in your workspace.

- **Exercise**

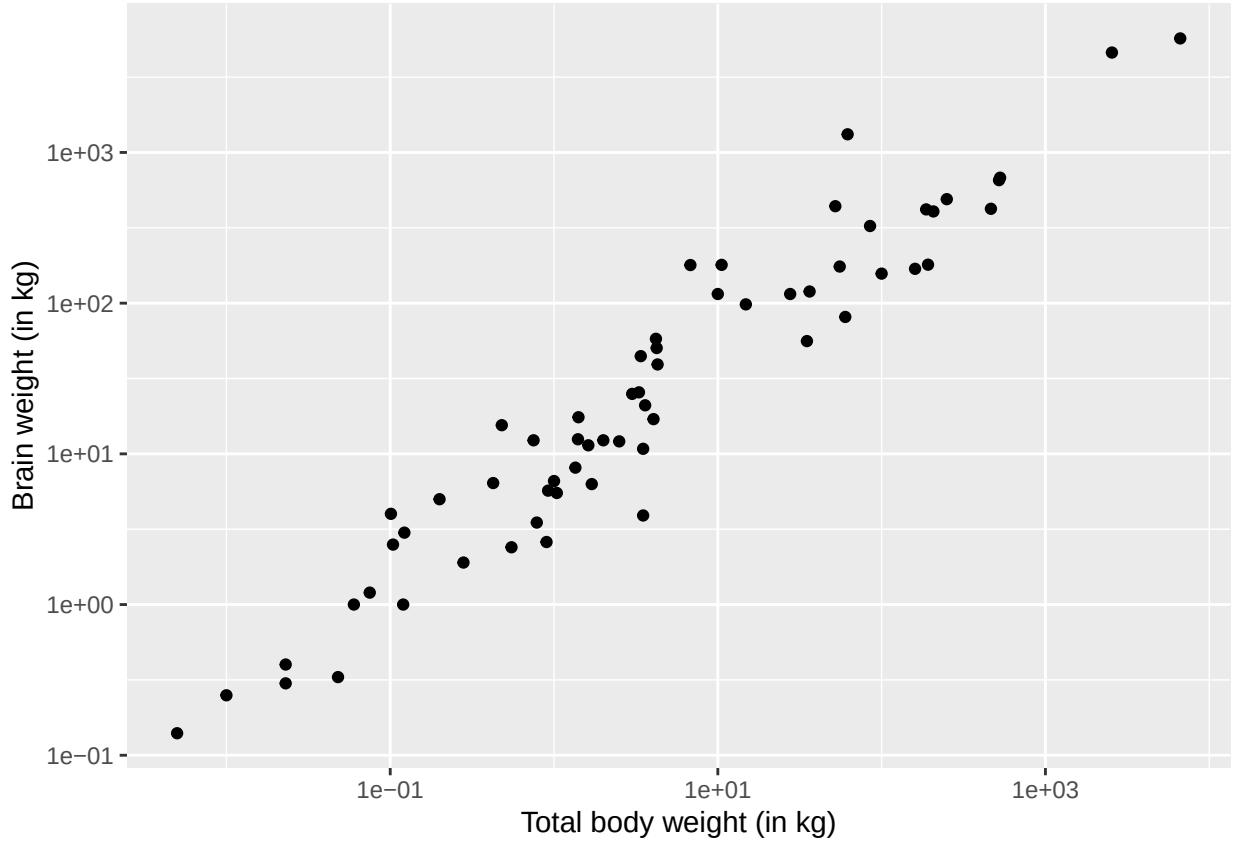
Use `coord_trans()` to create a scatterplot showing how a mammal's brain weight varies as a function of its body weight, where both the x and y axes are on a `"log10"` scale.

```
# Scatterplot with coord_trans()
ggplot(data = mammals, aes(y = brain_wt, x = body_wt)) +
geom_point() +
coord_trans(x = "log10", y = "log10") +
labs(x = "Total body weight (in kg)",
y = "Brain weight (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()
ggplot(data = mammals, aes(x = body_wt, y = brain_wt)) +
geom_point() +
scale_x_log10() +
scale_y_log10() +
labs(x = "Total body weight (in kg)",
y = "Brain weight (in kg)")
```



1.6 Identifying outliers

In Chapter 5, 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.

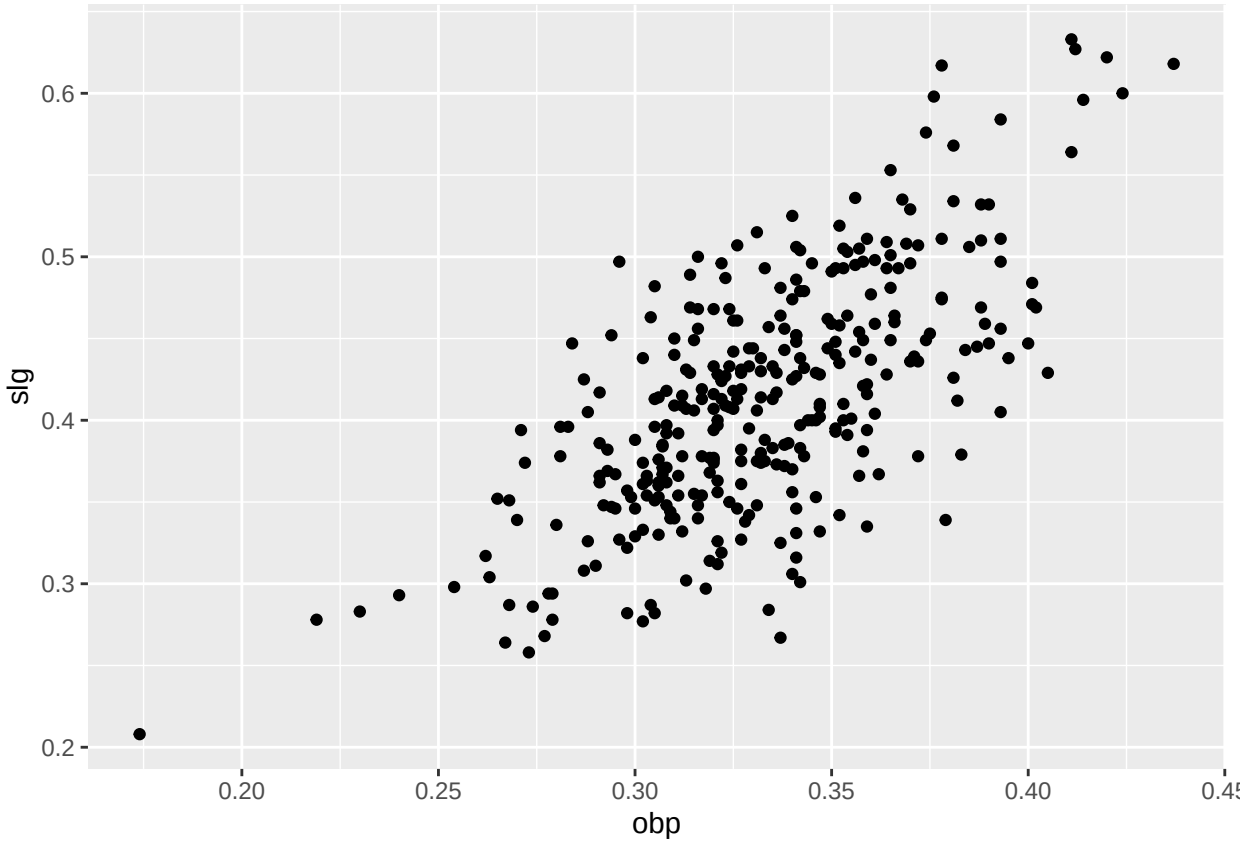
Both OBP and SLG are known as rate statistics, since they measure the frequency of certain events (as opposed to their count). In order to compare these rates sensibly, it makes sense to include only players with a reasonable number of opportunities, so that these observed rates have the chance to approach 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 (`AB`) – which constitute a subset of plate appearances – as a proxy.

- **Exercise**

Use `filter()` to keep only players who had at least 200 at-bats, assigning to `ab_gt_200`. Using `ab_gt_200`, create a scatterplot for `SLG` as a function of `OBP`. Find the row of `ab_gt_200` corresponding to the one player (with at least 200 at-bats) whose OBP was below 0.200.

```
# Filter for AB greater than or equal to 200
ab_gt_200 <- mlbbat10 %>% filter(at_bat >= 200)

# Scatterplot of SLG vs. OBP
ggplot(ab_gt_200, aes(x = obp, y = slg)) +
  geom_point()
```




```
# Identify the outlying player
ab_gt_200 %>%
  filter(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 Wood 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>
```

2 Chapter: Correlation

2.1 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.

- **Possible Answers:**
  - (1) is invalid.
  - **(2) is invalid.**
  - (3) is invalid.
  - Both (1) and (2) are invalid.
  - Both (2) and (3) are invalid.

2.2 Understanding correlation sign

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

- 0.582
- 0.134
- -0.795

Which of these values represents the strongest correlation?

- **Possible Answers:**
  - 0.582
  - 0.134
  - **-0.795**
  - Can't tell!

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

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

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

```
# 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
```

2.4 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
```

- **Exercise**

For each of the four `sets` of data points in the `Anscombe` dataset, compute the following in the order specified. Names are provided 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_of_x = mean(x),
    std_dev_of_x = sd(x),
    mean_of_y = mean(y),
    std_dev_of_y = sd(y),
    correlation_between_x_and_y = cor(x,y)
  )
```

```
## # A tibble: 4 x 7
##     set     N mean_of_x std_dev_of_x mean_of_y std_dev_of_y correlation_between~
##   <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
```

2.5 Perception of correlation

Recall the scatterplot between 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?

- **Possible Answers**
- -0.861
- **-0.681**
- -0.186
- 0.186
- 0.681
- 0.861

2.6 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.

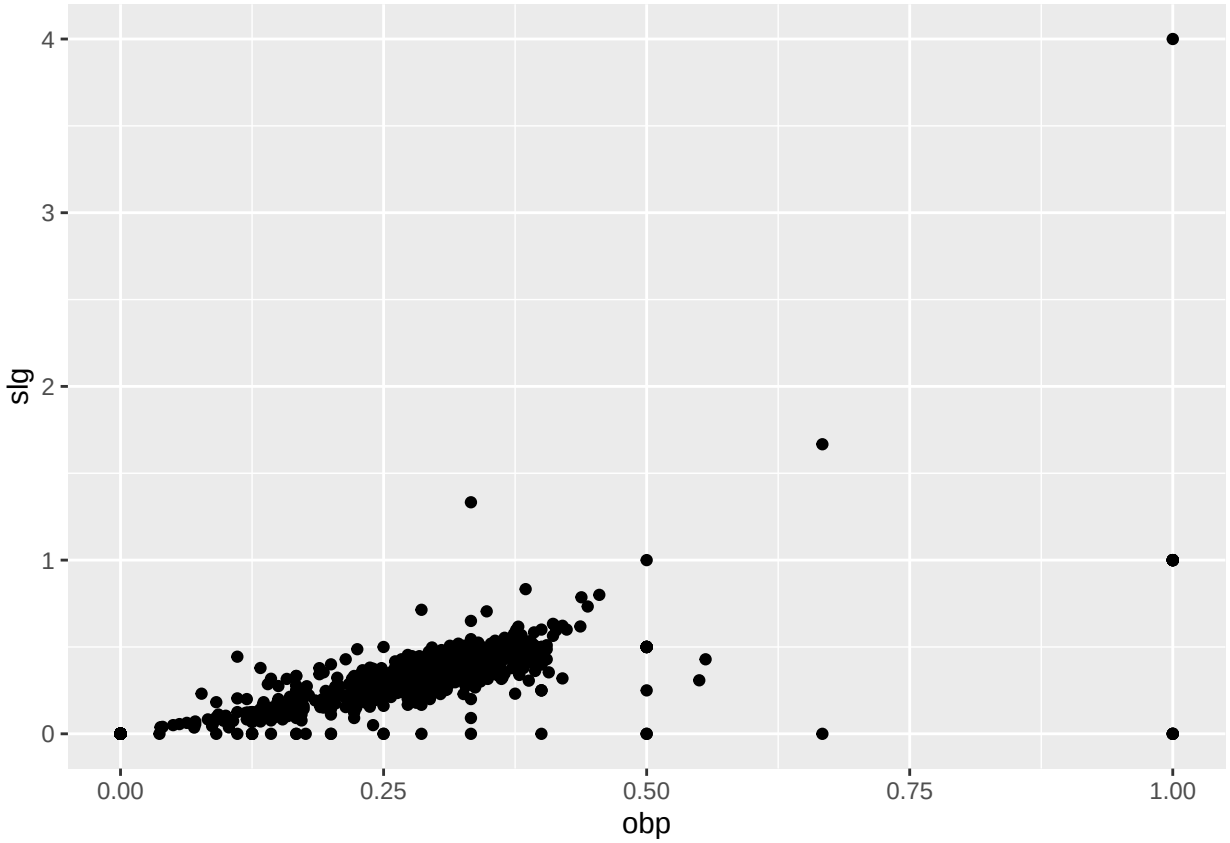
Nevertheless, with some practice your perception of correlation will improve. Toggle through the four scatterplots in the plotting window, each of which you've seen in a previous exercise. Jot down your best estimate of the value of the correlation coefficient between each pair of variables. Then, compare these values to the actual values you compute in this exercise.

If you're having trouble recalling variable names, it may help to preview a dataset in the console with `str()` or `glimpse()`.

- **Exercise**

Draw the plot then calculate the correlation between `OBP` and `SLG` for all players in the `mlbBat10` dataset.

```
# Run this and look at the plot
ggplot(data = mlbbat10, aes(x = obp, y = slg)) +
  geom_point()
```

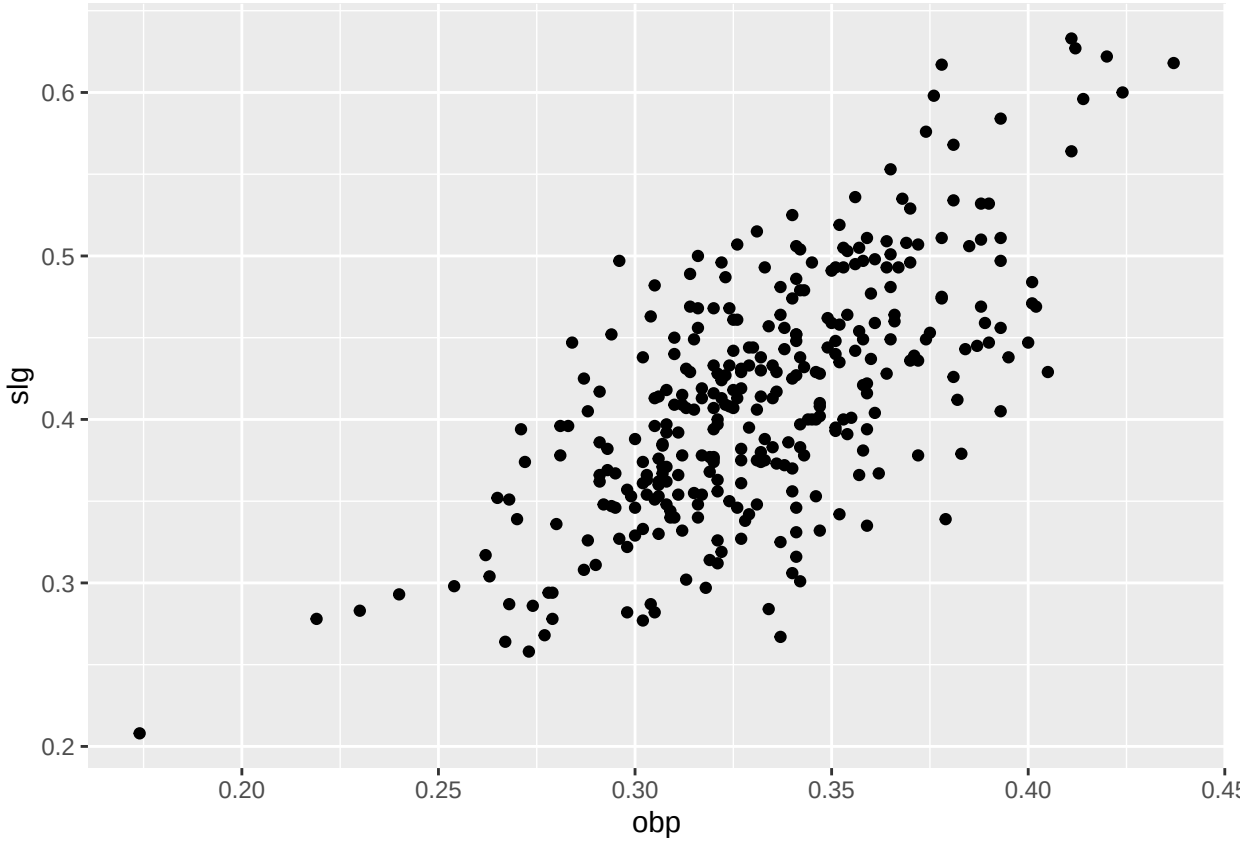



```
# Correlation for all baseball players
mlbbat10 %>%
  summarize(N = n(), r = cor(obp, slg))
```

```
## # A tibble: 1 x 2
##       N     r
##   <int> <dbl>
## 1  1199 0.815
```

Draw the plot then calculate the correlation between `OBP` and `SLG` for all players in the `mlbBat10` dataset with at least 200 at-bats.

```
# Run this and look at the plot
mlbbat10 %>%
    filter(at_bat > 200) %>%
    ggplot(aes(x = obp, y = slg)) +
    geom_point()
```

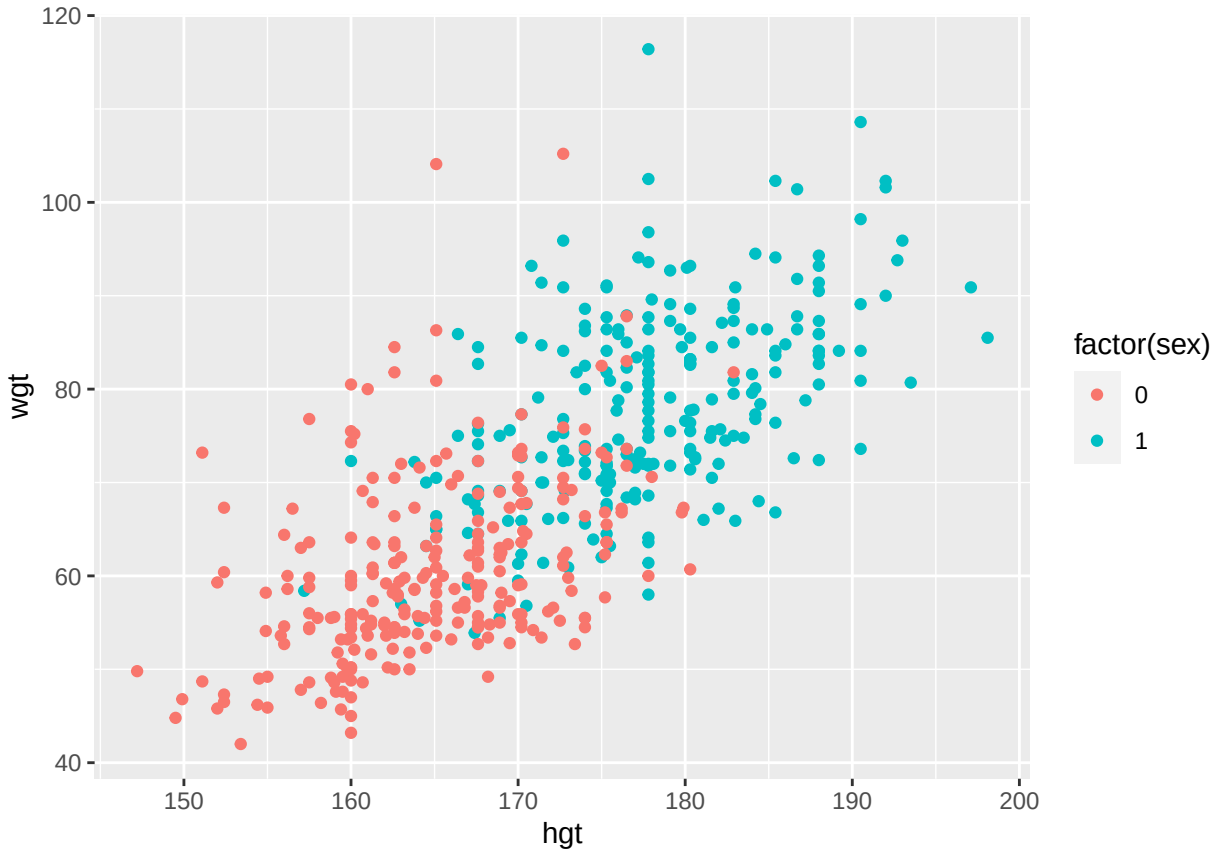



```
# Correlation for all players with at least 200 ABs
mlbbat10 %>%
  filter(at_bat >= 200) %>%
  summarize(N = n(), r = cor(obp, slg))
```

```
## # A tibble: 1 x 2
##       N     r
##   <int> <dbl>
## 1   329 0.686
```

Draw the plot then calculate the correlation between height and weight for each sex in the `bdims` dataset.

```
# Run this and look at the plot
ggplot(data = bdims, aes(x = hgt, y = wgt, color = factor(sex))) +
  geom_point()
```

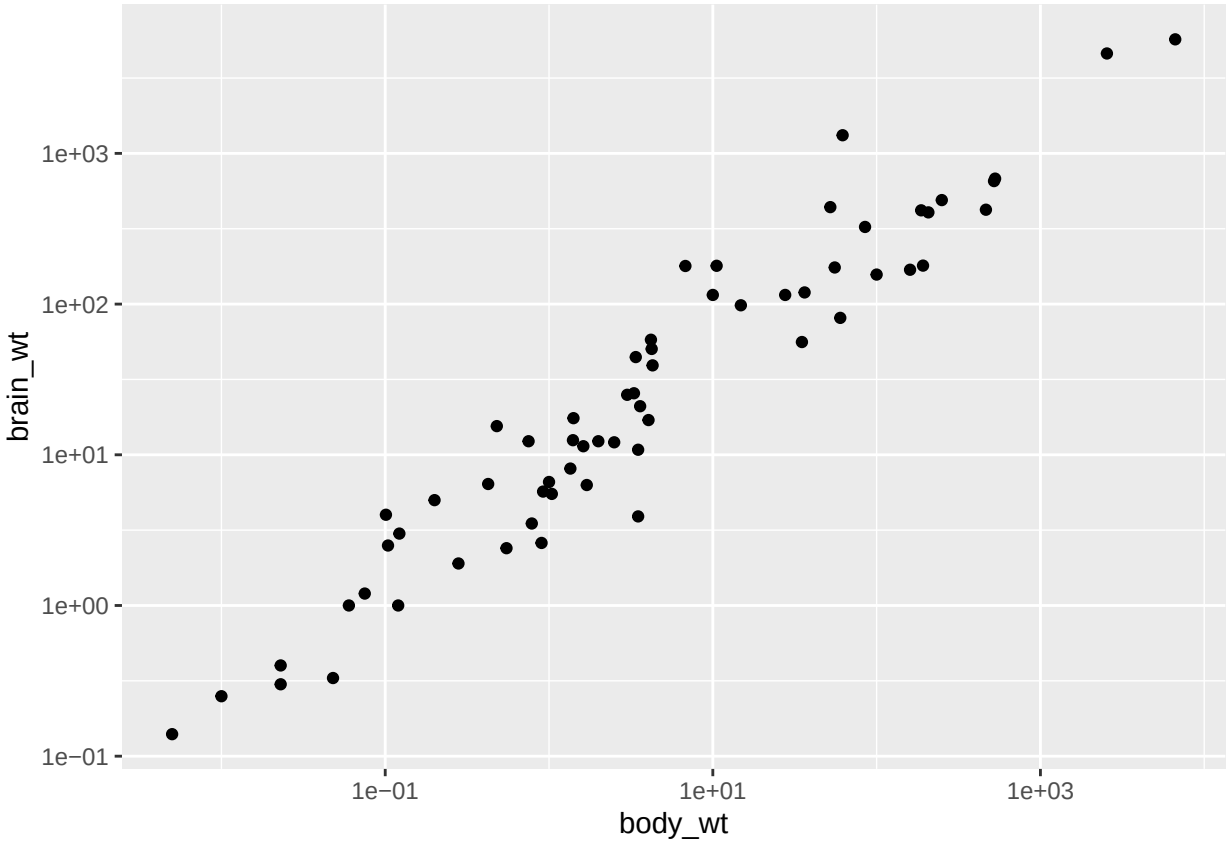



```
# Correlation of body dimensions
bdims %>%
  group_by(sex) %>%
  summarize(N = n(),  r = cor(hgt, wgt))
```

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

Draw the plot then calculate the correlation between 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.

```
# Run this and look at the plot
ggplot(data = mammals, aes(x = body_wt, y = brain_wt)) +
  geom_point() + scale_x_log10() + scale_y_log10()
```




```
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
```

2.7 Interpreting correlation in context

Recall that 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.

- **Possible Answers**
- 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.

2.8 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.

- **Possible Answers**
- 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.

2.9 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.

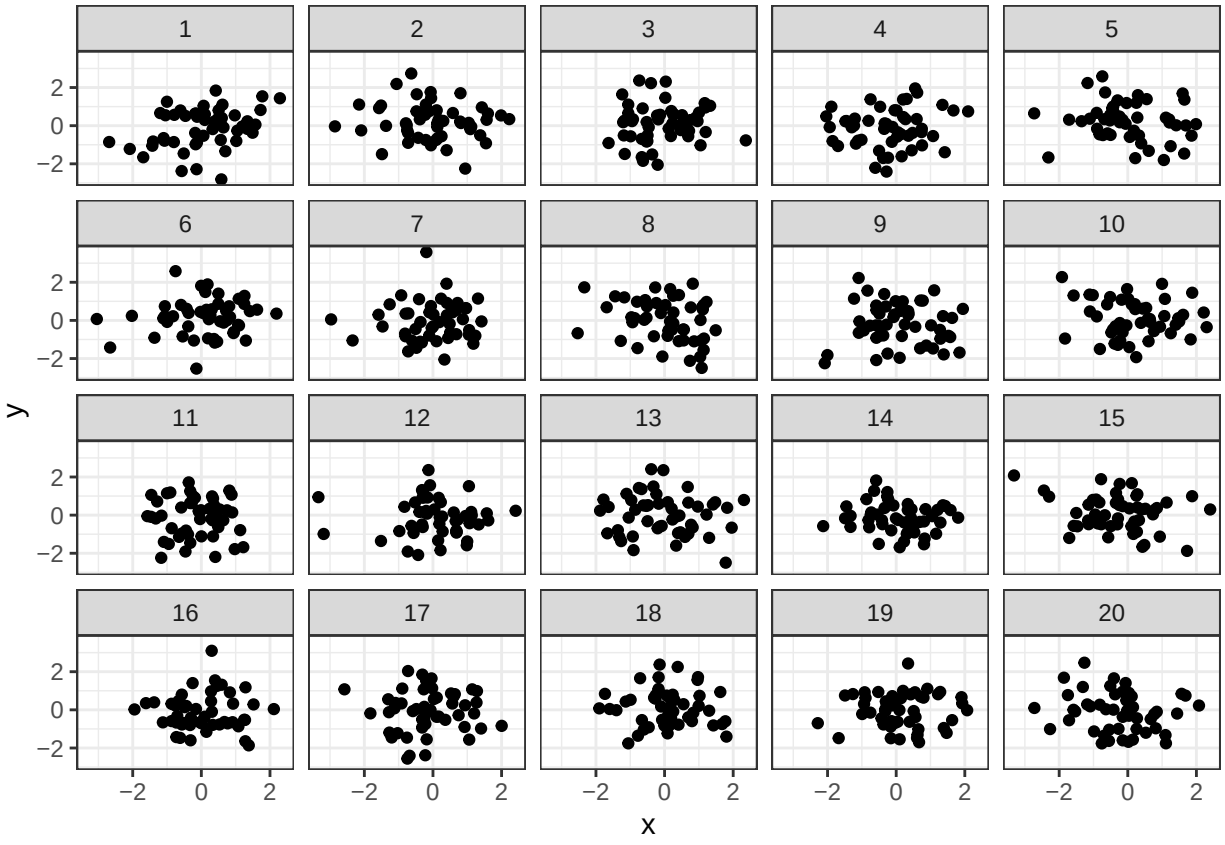
The `noise` dataset contains 20 sets of `x` and `y` variables drawn at random from a standard normal distribution. Each set, denoted as `z`, has 50 observations of `x`, `y` pairs. Do you see any pairs of variables that might be meaningfully correlated? Are all of the correlation coefficients close to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

```
# noise dataset must be created since Datacamp provides the appropriate datasets
noise <- data.frame(x = rnorm(1000), y = rnorm(1000), z = rep(1:20, 50))
```

- **Exercise**

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. Compute the actual correlation between each of the 20 sets of pairs of `x` and `y`. Identify the datasets that show non-trivial correlation of greater than 0.2 in absolute value.

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



```r
# 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      0.331
##  2     2    50     -0.103
##  3     3    50      0.0777
##  4     4    50      0.116
##  5     5    50     -0.143
##  6     6    50      0.111
##  7     7    50      0.0787
##  8     8    50     -0.266
##  9     9    50     -0.0355
## 10    10    50     -0.0968
## 11    11    50     -0.0320
## 12    12    50     -0.00654
## 13    13    50     -0.0915
## 14    14    50     -0.122
## 15    15    50     -0.228
## 16    16    50     -0.0134
## 17    17    50      0.0277
## 18    18    50     -0.0507
## 19    19    50      0.0160
## 20    20    50     -0.152
```

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

```
## # A tibble: 3 x 3
##       z     N spurious_cor
##   <int> <int>        <dbl>
## 1     1    50        0.331
## 2     8    50       -0.266
## 3    15    50       -0.228
```

3 Chapter: Simple Linear Regression

3.1 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"`.

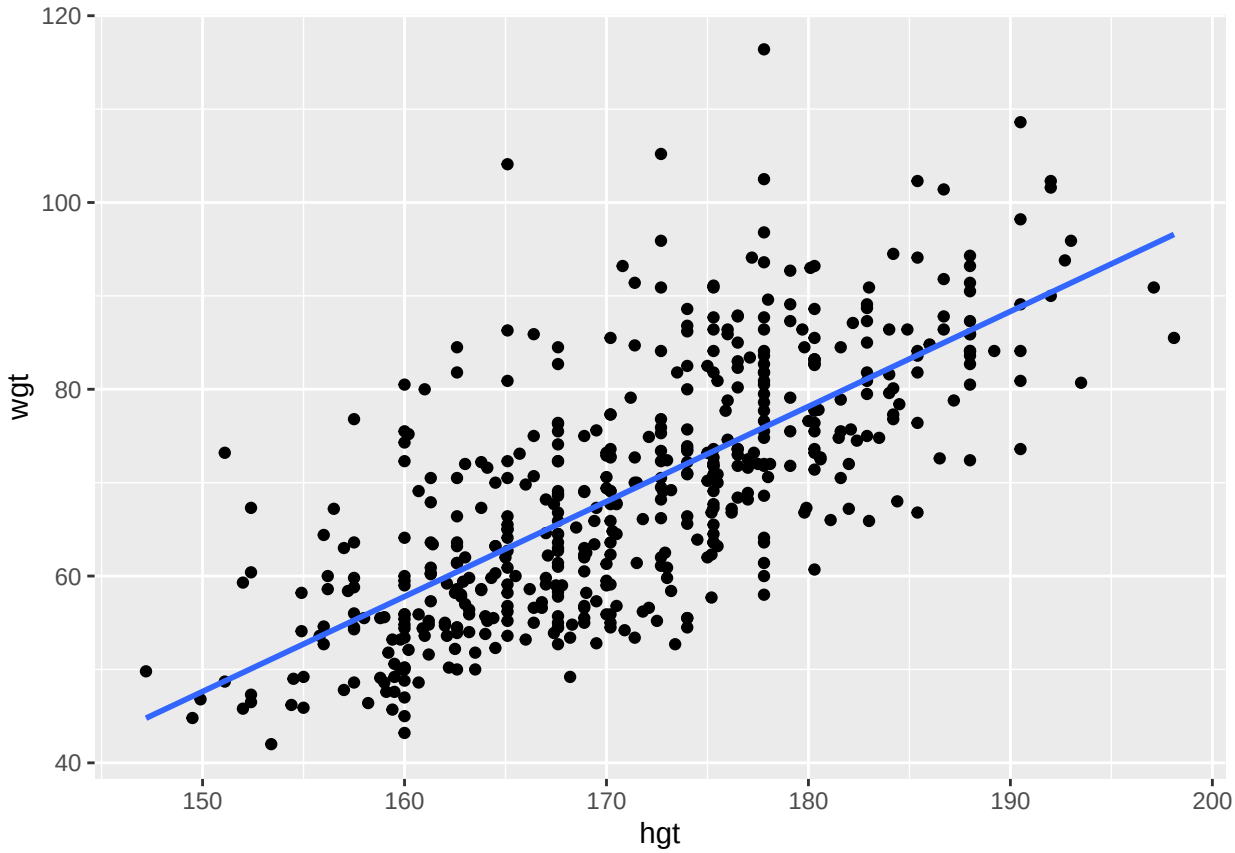
Note that `geom_smooth()` also takes an `se` argument that controls the standard error, which we will ignore for now.

- **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.

```
# Scatterplot with regression line
ggplot(data = bdims, aes(x = hgt, y = wgt)) +
geom_point() +
geom_smooth(method = "lm", se = FALSE)
```

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


120
100
80
60
40
wgt
150
160
170
180
190
200
hgt

3.2 Regression model terminology

Consider a linear regression model of the form:

$$Y = \beta_0 + \beta_1 \cdot X + \epsilon, \text{ where } \epsilon \sim N(0, \sigma_\epsilon).$$

The slope coefficient is:

- **Possible Answers**
  - Y
  - β_0
  - β_1
  - ϵ

3.3 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 _____.

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

3.4 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 and intercept 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 and represent the standard deviation (`sd()`) of X and Y, respectively.

Second, the point (x, y) is always on the least squares regression line, where x and 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
```

- **Exercise**

Print the `bdims_summary` data frame. Use `mutate()` to add the `slope` and `intercept` to the `bdims_summary` data frame.

```
# Print bdims_summary
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.
```

3.5 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.

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

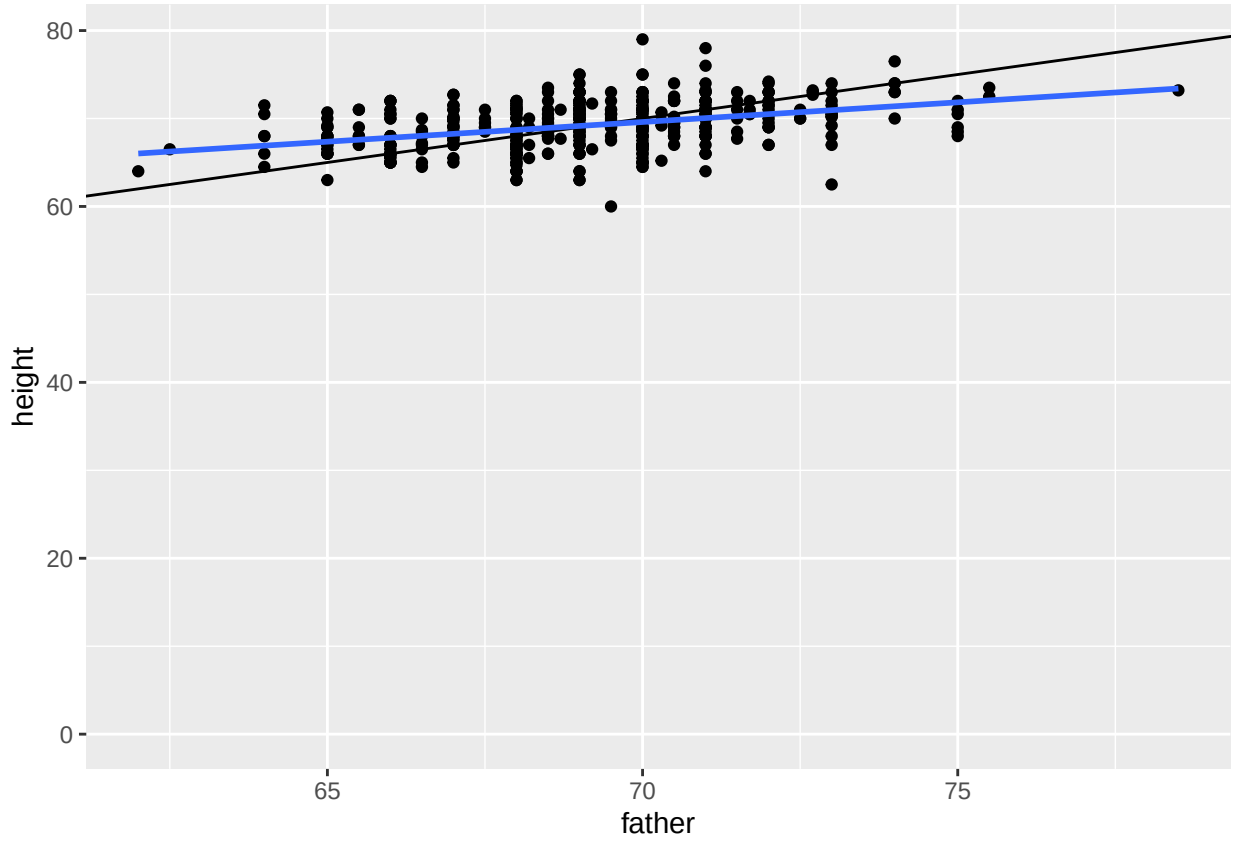
- **Exercise**

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. 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.

```
library(mosaicData)
Galton_men <- Galton %>% filter(sex == "M")
Galton_women <- Galton %>% filter(sex == "F")
```

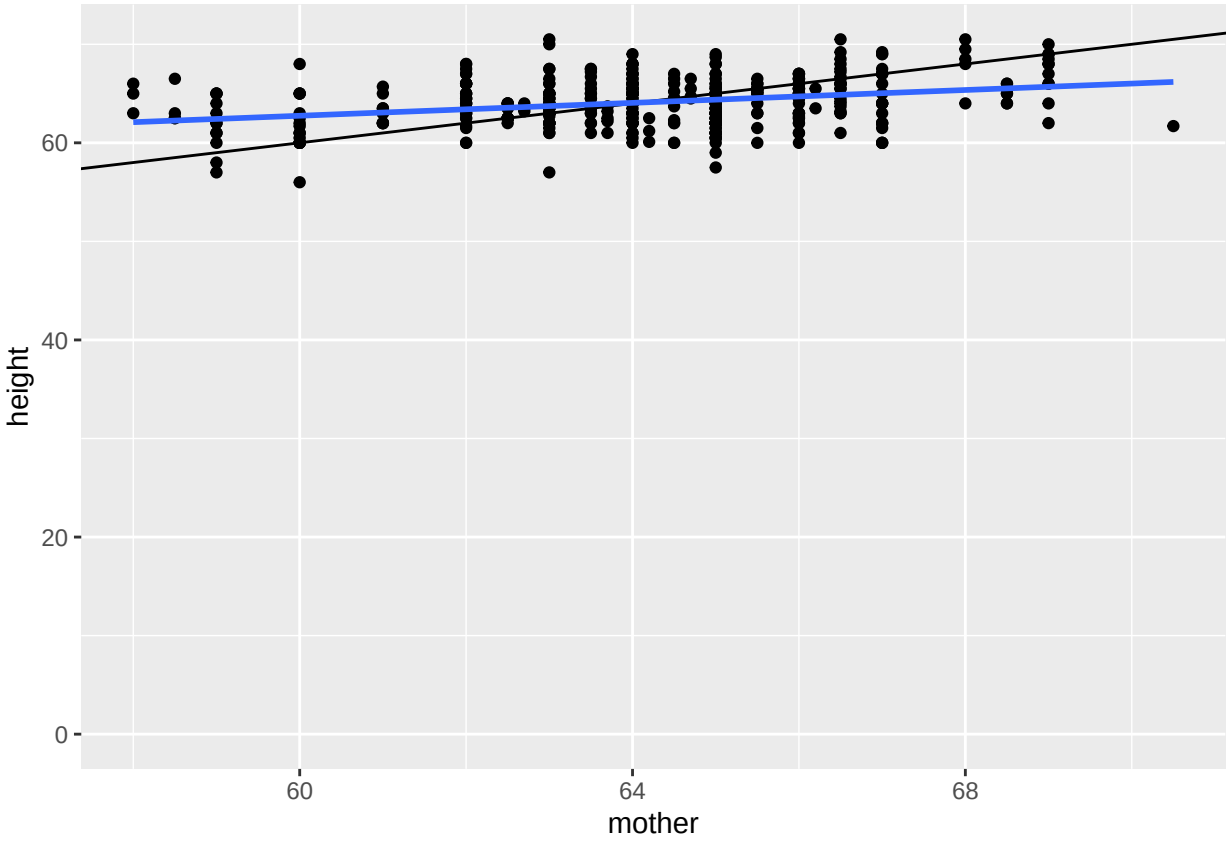
```
# Height of children vs. height of father
ggplot(data = Galton_men, aes(x = father, y = height)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0) +
  geom_smooth(method = "lm", se = FALSE)
```

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




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

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


height
60
40
20
0
60
64
68
mother

3.6 "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...

- **Possible Answers**
  - 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.

4 Chapter: Interpreting Regression Models

4.1 Interpretation of coefficients

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

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

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

- **Possible Answers**
  - 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.

4.2 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?

- **Possible Answers**
  - 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.

4.3 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 use to fit 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.

- **Exercise**

Using the `bdims` dataset, create a linear model for the weight of people as a function of their height. Using the `mlbBat10` dataset, create a linear model for `SLG` as a function of `OBP`. 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.

```
# 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
```

```
# 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
```

```
# 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
```

4.4 Units and scale

In the previous examples, we fit two regression models:

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

and

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

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

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

4.5 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 , adjusted , 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.)

- **Exercise**

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

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

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

```
# Show the full output
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
```

4.6 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 indexes the observations. Recall that:

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

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:

`{r, eval=FALSE} mod <- lm(wgt ~ hgt, data = bdims)`

- **Exercise**

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

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

```
## [1] TRUE
```

```
# Mean of the residuals
mean(residuals(mod))
```

```
## [1] -1.266971e-15
```

4.7 Tidying your linear model

As you fit a regression model, there are some quantities (e.g. R^2) that 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.

```
# Load broom
library(broom)

# Create bdims_tidy
bdims_tidy <- augment(mod)

# 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, 81.6,~
## $ hgt        <dbl> 174.0, 175.3, 193.5, 186.5, 187.2, 181.5, 184.0, 184.5, 175~
## $ .fitted    <dbl> 72.05406, 73.37697, 91.89759, 84.77427, 85.48661, 79.68619,~
## $ .resid     <dbl> -6.4540648, -1.5769666, -11.1975919, -12.1742745, -6.686606~
## $ .hat       <dbl> 0.002154570, 0.002358152, 0.013133942, 0.007238576, 0.00772~
## $ .sigma     <dbl> 9.312824, 9.317005, 9.303732, 9.301360, 9.312471, 9.314716,~
## $ .cooksd    <dbl> 5.201807e-04, 3.400330e-05, 9.758463e-03, 6.282074e-03, 2.0~
## $ .std.resid <dbl> -0.69413418, -0.16961994, -1.21098084, -1.31269063, -0.7211~
```

4.8 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. 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.

```
# ben defined by Datacamp
ben <- data.frame(74.8, 182.8)
names(ben) <- c("wgt", "hgt")
```

- **Exercise**

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

```
# Print ben
ben
```

```
##    wgt   hgt
## 1 74.8 182.8
```

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

```
##        1
## 81.00909
```

4.9 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.

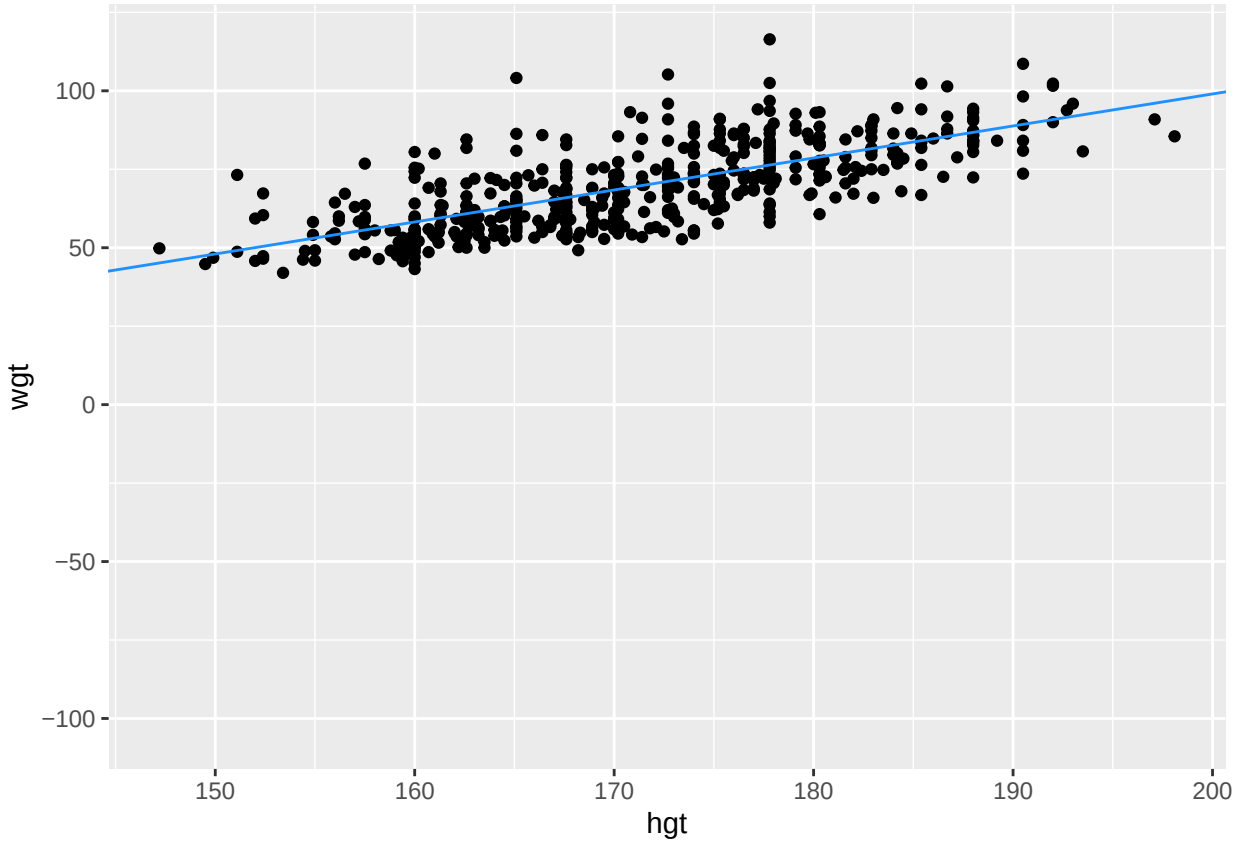
```
# coefs defined by Datacamp
coefs <- data.frame(-105, 1.02)
names(coefs) <- c("(Intercept)", "hgt")
str(coefs)
```

```
## 'data.frame':    1 obs. of  2 variables:
##  $ (Intercept): num -105
##  $ hgt        : num 1.02
```

- **Exercise**

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.

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


100
50
0
−50
−100
wgt
150
160
170
180
190
200
hgt

5 Chapter: Model Fit

5.1 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?

- **Possible Answers**
  - **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.

5.2 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,

$$\sqrt{\frac{\Sigma_i e_i^2}{d.f.}} = \sqrt{\frac{SSE}{d.f.}}$$

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

- **Exercise**

View a `summary()` of mod. Compute the mean of the `residuals()` and verify that it is approximately zero. Use `residuals()` and `df.residual()` to compute the root mean squared error (RMSE), a.k.a. *residual standard error*.

```
# 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
mean(residuals(mod))
```

```
## [1] -1.266971e-15
```

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

```
## [1] 9.30804
```

5.3 Assessing simple linear model fit

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

$$\sqrt{\frac{\Sigma_i e_i^2}{d.f.}} = \sqrt{\frac{SSE}{d.f.}}$$

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.

- **Exercise**

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`.

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

```
# 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
```

```
# Compute R-squared
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
```

5.4 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.

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

How should this result be interpreted?

- **Possible Answers**
  - 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.

5.5 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:

$$\hat{y}_{null} = \bar{y}$$

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 the graphic, 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.

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

```
# Datacamp does not provide mod_null and mod_hgt.
mod_null <- lm(wgt ~ 1, data = bdims) %>%
  augment()

mod_hgt <- mod %>%
  augment()
```

```
glimpse(mod_null)
```

```
## Rows: 507
## Columns: 7
## $ wgt        <dbl> 65.6, 71.8, 80.7, 72.6, 78.8, 74.8, 86.4, 78.4, 62.0, 81.6,~
## $ .fitted    <dbl> 69.14753, 69.14753, 69.14753, 69.14753, 69.14753, 69.14753,~
## $ .resid     <dbl> -3.5475345, 2.6524655, 11.5524655, 3.4524655, 9.6524655, 5.~
## $ .hat       <dbl> 0.001972387, 0.001972387, 0.001972387, 0.001972387, 0.00197~
## $ .sigma     <dbl> 13.35803, 13.35845, 13.34906, 13.35808, 13.35205, 13.35660,~
## $ .cooksd    <dbl> 1.399179e-04, 7.822033e-05, 1.483780e-03, 1.325192e-04, 1.0~
## $ .std.resid <dbl> -0.26607983, 0.19894594, 0.86648293, 0.25894926, 0.72397503~
```

```
glimpse(mod_hgt)
```

```
## Rows: 507
## Columns: 8
## $ wgt        <dbl> 65.6, 71.8, 80.7, 72.6, 78.8, 74.8, 86.4, 78.4, 62.0, 81.6,~
## $ hgt        <dbl> 174.0, 175.3, 193.5, 186.5, 187.2, 181.5, 184.0, 184.5, 175~
## $ .fitted    <dbl> 72.05406, 73.37697, 91.89759, 84.77427, 85.48661, 79.68619,~
## $ .resid     <dbl> -6.4540648, -1.5769666, -11.1975919, -12.1742745, -6.686606~
## $ .hat       <dbl> 0.002154570, 0.002358152, 0.013133942, 0.007238576, 0.00772~
## $ .sigma     <dbl> 9.312824, 9.317005, 9.303732, 9.301360, 9.312471, 9.314716,~
## $ .cooksd    <dbl> 5.201807e-04, 3.400330e-05, 9.758463e-03, 6.282074e-03, 2.0~
## $ .std.resid <dbl> -0.69413418, -0.16961994, -1.21098084, -1.31269063, -0.7211~
```

- **Exercise**

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

```
# Compute SSE for null model
mod_null %>%
  summarize(SSE = sum(.resid^2))
```

```
## # A tibble: 1 x 1
##       SSE
##     <dbl>
## 1 90123.
```

```
# Compute SSE for regression model
mod_hgt %>%
  summarize(SSE = sum(.resid^2))
```

```
## # A tibble: 1 x 1
##       SSE
##     <dbl>
## 1 43753.
```

5.6 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.

- **Exercise**

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

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

```
## # A tibble: 6 x 8
##     wgt   hgt .fitted .resid   .hat .sigma  .cooksd .std.resid
##   <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>    <dbl>      <dbl>
## 1  85.5  198.    96.6 -11.1  0.0182   9.30 0.0134       -1.20
## 2  90.9  197.    95.6  -4.66 0.0170   9.31 0.00221      -0.505
## 3  49.8  147.    44.8   5.02 0.0148   9.31 0.00221       0.543
## 4  80.7  194.    91.9 -11.2  0.0131   9.30 0.00976      -1.21
## 5  95.9  193     91.4   4.51 0.0126   9.32 0.00152       0.488
## 6  44.8  150.    47.1  -2.32 0.0124   9.32 0.000397     -0.251
```

5.7 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 (y), the residual depends on the response variable (x) 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.

```
mod %>% augment() %>% arrange(desc(.cooksd)) %>% head()
```

```
## # A tibble: 6 x 8
##     wgt   hgt .fitted .resid    .hat .sigma .cooksd .std.resid
##   <dbl> <dbl>   <dbl>  <dbl>   <dbl>  <dbl>   <dbl>      <dbl>
## 1  73.2  151.    48.8   24.4 0.0109    9.25  0.0386       2.64
## 2 116.   178.    75.9   40.5 0.00296   9.14  0.0282       4.36
## 3 104.   165.    63.0   41.1 0.00279   9.14  0.0273       4.42
## 4 109.   190.    88.8   19.8 0.0103    9.28  0.0238       2.13
## 5  67.3  152.    50.1   17.2 0.00982   9.29  0.0171       1.86
## 6  76.8  158.    55.3   21.5 0.00613   9.27  0.0166       2.32
```

5.8 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.

- **Exercise**

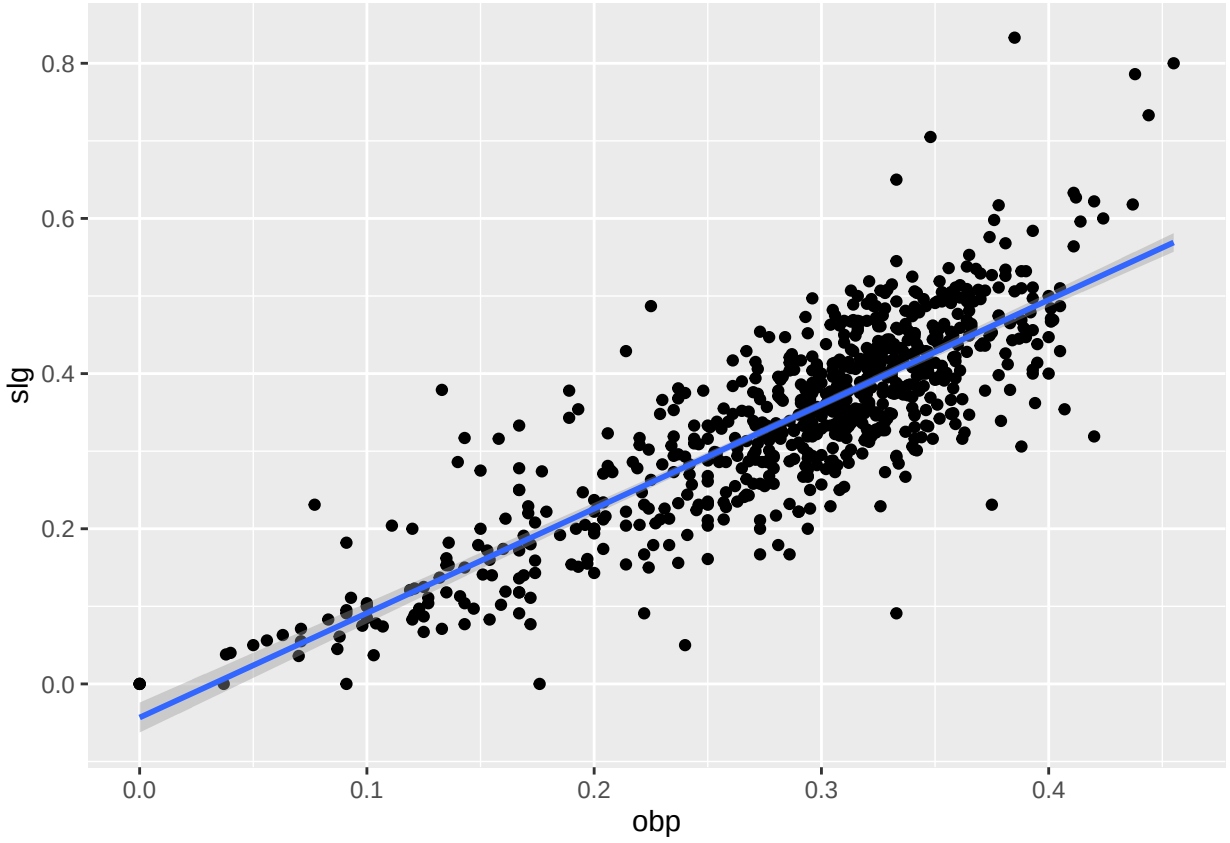
Use `filter()` to create a subset of `mlbBat10` called `nontrivial_players` consisting of only those players with at least 10 at-bats and OBP of below 0.500. Fit the linear model for `SLG` as a function of `OBP` for the `nontrivial_players`. Save the result as `mod_cleaner`. View the `summary()` of the new model and compare the slope and R^2 to those of `mod`, the original model fit to the data on all players. Visualize the new model with `ggplot()` and the appropriate `geom_*()` functions.

```
# Create nontrivial_players
nontrivial_players <- mlbbat10 %>%
filter(at_bat >= 10, obp < 0.5)

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

```
# Visualize new model
ggplot(data = nontrivial_players, aes(x = obp, y = slg)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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



5.9 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.

- **Exercise**

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
##     wgt   hgt .fitted .resid   .hat .sigma  .cooksd .std.resid
##   <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>    <dbl>      <dbl>
## 1  85.5  198.    96.6 -11.1  0.0182   9.30 0.0134       -1.20
## 2  90.9  197.    95.6  -4.66 0.0170   9.31 0.00221      -0.505
## 3  49.8  147.    44.8   5.02 0.0148   9.31 0.00221       0.543
## 4  80.7  194.    91.9 -11.2  0.0131   9.30 0.00976      -1.21
## 5  95.9  193     91.4   4.51 0.0126   9.32 0.00152       0.488
## 6  44.8  150.    47.1  -2.32 0.0124   9.32 0.000397     -0.251
```

6 Chapter: Conclusion

6.1 Conclusion

This course was about analyzing the relationship between two numeric variables. We learned a variety of techniques for doing this.

6.2 Graphical: scatterplots

First, we explored how powerful scatter plots can be in revealing bivariate relationships in an intuitive, graphical form. We built a framework for describing what we see in those scatter plots and practiced implementing that framework on real data.

6.3 Numerical: correlation

Second, we learned about correlation, a simple way to quantify the strength of the linear relationship between two variables using a single number. We emphasized the value of such measurements,

6.4 Numerical: correlation

but illustrated how their careless application can lead to erroneous results.

6.5 Modular: linear regression

Third, we learned about linear regression a relatively simple, yet powerful technique for modeling a response variable in terms of a single explanatory variable. We built our intuition about how these models work and identified some of their key properties.

6.6 Focus on interpretation

Fourth, we focused carefully on how to interpret the coefficients of regression models and how those interpretations can bring real insight into complex problems.

6.7 Objects and formulas

We also developed a foundational understanding of how to build these models in R and how to work with them afterwards.

6.8 Model fit

Finally, we introduced the notion of model fit and developed tools for helping us reason about the quality of our models and how much we can learn from them. Together, we hope that these concepts and techniques will inform your thinking about the nature of the relationship between variables and help to you unravel them on your own.