A Tidy Introduction To Statistical Learning

Beau Lucas

2019-10-18

Contents

Preface

This book will serve as a source of notes and exercise solutions for *An Introduction to Statistical Learning*. My approach will be centered around the `tidyverse`. This is not a replacement for the book, which should be read front to back by all machine learning enthusiasts.

Chapter names will line up, and certain subheadings will also match. Sometimes my notes will contain text lifted straight from the book without modification. This is not an attempt to plagiarize or claim their writing as my own. My goal is for this bookdown project to be a quick stop for machine learning enthusiasts to reference high-level ideas from ISLR in a modern media format.

Chapter 1

Introduction

1.1 An Overview of Statistical Learning

Statistical learning is focused on supervised and unsupervised modeling and prediction.

1.2 Data Sets Used in Labs and Exercises

All data sets used in this book can be found in `ISLR` and `MASS` packages, with some also being found in the base `R` distribution.

We will utilize the `tidyverse` ecosystem to tackle the exercises and labs, as the `R` code found in the original textbook is outdated.

```
library(ISLR)
library(MASS)
library(tidyverse)
library(knitr)
library(kableExtra)
library(modelr)
library(broom)
```

1.3 Book Website

The website and free PDF for the book can be found here:

www.statlearning.com

And here are the YouTube lectures an R labs:

Youtube - Statistical Learning

Chapter 2

Statistical Learning

2.1 What is Statistical Learning?

Methods to estimate functions that connect inputs to outputs.

If there exists a quantitative response variable Y and p different predictors (X_1, X_2, ..., X_p), we can write this relationship as:

$Y = f(X) +$

2.1.1 Why Estimate f?

2.1.1.1 Prediction

We can predict Y using:

$\hat{Y} = \hat{f}(X)$

Accuracy of Y is dependant on:

- *reducible error*
  - $\hat{f}$ will never be perfect estimate of f, and model can always be potentially improved
  - Even if $\hat{f} = f$, prediction would still have some error
- *irreducible error*
  - Because Y is also a function of random , there will always be variability
  - We cannot reduce the error introduced by

2.1.1.2 Inference

How does Y respond to changes in $X_1, X_2, ..., X_p$?

2.1.2 How do we estimate f?

- Use *training data* to train method
- x_ij is value of jth predictor for observation i, y_i is value of response variable
  - $i = 1, 2, ..., n$, $j = 1, 2, ..., p$
- Using training data, apply statistical learning method estimate unknown function f
- Most statistical learning methods can be characterized as either *parametric* or *non-parametric*

2.1.2.1 Parametric Methods

Two-step model-based approach:

1. Make an assumption about functional form of f, such as "f is linear in X"
2. Perform procedure that uses training data to train the model * In case of linear model, this procedure estimates parameters $_0, _1, ..., _p$ * Most common approach to fit linear model is *(ordinary) least squares*

This is *parametric*, as it reduces the problem of estimating f down to one of estimating a set of parameters. Problems that can arise:

- Model will not match the true unknown form of f
- If model is made more *flexible*, which generally requires estimating a greater number of parameters, *overfitting* can occur

2.1.2.2 Non-parametric Methods

Non-parametric methods do not make assumptions about the form of f. An advantage of this is that they have the potential to fit a wider range of possible shapes for f. A disadvantage is that, because there are no assumptions about the form of f, the problem of estimating f is not reduced to a set number of parameters. This means more observations are needed compared to a parametric approach to estimate f accurately.

2.1.3 The Trade-Off Between Prediction Accuracy and Model Interpretability

Restrictive models are much more intepretable than flexible ones. Flexible approaches can be so complicated that it is hard to understand how predictors affect the response.

If inference is the goal, simple and inflexible methods are easier to interpret. For prediction, accuracy is the biggest concern. However, flexible models are more prone to overfitting.

2.1.4 Supervised Versus Unsupervised Learning

Most machine learning methods can be split into *supervised* or *unsupervised* categories. Most of this textbook involves supervised learning methods, in which a model that captures the relationship between predictors and response measurements is fitted. The goal is to accurately predict the response variables for future observations, or to understand the relationship between the predictors and response.

Unsupervised learning takes place when we have a set of observations and a vector of measurements x_i, but no response y_i. We can examine the relationship between the variables or between the observations. A popular method of unsupervised learning is cluster analysis, in which observations are grouped into distinct groups based on their vector of measurements x_i. An example of this would be a company segmenting survey respondents based on demographic data, in which the goal is to ascertain some idea about potential spending habits without possessing this data.

Clustering has some drawbacks. It works best when the groups are significantly distinct from each other. In reality, it is rare for data to exhibit this characteristic. There is often overlap between observations in different groups, and clustering will inevitably place a number of observations in the wrong groups. Further more, visualization of clusters breaks down as the dimensionality of data increases. Most data contains at least several, if not dozens, of variables.

It is not always clear-cut whether a problem should be handled with supervised or unsupervised learning. There are some scenarios where only a subset of the observations have response measurements. This is a *semi-supervised learning* problem, in which a statistical learning method that can utilize all observations is needed.

2.1.5 Regression Versus Classification Problems

Variables can be categorized as either *quantitative* or *qualitative*. Both qualitative and quantatitive predictors can be used to predict both types of response

variables. The more important part of choosing an appropriate statistical learning method is the type of the response variable.

2.2 Assessing Model Accuracy

Every data set is different and there is no one statistical learning method that works best for all data sets. It is important for any given data set to find the statistical learning method that produces the best results. This section presents some concepts that are part of that decision-making process.

2.2.1 Measuring the Quality of Fit

We need to be able to quantify how well a model's predictions match the observed data. How close are the model's predicted response values to the true response values?

In regression, *mean squared error (MSE)* is the most commonly-used measure. A small MSE indicates the predicted responses are very close to the true ones. MSE used on training data is more accurately referred to as the *training MSE*.

We are most concerned with the accuracy of the predictions when we apply our methods to **previously unseen data**. If you are trying to predict the value of a stock, your concern is how it performs in the future, not on known data from the past. Thus, the goal is then minimizing the *test MSE*, which measures the accuracy of a model on **observations that were not used to train the model**. Imagine a set of observations (x_0, y_0) that were not used to train the statistical learning method.

$Ave(y_0 - \hat{f}(x_0))2$

The goal is to select the model that minimizes the test MSE shown above. How can we do this?

Sometimes, there is an available test data set full of observations that were not used in training the model. The test MSE can be evaluated on these observations, and the learning method which produces the smallest TSE will be chosen. If no test observations are available, picking the method that minimizes the training MSE might seem to be a good idea. However, there is no guarantee that a model with the lowest training MSE also has the lowest test MSE. Models often work in minimizing the training MSE, and can end up with large test MSE.

There is a tradeoff in model flexibility, training MSE, and test MSE. A model that is too flexible can closely match the training data, but perform poorly on

the test data. There is a sweet spot to find between model flexibility, training MSE, and test MSE that varies for each unique data set.

Degrees of freedom is a quantity that summarizes the flexibility of a curve, discused more fully in Chapter 7. The more inflexible a model is, the fewer degrees of freedom.

As model flexibility increases, training MSE will inevitably decrease, but test MSE may plateau or even rise. A model with a small training MSE and large test MSE is *overfitting the data*, picking up patterns on the training data that don't exist in the test data. Since we expect the training MSE to almost always be lower than the test MSE, overfitting is a specific case when there exists a less flexible model with a smaller test MSE.

2.2.2 The Bias-Variance Trade-Off

The expected test MSE can be broken down into the sum of three quantities:

1. the *variance* of $\hat{f}(x_0)$
2. the squared *bias* of $\hat{f}(x_0)$
3. the variance of the error terms

$$E(y_0 - \hat{f}(x_0)^2 = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var()$$

The formula above defines the *expected test MSE*, which can be thought of the average test MSE that would be obtained if we repeatedly estimated f and tested each at x_0. To minimize expected test MSE, we need to choose a statistical learning method that achieves both low variance and low bias. Since variance and squared bias are nonnegative, the expected test MSE can never be lower than $Var()$, the irreducible error.

Variance refers to how much $\hat{f}$ would change if repeatedly estimated with different training data sets. Methods with high variance can produce large changes in $\hat{f}$ through small changes in the training data. Generally, the more flexible a model it is, the higher the variance. Following the observations so closely can cause changes in just a single observation of the training data to result in significant changes to $\hat{f}$. More inflexible models, such as linear regression, are less susceptible to the effects of changing a single observation.

Bias is the error introduced from approximating a complicated problem by a much simpler model. Fitting a linear regression to data that is not linear will always lead to high bias, no matter how many observations are in the training set. More flexible models tend to result in less bias.

More flexible methods lead to higher variance and lower bias. The rate of change between the quantities determines at which point the test MSE is minimized. Bias tends to decrease at a faster rate in the beginning, causing the test MSE to decline. However, when flexibility reaches a certain point, variance will begin to increase faster than bias is decreasing, causing test MSE to rise.

This relationship between bias, variance, and test MSE is known as the *bias-variance tradeoff*. Here is a good article on it: Understanding the Bias-Variance Tradeoff

In real-life scenarios where f is unknown, we cannot explicitly compute the test MSE, bias, or variance. However, there are methods to estimate this, such as *cross-validation*, which will be discussed in Chapter 5.

2.2.3 The Classification Setting

For classification problems where $y_i, ..., y_n$ are qualitative, we can quantify the accuracy of our estimate by using the *training error rate*, the proportion of mistakes that are made when applying our model $\hat{f}$ to the training observations.

$1/n \sum_{i=1}^{n} I(y_i \neq \hat{y_i})$

training error rate

Breaking the formula above down.

- $\hat{y_i}$ is the predicted class label for the ith observation using $\hat{f}$
- $I(y_i \neq \hat{y_i})$ is an *indicator variable* that equals 1 if $y_i \neq \hat{y_i}$, and 0 if $y_i = \hat{y_i}$
- If $I(y_i \neq \hat{y_i}) = 0$, then the ith observation was classified correctly

Similar to our regression problems, we are more interested in the model's performance on test observations not used in training. The formula below gives us the *test error rate* for a set of observations of the form (x_0, y_0).

$Ave(I(y_0 \neq \hat{y_0}))$

test error rate

A good classifier will minimize the above.

2.2.3.1 The Bayes Classifier

The *test error rate* is minimized by the classifier that assigns each observation to the most likely class, given its predictor values. Our decision is then based on finding the value at which the formula below is largest.

$Pr(Y = j|X = x_0)$

If the response values are binomial (let's call them A and B) the classifier simplifies to:

$Pr(Y = A|X = x_0) > 0.5 \, then \, A, \, else \, B$

The *Bayes decision boundary* is the point where the probabilities are equal for both groups. Points on either side of this line are assigned to the group predicted

by the classifier. The *Bayes error rate* averaged over all possible values of X is below.

$1 - E(max_j Pr(Y = j|X))$

Bayes error rate

The *Bayes error rate* is often greater than zero, as observations between classes overlap in real-world data.

2.2.3.2 K-Nearest Neighbors

Since the true conditional distribution of Y given X cannot be known in real data, the Bayes classifier is used as a "gold standard" to compare other models to. Many methods attempt to estimate this conditional distribution, and then classify an observation based on the estimated probability. A common method is *K-nearest neighbors (KNN)*. Given a positive integer K and a test observation x_0, KNN then does the following:

1. identifies the K points in the training data that are closest to x_0, represented by N_0
2. estimates conditional probability for class j as the fraction of the points in N_0 whose response values equal j:

$Pr(Y = j|X = x_0) = 1/K \sum_{i \in N_0} I(y_i = j)$

3. applies Bayes rule and classifies test observation x_0 to class with largest probability

KNN can be surprisingly robust to the optimal Bayes classifier. The choice in K makes a huge difference. For example, a $K = 1$ is highly flexible, classifying observations based off of the closest nearby training observation. $K = 100$ would do the opposite, basing its classification off a large pool of training observations compared to the $K = 1$ version. The higher K value produces a more linear model. The trade-off between flexibility, training error rate, and test error rate applies to both classification and regression problems.

2.3 Lab: Introduction to R

Finally we get to some `R` code. This chapter of ISLR introduces basic `R` syntax, and most of it is unchanged in my version. This should all be familiar to anyone who has used R before.

We are going to be working with `tibbles` as our primary data structure throughout this book. Please read here: tibbles

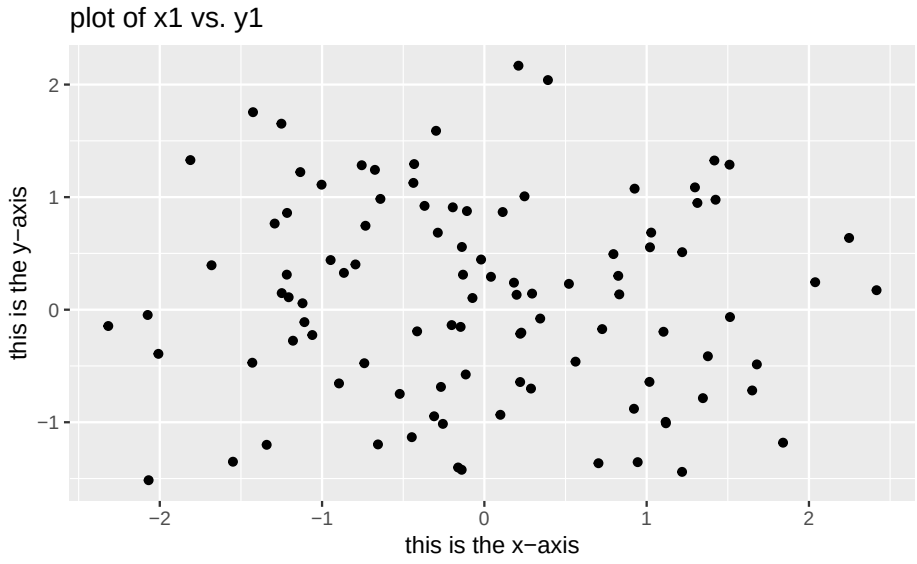
2.3.0.1 Basic Commands

Skipping this.

2.3.0.2 Graphics (Plotting)

Here we begin to explore the "tidy" approach to R. We will abstain from base R plotting and use `ggplot2`, which is a more powerful tool. Let's plot a scatterplot with some basic labels.

```
tbl_rnorm <- tibble(
  x1 = rnorm(100),
  y1 = rnorm(100)
)

ggplot(tbl_rnorm, aes(x = x1, y = y1)) +
  geom_point() +
  labs(title = "plot of x1 vs. y1",
       x = "this is the x-axis",
       y = "this is the y-axis")
```



2.3.0.3 Indexing data

We will skip this.

2.3.0.4 Loading data

ISLR mentions insuring proper working directory before loading data. Dealing with working directories in R is a bad idea. Fortunately, it's easily avoidable through the use of RStudio *projects*, which keep all files used in analysis together and make your work more robust and reproducible. See the RStudio Projects chapter in *r4ds* for more information.

We will opt for the `readr` (part of the `tidyverse`) package instead of base R. Take a look at this subsection of *r4ds* for reasons why: *11.2.1 Compared to base R*

Below is a reproducible example in which we create a tibble, save it as a .txt file, and then read it in with `write_tsv()`. The set of `read_*` functions in `readr` will be the standrad way to read local files into R. If you are using RStudio projects, there is no need to worry about working directories.

```
# generate dummy data to read in
generic_company_tibble <- tibble(
  x = rnorm(100, mean = 25),
  y = rnorm(100, mean = 50),
  z = sample(c("apple", "uber", "facebook", "twitter", "tesla", "google", "microsoft"), 1)
)

tmp <- tempfile()
write_tsv(generic_company_tibble, tmp)
company_data <- read_tsv(tmp)
```

`readr` provides a nice summary of the imported tibble. Calling the tibble by name will also give a breakdown of column names, data types, and number of observations.

```
company_data
```

```
## # A tibble: 100 x 3
##       x     y z
##   <dbl> <dbl> <chr>
## 1  23.9  50.5 uber
## 2  25.3  49.2 uber
## 3  24.6  48.6 uber
## 4  24.9  51.0 uber
## 5  26.4  49.1 uber
## 6  24.8  50.7 uber
## # ... with 94 more rows
```

2.3.0.5 Additional Graphical and Numerical Summaries

ISLR mentions the `attach()` function, which allows R to reference column names of dataframes without specifying the dataframe. `attach` can lead to confusion and errors when working on a project with multiple sources of data. This is a bad practice, and should always be avoided.

The book then goes into some explanation of `plot()`, which we will not be using.

2.4 Exercises

1. For each of parts (a) through (d), indicate whether we would generally expect the performance of a flexible statistical learning method to be better or worse than an inflexible method.
   (a) The sample size n is extremely large, and the number of predictors p is small
      - **(better)** given large sample size, a flexible model would be able to capture a trend without being influenced too heavily by a small number of observations.
   (b) The number of predictors p is extremely large, and the number of observations n is small.
      - **(worse)** given the small sample size, an inflexible model would do better at not overfitting to a small number of observations (capturing patterns in the training data that dont really exist)
   (c) The relationship between the predictors and response is highly non-linear.
      - **(better)** highly flexible methods are highly non-linear and can produce better fits on non-linear data compared to inflexible methods such as linear regression
   (d) The variance of the error terms, i.e. 2 = Var( ), is extremely high.
      - **(worse)** given the high variance in the data, an inflexible method would overfit to the noise
2. Explain whether each scenario is a classification or regression problem, and indicate whether we are most interested in inference or prediction. Finally, provide n and p.

(a) We collect a set of data on the top 500 firms in the US. For each firm we record profit, number of employees, industry and the CEO salary. We are interested in understanding which factors affect CEO salary.
   - **(regression; inference)** This is a regression problem with both qualitative and quantitative predictors. Inference is the main goal, as the company probably wants a model that is human-readable in

order to understand what determines a CEO's salary.
  - `n = 500, p = 3`

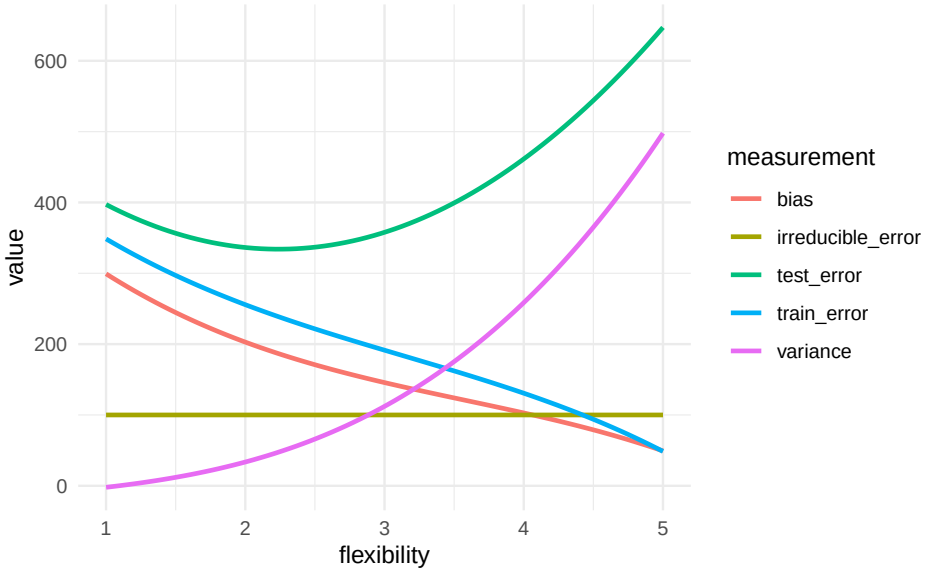
(b) We are considering launching a new product and wish to know whether it will be a success or a failure. We collect data on 20 similar products that were previously launched. For each product we have recorded whether it was a success or failure, price charged for the product, marketing budget, competition price, and ten other variables.
  - **(classification; prediction)** The goal is to classify whether a product will be a success or failure. Prediction is the goal, as they want to accurately determine if their product will succeed or fail.
    - `n = 20, p = 4`

(c) We are interested in predicting the % change in the USD/Euro exchange rate in relation to the weekly changes in the world stock markets. Hence we collect weekly data for all of 2012. For each week we record the % change in the USD/Euro, the % change in the US market, the % change in the British market, and the % change in the German market.
  - **(regression; prediction)** The goal is to predict the % change of the exchange rate.

3. We now revisit the bias-variance decomposition

(a) Provide a sketch of typical (squared) bias, variance, training error, test error, and Bayes (or irreducible) error curves, on a single plot, as we go from less flexible statistical learning methods towards more flexible approaches. The x-axis should represent the amount of flexibility in the method, and the y-axis should represent the values for each curve. There should be five curves. Make sure to label each one.

```
bias_variance <- tibble(
  flexibility = c(1:5),
  bias = c(300,200,150,100,50),
  variance = c(0,25,125,250,500),
  train_error = c(350,250,200, 125, 50),
  irreducible_error = 100,
  test_error = variance + bias + irreducible_error) %>%
  gather(`bias`, `variance`, `train_error`, `irreducible_error`, `test_error`,
         key = "measurement", value = "value")

ggplot(bias_variance, aes(x = flexibility, y = value, colour = measurement)) +
  geom_smooth(se = FALSE, method = "lm", formula = y ~ poly(x,3)) +
  theme_minimal()
```



4. You will now think of some real-life applications for statistical learning.
   (a) Describe three real-life applications in which classification might be useful. Describe the response, as well as the predictors. Is the goal of each application inference or prediction? Explain your answer.
      1. predicting diabetes.
         - response: future diabetes
         - predictors: health and body measurements of patient
         - goal: prediction, model complexity and human understanding is not important
      2. demographics that determine future education level
         - response: education level
         - predictors: demographic data
         - goal: inference, prediction is important here too but researchers would probably want to understand and share which factors determine the response in order to raise awareness
      3. faulty parts in manufacturing
         - response: whether or not part is faulty
         - predictors: various tests on part
         - goal: prediction, it is most important to have an accurate model, especially if faulty parts can lead to deaths
   (b) Describe three real-life applications in which regression might be useful. Describe the response, as well as the predictors. Is the goal of each application inference or prediction? Explain your answer.
      1. number of riders on public transit over time
         - response: how many riders are expected to use public transit
         - predictors: current transit usage data, local population data,

etc.
- goal: prediction, it is important to prepare for growth in transit usage so governments have enough time to make necessary changes

2. demand for product
   - response: how many units to expect to be sold
   - predictors: current demand data, revenue growth, business expansion, changing in market trends, economy health
   - goal: prediction, figure out in X amount of time demand for product in order to upsize/downsize to appropriate level
3. determine future salary
   - response: future expeceted salary
   - predictors: employment history, education, geolocation, etc.
   - goal: inference, researchers might want to know the most important factors that lead to higher salaries rather than a model that is too complex to understand

5. What are the advantages and disadvantages of a very flexible (versus a less flexible) approach for regression or classification? Under what circumstances might a more flexible approach be preferred to a less flexible approach? When might a less flexible approach be preferred?

- Very flexible approach allows you to fit a more flexible function to the data. The advantage is that you have the potential to accurately predict data even as it moves away from linearity. The disadvantage are potential overfitting to the training data, increasing variance (individual observations affect the model to higher degree than non-flexible counterpart), and higher computational costs (as well as less human-readable explanations)
- When a more flexible approach is preferred: data that is non-linear, large sample size, prediction more important than inference
- When a less flexible approach is preferred: data that is more linear, smaller sample size, inference more important than prediction

6. Describe the differences between a parametric and a non-parametric statistical learning approach. What are the advantages of a parametric approach to regression or classification (as opposed to a nonparametric approach)? What are its disadvantages?

- A parametric approach assumes a form of f and has to estimate an often known number of parameters (for example, linear regression simply requires estimating $p + 1$ coefficients)
- A non-parametric approach makes no assumptions about the true form of f. They simply want to get as close as possible to f. This allows them to take on a larger variety of shapes, and accomodate a larger variety of patterns.
- Advantages of a parametric approach are that they take less observations to generate (the problem is reduced to applying a known form to f, such

as linear regression), are less inclined to overfit, and generally less computationally intensive.
- Disadvantages of a parametric approach making assumptions about the form of f, which may not match the real form and could lead to a model that doesn't fit the data well

7. The table below provides a training data set containing six observa- tions, three predictors, and one qualitative response variable.

```
training_set <- tibble(
  x1 = c(0,2,0,0,-1,1),
  x2 = c(3,0,1,1,0,1),
  x3 = c(0,0,3,2,1,1),
  y = c("red","red","red","green","green","red"))

kable(training_set)
```

x1

x2

x3

y

0

3

0

red

2

0

0

red

0

1

3

red

0

1

2

green

-1

0

1

green

1

1

1

red

Suppose we wish to use this data set to make a prediction for Y when $X1 = X2 = X3 = 0$ using K-nearest neighbors.

(a) Compute the Euclidean distance between each observation and the test point, $X1 = X2 = X3 = 0$.

The Euclidean Distance for three dimensions can be written as:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

Let's write a function that can handle this in R, then use the `rowwise()` feature of dplyr to apply it across the rows of our tibble.

```
euc_dist <- function(x1, x2) sqrt(sum((x1 - x2) ^ 2))
training_set <- training_set %>%
  rowwise() %>%
  mutate(distance = euc_dist(c(x1,x2,x3), c(0,0,0))) %>%
  ungroup()

kable(training_set)
```

x1

x2

x3

y

distance

0

3

0

red

3.000000

2

0

0

red

2.000000

0

1

3

red

3.162278

0

1

2

green

2.236068

-1

0

1

green

1.414214

1

1

1

red

1.732051

(b) What is our prediction with K = 1? Why?

Let's find the `y` value of the single closest (`k = 1`) training observation.

```
training_set %>%
  filter(distance == min(distance)) %>%
  select(y) %>%
  pull()
```

```
## [1] "green"
```

Since the closest observation in the training data is `green`, `K = 1` classifies our test observation as `green`.

(c) What is our prediction with K = 3? Why?

First we find the three closest values. Then we measure the breakdown of `y` responses in this group of three observations. We find that two observations have value of `red`, and one has `green`. Given `red` has the highest probability of the two `y` values, we assign the training observation as `red`.

```
training_set %>%
  top_n(3, -distance) %>%
  mutate(n = n()) %>%
  group_by(y) %>%
  summarise(prop = n()/max(n)) %>%
  filter(prop == max(prop)) %>%
  pull(y)
```

```
## [1] "red"
```

(d) If Bayes decision boundary is highly non-linear, then do we expect the best value of K to be large or small? Why?

- We expect the value of K to decline as the decision boundary grows more non-linear. A smaller value of K is more suspectible to small changes between observations, which is the type of pattern highly non-linear decision boundary would depict.

The R exercises are pretty basic after this. I am going to skip them for now.

Chapter 3

Linear Regression

Linear regression is a simple yet very powerful approach in statistical learning. It is important to have a strong understanding of it before moving on to more complex learning methods.

3.1 Packages used in this chapter

```
library(tidyverse)
library(modelr)
```

3.2 Simple Linear Regression

Simple linear regression is predicting a quantitative response Y based off a single predcitor X.

It can be written as below:

$Y \approx \beta_0 + \beta_1 X$

simple linear regression

β_0 and β_1 represent the *intercept* and *slope* terms and are together known as the *coefficients*. β_0 and β_1 represent the unknown *intercept* and *slope* terms and are together known as the *coefficients*. We will use our training data to estimate these parameters and thus estimate the response Y based on the value of $X = x$:

$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$

3.2.1 Estimating the Coefficients

We need to use data to estimate these coefficients.

$(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$

These represent the training observations, in this case pairs of X and Y measurements. The goal is to use these measurements to estimate β_0 and β_1 such that the linear model fits our data as close as possible. Measuring *closeness* can be tackled a number of ways, but least squares is the most popular.

If we let $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ be the prediction of Y at observation X_i, then $e_i = y_i - \hat{y}_i$ represents the ith *residual*, the difference between the observed value y_i and the predicted value $\hat{y}_i$. Now we can define the *residual sum of squares (RSS)* as

$RSS = e_1^2 + e_2^2 + ... + e_n^2$

residual sum of squares

or more explicitly as

$RSS = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + ... + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$

Minimizing the RSS (proof can be found here) using β_0 and β_1 produces:

$$\frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

least squares coefficient estimates (simple linear regression)

3.2.2 Assessing the Accuracy of the Coefficient Estimate

Remember that the true function for f contains a random error term ϵ. This means the linear relationship can be written as

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

population regression line

β_0 is the intercept term (value of Y when $X = 0$). β_1 is the slope (how much does Y change with one-unit change of X). ϵ is the error term that captures everything our model doesn't (unknown variables, measurement error, unknown true relationship).

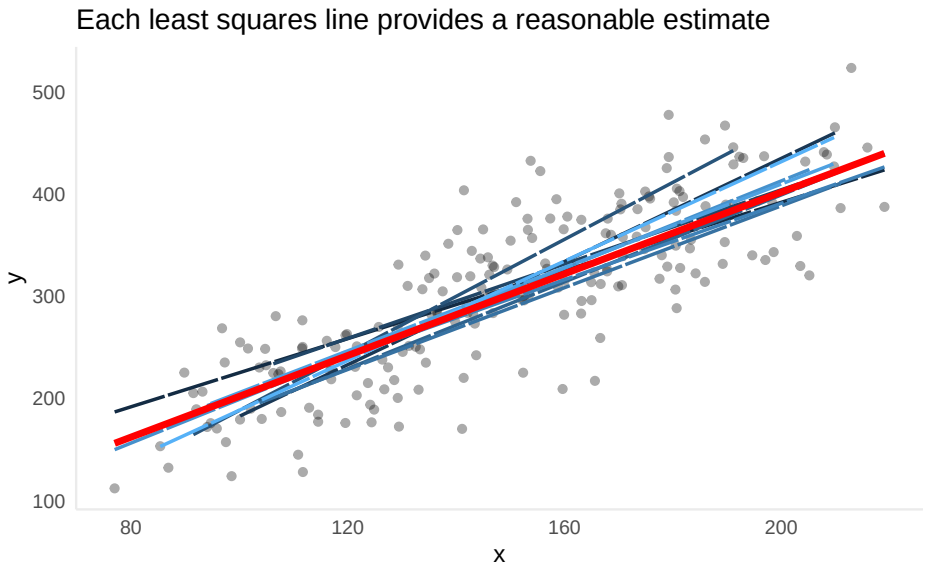
The population regression line captures the best linear approximation to the true relationship between X and Y. In real data, we often don't know the true

relationship and have to rely on a set of observations. Using the observations to estimate the coefficients via least squares produces the *least squares line*. Let's simulate and visualize this relationship:

- simulate `n = 200` observations
  - compare the population regression line (`sim_y`) to a number of possible least squares lines (generated from 10 different training sets of the data)

```
# f(x), or Y = 2 + 2x + error

sim_linear <- tibble(
  b0 = 2,
  b1 = 2,
  x = 1:100 + rnorm(n = 200, mean = 100, sd = 15),
  err = rnorm(200, sd = 50),
  sim_y = b0 + b1 * x,
  true_y = b0 + b1 * x + err
)

# generate 10 training sets
y <- tibble()
for (i in 1:10) {
  x <- sample_frac(sim_linear, 0.1) %>% mutate(iter_set = i)
  y <- y %>% bind_rows(x)
}

# apply linear model to each sample
by_iter <- y %>%
  group_by(iter_set) %>%
  nest()
lm_model <- function(df) {
  lm(true_y ~ x, data = df)
}
by_iter <- by_iter %>%
  mutate(
    model = map(data, lm_model),
    preds = map2(data, model, add_predictions)
  )

# extract predictions
preds <- unnest(by_iter, preds)

ggplot(data = sim_linear, aes(x = x, y = true_y)) +
  geom_point(alpha = 1 / 3) +
  geom_line(data = preds, aes(x = x, y = pred, colour = iter_set, group = iter_set), linetype = '
```

```
  geom_line(aes(y = sim_y), colour = "red", size = 1.5) +
  theme_minimal() +
  theme(
    legend.position = "none", panel.grid.minor = element_blank(),
    panel.grid.major = element_blank(), axis.line = element_line(colour = "grey92")
  ) +
  labs(
    title = "Each least squares line provides a reasonable estimate",
    y = "y"
  )
```



The chart above demonstrates the population regression line (red) surrounded by ten different estimates of the least squares line. Notice how every least squares line (shades of blue) is different. This is because each one is generated from a random sample pulled from the simulated data. For a real-world comparison, the simulated data would be the entire population data which is often impossible to obtain. The observations used to generate the least squares line would be the sample data we have access to. In the same way a sample mean can provide a reasonable estimate of the population mean, fitting a least squares line can provide a reasonable estimate of the population regression line.

This comparison of linear regression to estimating population means touches on the topic of bias. An estimate of μ using the the sample mean $\hat{\mu}$ is unbiased. On average, the sample mean will not systemically over or underestimate μ. If we were to take a large enough estimates of μ, each produced by a particular set of observations, then this average would exactly equal μ. This concept applies to our estimates of β_0, β_1 as well.

A question that can be asked is how close on average the sample mean $\hat{\mu}$ is to μ. We can compute the *standard error* of $\hat{\mu}$ to answer this.

$Var(\hat{\mu}) = SE(\hat{\mu})^2 = \sigma^2/n$

standard error

This formula measures the average amount that $\hat{\mu}$ differs from μ. As the number of observations n increases, the standard error decreases.

We can also use this to calculate how close $\hat{\beta}_0, \hat{\beta}_1$ are to β_0, β_1.

$$SE(\hat{\beta}_0)^2 = \sigma^2 \left[1/n + \frac{\bar{x}^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \right]$$

$$SE(\hat{\beta}_1)^2 = \frac{\sigma^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

where $\sigma^2 = Var(\epsilon)$. For this to work, the assumption has to be made that the error terms ϵ_i are uncorrelated and all share a common variance. This is often not the case, but it doesn't mean the formula can't be used for a decent approximation. σ^2 is not known, but can be estimated from training observations. This estimate is the *residual standard error* and is given by formula $RSE = \sqrt{RSS/(n-2}$.

What can we use these standard error formulas for? A useful technique is to calculate *confidence intervals* from the standard error. If we wanted to compute a 95% confidence interval for β_0, β_1, it would take the form below.

$\hat{\beta}_1 \pm 2 * SE(\hat{\beta}_1)$

$\hat{\beta}_0 \pm 2 * SE(\hat{\beta}_0)$

Standard errors can also be used to perform hypotheses tests.

H_0: There is no relationship between X and Y, or $\beta_1 = 0$

null hypothesis

H_0: There exists a relationship between X and Y, or $\beta_1 \neq 0$

alternative hypothesis

To test the null hypothesis, we need to test whether $\hat{\beta}_1$ is far enough away from zero to conclude that is it non-zero. How far enough from zero is determined by the value of $\hat{\beta}_1$ as well as $SE(\hat{\beta}_1)$. We compute a *t-statistic*

$t = (\beta_1 - 0)/SE(\hat{\beta}_1)$

t-statistic

This measures how many standard deviations $\hat{\beta}_1$ is from 0. If there is no relationship between X and Y, then t will follow a t-distribution. The t-distribution is similar to the normal distribution, but has slightly heavier tails. Like the normal distribution, we can use this to compute the probability of observing any number equal to or larger than $|t|$. This probability is the *p-value*. We can interpret a p-value as the probability we would observe the sample data that produced the t-statistic, given that there is no actual relationship between the predictor X and the response Y. This means that a small p-value supports the inference that there exists a relationship between the predictor and the response. In this case, based on whichever threshold α (common value is 0.05) we set, a small enough p-value would lead us to reject the null hypothesis.

3.2.3 Assessing the Accuracy of the Model

Now that we determined the existence of a relationship, how can we measure how well the model fits the data?

Measuring the quality of a linear regression fit is often handled by two quantities: the *residual standard error* and the *R^2* statistic.

3.2.3.1 Residual Standard Error

Since every observation has an associated error term ϵ, having the knowledge of true β_0 and β_1 will still not allow one to perfectly predict Y. The residual standard error estimates the standard deviation of the error term.

$RSE = \sqrt{1/(n-2) * RSS} = \sqrt{1/(n-2) \sum_{i=1}^{n} (y_i - \hat{y})^2}$

residual standard error

We can interpret the residual standard error as how much, on average, our predictions deviate from the true value. Whether the value is acceptable in terms of being a successful model depends on the context of the problem. Predicting hardware failure on an airplane would obviously carry much more stringent requirements than predicting the added sales from a change in a company's advertising budget.

3.2.3.2 R^2 statistic

The RSE provides an absolute number. Given that it depends on the scale of Y, comparing RSE values across different domains and datasets isn't useful. The R^2 statistic solves this problem by measuring in terms of proportion – it measures the variance explained and so always takes a value between 0 and 1.

$R^2 = (TSS - RSS)/TSS = 1 - RSS/TSS$

R^2 statistic

where $TSS = \sum_{i=1}^{n}(y_i - \bar{y})^2$ is the *total sum of squares*. TSS can be thought of the amount of total variability in the response variable before any model is fitted to it. RSS is measured after fitting a model, and measures the amount of unexplained variance remaining in the data. Therefore, R^2 can be thought of as the proportion of variance in the data that is explained by fitting a model with X. While R^2 is more intrepetable, determing what constitutes a R^2 is subjective to the problem. Relationships that are known to be linear with little variance would expect an R^2 very close to 1. In reality, a lot of real-world data is not truly linear and could be heavily influenced by unknown, immeasurable predictors. In such cases a linear approximation would be a rough fit, and a smaller R^2 would not be unordinary.

There is a relation between R^2 and the correlation.

$r = Cor(X, Y) = \sum_{i=1}^{n}((x_i - \bar{x})(y_i - \bar{y}))/(\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2})$

correlation

Both measure the linear relationship between X and Y, and within the simple linear regression domain, $r^2 = R^2$. Once we move into multiple linear regression, in which we are using multiple predictors to predict a response, correlation loses effectiveness at measuring a model in whole as it can only measure the relationship between a single pair of variables.

3.3 Multiple Linear Regression

Simple linear regression works well when the data involves a single predictor variable. In reality, there are often multiple predictor variables. We will need to extend the simple linear regression model and provide each predictor variable p with a slope coefficient.

$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \epsilon$

multiple linear regression

3.3.1 Estimating the Regression Coefficients

Again, we need to estimate the regression coefficients.

$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + ... + \hat{\beta}_p X_p$

We will utilize the same approach of minimizing the sum of squared residuals (RSS).

$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - ... - \hat{\beta}_p x_{ip})^2$

Minimizing these coefficients is more complicated than the simple linear regression setting, and is best represented using linear algebra. See this Wikipedia section for more information on the formula.

Interpreting a particular coefficient, (say β_1) in a multiple regression model can be thought of as follows: if constant value for all other β_p are maintained, what effect would an increase in $beta_1$ have on Y?

A side effect of this is that certain predictors which were deemed significant when contained in a simple linear regression can become insignificant when multiple predictors are involved. For an advertising example, `newspaper` could be a significant predictor of `revenue` in the simple linear regression context. However, when combined with `tv` and `radio` in a multiple linear regression setting, the effects of increasing `newspaper` spend while maintaining `tv` and `radio` becomes insignificant. This could be due to a correlation of `newspaper` spend in markets where `radio` spend is high. Multiple linear regression exposes predictors that act as "surrogates" for others due to correlation.

3.3.2 Some Important Questions

3.3.2.1 Is There a Relationship Between the Response and Predictors?

To check this, we need to check whethere all p coefficients are zero, i.e. $\beta_1 = \beta_2 = ... = \beta_p = 0$. We test the null hypothesis,

$H_o : \beta_1 = \beta_2 = ... = \beta_p = 0$

against the alternative

H_a : at least one β_j is non-zero

The hypothesis test is performed by computing the $F - statistic$,

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

correlation

If linear model assumptions are correct, one can show that

$E\{RSS/(n-p-1)\} = \sigma^2$

and that, provided H_o is true,

$E\{(TSS-RSS)/p\} = \sigma^2$

In simple terms, if H_o were true and all of the predictors have regression coefficients of 0, we would expect the unexplained variance of the model to be approximately equal to that of the total variance, and both the numerator and the denominator of the F-statistic formula to be equal. When there is no relationship between the response and predictors, the F-statistic will take on a value

close to 1. However, as RSS shrinks (the model begins to account for more of the variance), the numerator grows and the denominator shrinks, both causing the F-statistic to increase. We can think of the F-statistic as a ratio between the explained variance and unexplained variance. As the explained variance grows larger than the unexplained portion, the likelihood that we reject the null hypothesis grows.

How large does the F-statistic need to be to reject the null hypothesis? This depends on n and p. As n grows, F-statistics closer to 1 may provide sufficient evidence to reject H_o. If H_o is true and ϵ_i have a normal distribution, the F-statistic follows an F-distribution. We can compute the p-value for any value of n and p associated with an F-statistic.

```
# TODO remove this section?
```

Sometimes we want to test whether a particular subset of q of the coefficients are zero.

The null hypothesis could be

$$H_o : \beta_{p-q+1} = \beta_{p-q+2} = \beta_p = 0$$

In this case we fit a second model that uses all the variables except the last q. We will call the residual sum of squares for the second model RSS_0.

Then, the F-statistic is,

$$F = \frac{(RSS_0 - RSS)/q}{RSS(n-p-1)}$$

We are testing a model without the q predictors and seeing how it compares to the original model containing all the predictors.

Why do we need to look at overall F-statistics if we have individual p-values of the predictors? There are scenarios where individual predictors, by chance, will have *small* p-values, even in the absence of any true association. This could lead us to incorrectly diagnose a relationship.

The overall F-statistic does not suffer this problem because it adjusts for the number of predictors.

The F-statistic approach works when the number of predictors p is small compared to n. Sometimes, we have situations where $p > n$. In this situation, ther eare more coefficients β_j to estimate than observations from which to estimate them. Such situations requires different approaches that we haven't discussed yet (see chapter 6)

```
# TODO add chapter 6 link
```

3.3.2.2 Deciding on Important Variables

The first thing we do in a multiple regression is to compute the F-statistic and determine that at least one of the predictors is related to the response.

The task of determining which predictors are associated with the response is referred to as *variable selection*. We could try out a lot of different models with combinations of predictors, 2^p, but this is not practical as p grows.

There are three ways to approach this task:

- *Forward selection*: we begin with the *null model*, which contains an intercept but no predictors. We then fit p simple linear regressions and add to the null model the variable that results in the lowest RSS. We then repeat the process to determine the lowest RSS of the now two-variable model, continuing until some stopping rule is satisfied.
- *Backward selection*: Start with all the variables in the model, remove the variable with the largest p-value. Then, for the new $(p-1)$-variable model, do the same. Continue until stopping rule is reached (for example, some p-value threshold)
- *Mixed selection*: Start with no variables, and proceed with forward selection. If any p-value of added variables pass a threshold once new predictors are added, we remove them. We continue the forward and backward until all variables in model have a sufficiently low p-value.

3.3.2.3 Model Fit

Two common methods of model fit are the RSE and R^2, the fraction of variance explained.

More on R^2:

- Values closer to `1` indicate a better fit
- Adding more variables can only increase it
  - Adding variables that barely increase it can lead to overfitting

Plotting the model can also be useful.

3.3.2.4 Predictions

Three sorts of uncertainty within a given model:

1. The coefficient estimates $\hat{\beta}_0 + \hat{\beta}_1 ..., \hat{\beta}_p$ are estimates for $\beta_0 + \beta_1 ..., \beta_p$. This inaccuracy is part of the *reducible error*. We can compute a confidence interval to determine how close $\hat{Y}$ is to $f(X)$.

2. *Model bias* can result from the fact that we are fitting a linear approximation to the true surface of $f(X)$.

3. Even if we knew $f(X)$, we still have random error ϵ, which is the *irreducible error*. We can use prediction intervals to estimate how far Y will differ from $\hat{Y}$. These will always be larger than confidence intervals, because they incorporate both the reducible + irreducible error.

3.3.3 Other Considerations in the Regression Model

So far, all predictors have been *quantitative*. However, it is common to have *qualitative* variables as well.

Take a look at the `ISLR::Credit` dataset, which has a mix of both types.

```
tidy_credit <- ISLR::Credit %>%
  as_tibble() %>%
  janitor::clean_names()
tidy_credit
```

```
## # A tibble: 400 x 12
##      id income limit rating cards   age education gender student married
##   <int>  <dbl> <int>  <int> <int> <int>     <int> <fct>  <fct>   <fct>
## 1     1   14.9  3606    283     2    34        11 " Mal~ No      Yes
## 2     2  106.   6645    483     3    82        15 Female Yes     Yes
## 3     3  105.   7075    514     4    71        11 " Mal~ No      No
## 4     4  149.   9504    681     3    36        11 Female No      No
## 5     5   55.9  4897    357     2    68        16 " Mal~ No      Yes
## 6     6   80.2  8047    569     4    77        10 " Mal~ No      No
## # ... with 394 more rows, and 2 more variables: ethnicity <fct>,
## #   balance <int>
```

3.3.3.1 Predictors with only Two Levels

Suppose we wish to investigate difference in credit card balance between males and females, ignoring all other variables. If a *qualitative* variable (also known as a *factor*) only has two possible values, then incorporating it into a model is easy. We can create a binomial dummy variable that takes on two values. For `gender`, this could be a variable that is `0` if observation has value `male`, and `1` if observation has value `female`. This variable can then be used in the regression equation.

Take note that `lm()` automatically creates dummy variables when given qualitative predictors.

```
# TODO insert regression equation
```

```
credit_model <- lm(balance ~ gender, data = tidy_credit)
tidy_credit_model <- broom::tidy(credit_model)
tidy_credit_model
```

```
## # A tibble: 2 x 5
##   term         estimate std.error statistic  p.value
##   <chr>           <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)     510.       33.1    15.4   2.91e-42
## 2 genderFemale     19.7      46.1     0.429 6.69e- 1
```

How to interpret this: males are estimated to carry a balance of $510. Meanwhile, females are expected to carry an additional 19.70 in debt. Notice the p-value is very high, indicating there is no significant difference between genders.

3.3.4 Qualitative Predictors with More than Two Levels

A single dummy variable can not represent all the possible values. We can create additional dummy variables for this.

Let's make a dummy variable from `ethnicity` column, which takes three distinct values. This will yield two dummy variables.

```
tidy_credit %>% distinct(ethnicity)
```

```
## # A tibble: 3 x 1
##   ethnicity
##   <fct>
## 1 Caucasian
## 2 Asian
## 3 African American
```

`fastDummies` package will be used to generate these. In this case, `African American` serves as the baseline, and dummy variables are created for `Caucasian` and `Asian`. **There will always be one fewer dummy variable than the number of levels.**

```
tidy_credit_dummy <- tidy_credit %>%
  fastDummies::dummy_cols(select_columns = "ethnicity", remove_first_dummy = TRUE) %>%
  janitor::clean_names()

tidy_credit_dummy %>%
  select(starts_with("ethnicity"))
```

```
## # A tibble: 400 x 3
##   ethnicity ethnicity_asian ethnicity_caucasian
##   <fct>               <int>               <int>
```

```
## 1 Caucasian                0                   1
## 2 Asian                    1                   0
## 3 Asian                    1                   0
## 4 Asian                    1                   0
## 5 Caucasian                0                   1
## 6 Caucasian                0                   1
## # ... with 394 more rows
```

We can again run the model with newly created dummy variables. Keep in mind, prior creation is not necessary, as `lm` will generate them automatically.

```
ethnicity_model <- lm(balance ~ ethnicity_asian + ethnicity_caucasian, data = tidy_credit_dummy)
broom::tidy(ethnicity_model)
```

```
## # A tibble: 3 x 5
##   term                estimate std.error statistic  p.value
##   <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)            531.       46.3    11.5   1.77e-26
## 2 ethnicity_asian        -18.7      65.0    -0.287 7.74e- 1
## 3 ethnicity_caucasian    -12.5      56.7    -0.221 8.26e- 1
```

3.3.5 Extensions of the Linear Model

The linear regression model makes highly restrictive assumptions. Two of the most important are that the relationship between predictors and response are *additive* and *linear*.

Additive means that the effect of changes in a predictor X_j on the response Y is independet of the values of the other predictors. Linear means that the the change in response Y to a one-unit change in X_j is constant, regardless of the value of X_j.

Here are some common approaches of extending the linear model.

3.3.5.1 Removing the Additive Assumption

The additive property assumes that predictors slope terms are independent of the values of other predictors. However, this is not always the case. Imagine an advertising scenario where the effectiveness of TV spend is affected by the radio spend. This is known as an *interaction* effect. Imagine we have a model with two predictors, but they are not strictly additive. We could extend this model by adding an *interaction term* to it.

$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$

Now, the effect of X_1 on Y is no longer constant; adjusting X_2 will change the impact of X_1 on Y.

An easy scenario is the productivity of a factory. Adding lines and workers both would increase productivity. However, the effect is not purely additive. Adding lines without having workers to operate them would not increase productivity. There is an interaction between workers and lines that needs to be accounted for.

The *hierarchical principle* states that *if we include an interaction in a model, we should also include the main effects, even if the p-values associated with their coefficients are not significant.*

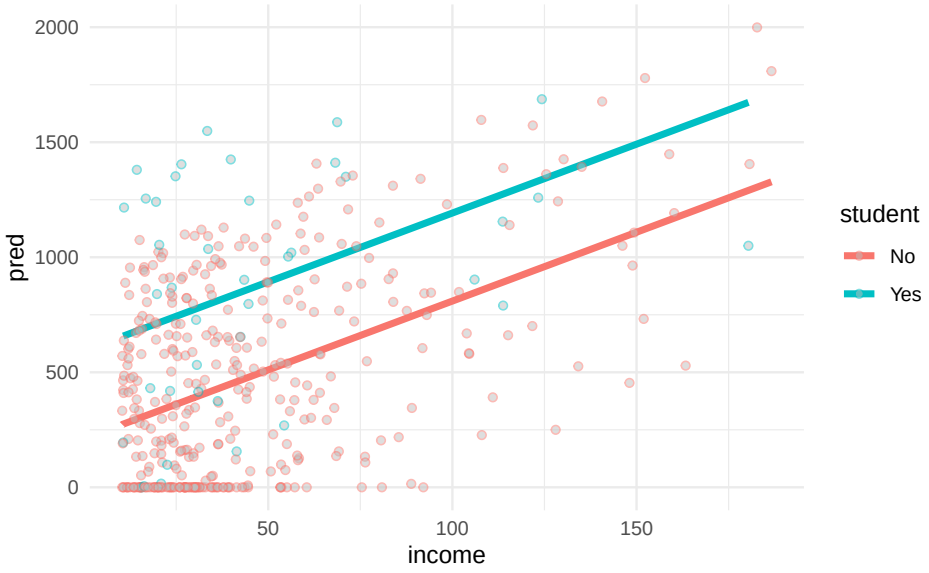
It's also possible for qualitative and quantitative variables to interact with each other. We will again use the `Credit` data set. Suppose we wish to predict `balance` using the `income` (quantitative) and `student` (qualitative) variables.

First, let's take a look at what it looks like to fit this model without an interaction term. Both `income` and `student` are significant.

```
lm_credit <- lm(balance ~ income + student, data = tidy_credit)
lm_credit %>% broom::tidy()
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   211.      32.5        6.51 2.34e-10
## 2 income          5.98     0.557     10.8  7.82e-24
## 3 studentYes    383.      65.3        5.86 9.78e- 9
```

```
tidy_credit %>%
  modelr::add_predictions(lm_credit) %>%
  ggplot(aes(x = income, y = pred, colour = student)) +
  geom_line(size = 1.5) +
  geom_point(aes(y = balance, colour = student), fill = "grey", pch = 21, alpha = 1 / 1
  theme_minimal()
```



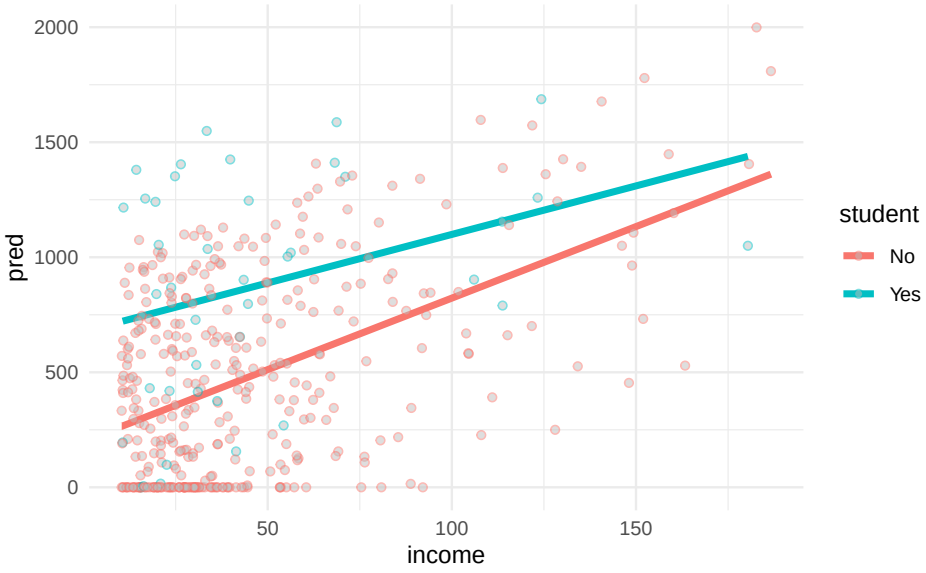
It's a pretty good fit, and because there is no interaction terms, the lines are parallel. Notice how many more observations there are to fit on for the non-students.

Now, let's add an interaction term.

```
lm_credit_int <- lm(balance ~ income + student + income * student, data = tidy_credit)
lm_credit_int %>% broom::tidy()
```

```
## # A tibble: 4 x 5
##   term              estimate std.error statistic  p.value
##   <chr>                <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)         201.      33.7        5.95 5.79e- 9
## 2 income                6.22     0.592     10.5  6.34e-23
## 3 studentYes          477.     104.         4.57 6.59e- 6
## 4 income:studentYes    -2.00     1.73      -1.15 2.49e- 1
```

```
tidy_credit %>%
  modelr::add_predictions(lm_credit_int) %>%
  ggplot(aes(x = income, y = pred, colour = student)) +
  geom_line(size = 1.5) +
  geom_point(aes(y = balance, colour = student), fill = "grey", pch = 21, alpha = 1 / 2) +
  theme_minimal()
```



The model now takes into account how `income` and `student` interact with each other. Interpreting the chart suggests that increases in `income` among students has a smaller effect on balance than it does to non-students.

Does it fit better?

```
models <- list(without_interaction = lm_credit, with_interaction = lm_credit_int)
purrr::map_df(models, broom::glance, .id = "model") %>%
  select(model, r.squared, statistic, p.value, df)
```

```
## # A tibble: 2 x 5
##   model               r.squared statistic  p.value    df
##   <chr>                   <dbl>     <dbl>    <dbl> <int>
## 1 without_interaction     0.277      76.2 9.64e-29     3
## 2 with_interaction        0.280      51.3 4.94e-28     4
```

Not by much. The model with the interaction term has a slightly higher R^2, but the added complexity of the model, combined with the small number of observations of students in the dataset, suggests overfitting.

3.3.5.2 Non-linear Relationships

The linear model assumes a linear relationship between the response and predictors. We can extend the linear model to accomodate non-linear relationships using *polynomial regression*.

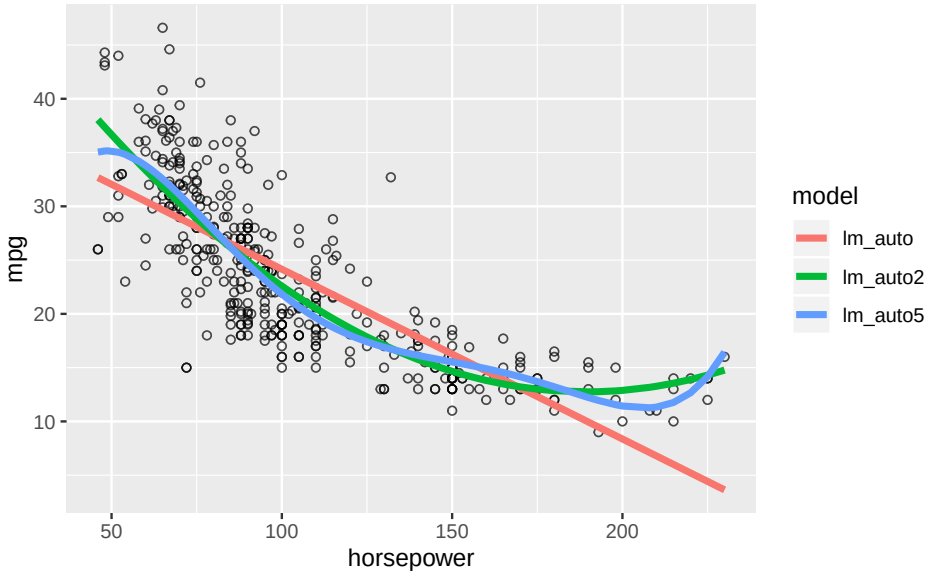
A way to incorporate non-linear associations into a linear model is to include transformed versions of the predictor in the model. For example, within the

`Auto` dataset, predicting `mpg` with a second-order polynomial of `horsepower` would look like this:

$$mpg = \beta_0 + \beta_1 horsepower + \beta_2 horsepower^2 + \epsilon$$

Let's look at the `Auto` dataset with models of different polynomial degrees overlaid. Clearly, the data is not linear, exhibiting a *quadratic* shape.

```
tidy_auto <- ISLR::Auto %>% as_tibble()

lm_auto <- lm(mpg ~ horsepower, data = tidy_auto)
lm_auto2 <- lm(mpg ~ poly(horsepower, 2), data = tidy_auto)
lm_auto5 <- lm(mpg ~ poly(horsepower, 5), data = tidy_auto)

tidy_auto %>%
  gather_predictions(lm_auto, lm_auto2, lm_auto5) %>%
  ggplot(aes(x = horsepower, y = mpg)) +
  geom_point(alpha = 1 / 3, pch = 21) +
  geom_line(aes(y = pred, colour = model), size = 1.5)
```



The second-order polynomial does a good job of fitting the data, while the fifth-order seems to be unnecessary. The model performance reflects that:

```
models <- list(
  linear = lm_auto,
  second_order = lm_auto2,
  fifth_order = lm_auto5
)
```

```
purrr::map_df(models, broom::glance, .id = "model") %>%
  select(model, r.squared, statistic, p.value, df)
```

```
## # A tibble: 3 x 5
##   model        r.squared statistic  p.value    df
##   <chr>            <dbl>     <dbl>    <dbl> <int>
## 1 linear           0.606      600. 7.03e-81     2
## 2 second_order     0.688      428. 5.40e-99     3
## 3 fifth_order      0.697      177. 1.16e-97     6
```

The second-order model has significantly higher R^2, and only one more degree of freedom.

This approach of extending linear models to accomodate non-linear relationships is known as polynomial regression.

3.3.6 Potential Problems

Many problems can occur when fitting a linear model to a data set.

1. Non-linearity of the response-predictor relationship.
2. Correlation of error terms.
3. Non-constant variance of error terms
4. Outliers
5. High-leverage points
6. Collinearity

3.3.6.1 1. Non-linearity of the data

The linear model assumes a straight-line relationship between the predictors and the response. If this is not the case, the inference and prediction accuracy of the fit are suspect.

We can use a *residual plot* to visualize when a linear model is placed on to a non-linear relationship. For a simple linear regression model, we plot the residuals $e_i = y_i - \hat{y}_i$ compared to the predictor. For multiple regression, we plot the residuals versus the predicted values $\hat{y}_i$. If the relationship is linear, the residuals should exhibit a random pattern.

Let's take the `Auto` dataset and plot the residuals compared to `horsepower` for each model we fit. Notice in the model containing no quadratic term is U-shaped, indicating a non-linear relationship. The model that contains `horsepower^2` exhibits little pattern in the residuals, indicating a better fit.

```
tidy_auto %>%
  gather_predictions(lm_auto, lm_auto2, lm_auto5) %>%
  ggplot(aes(x = horsepower, y = mpg-pred, colour=model)) +
```

```
  geom_point(alpha = 1 / 3) +
  geom_hline(yintercept = 0, size = 1.5, colour="grey") +
  facet_wrap(~model, nrow=3)
```

If the residual plot indicates that there non-linear associations in the data, a simple approach is to use non-linear transofmrations of the predictors.

3.3.6.2 2. Correlation of Error Terms

The linear regression model assumes that the error terms $\epsilon_1, \epsilon_2, ..., \epsilon_n$ are uncorrelated.

This means that for a given error term e_i, no information is provided about the value e_{i+1}. The standard errors that are computed for the estimated regression coefficients are based on this assumption.

If there is a correlation among the error terms, than the estimated standard errors will tend to underestimate the true standard errors, producing confidence and prediction intervals narrower than they should be. Given the incorrect assumption, a 95% confidence interval may have a much lower probability than 0.95 of containing the true value of the parameter. P-values would also be lower than they should be, giving us an unwarranted sense of confidence in our model.

These correlations occur frequently in *time series* data, which consists of observations obtained at discrete points in time. In many cases, observations that are obtained at adjacent time periods points will have positively correlated errors.

We can again plot the residuals as a function of time to see if this is the case. If no correlation, there should be no pattern in the residuals. If error terms exhibit correlation, we may see that adjacent residuals exhibit similar values, known as *tracking*.

This can also happen outside of time series data. The assumption of uncorrelated errors is extremely important for linear regression as well as other statistical methods.

```
# TODO add a residual plot for time series with correlated error terms, similar to pg.
```

```
tidy_sunspot <- data.frame(y=as.matrix(sunspot.year), ds=time(sunspot.year)) %>% as_ti
tidy_sunspot
sunspot_lm <- lm(data = tidy_sunspot, y ~ ds)
tidy_sunspot %>%
  add_predictions(sunspot_lm) %>%
  ggplot(aes(x=ds, y=pred-y)) +
  geom_point() +
  geom_line() +
  geom_smooth(method="lm", se = FALSE)
```

3.3.6.3 3. Non-constant Variance of Error Terms

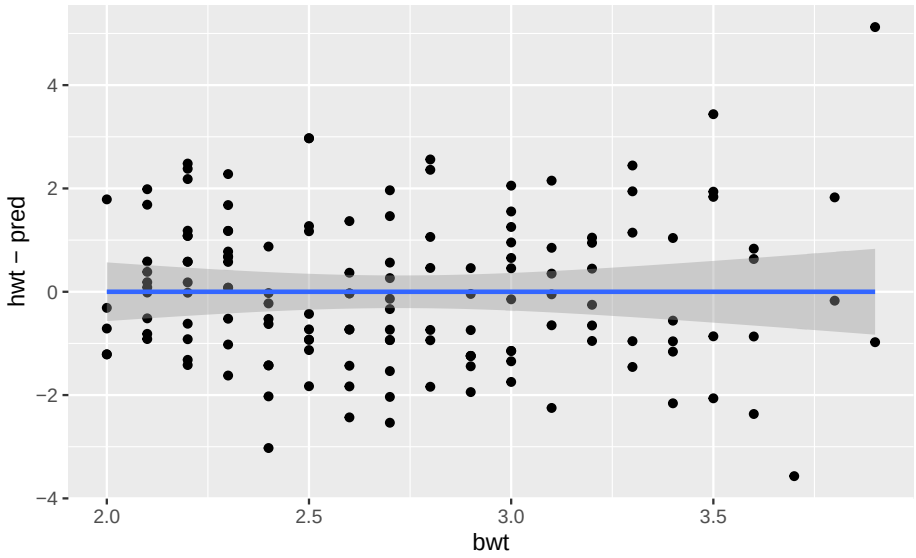
Another assumption of linear regression is that the error terms have a constant variance, $Var(\epsilon_i) = \sigma^2$. Standard errors, confidence intervals, and hypothesis tests rely upon this assumption.

It is common for error terms to exhiti non-constant variance. Non-constant variance in the errors, also known as *heteroscedasticity*, can be identified from a *funnel shape* in the residual plot.

Let's take a look at the `MASS::cats` dataset, which contains observations of various cats sex, body weight, and heart weight. We fit a linear model to it, and then plot the residuals.

I've added a linear fit to the residual plot itself. Observe how the error terms begin to funnel out as `bwt` increases, indicating non-constant variance of the error terms.

```
tidy_cats <- MASS::cats %>% as_tibble() %>% janitor::clean_names()
lm_cats <- lm(data = tidy_cats, hwt ~ bwt)

tidy_cats %>%
  add_predictions(lm_cats) %>%
  ggplot(aes(x = bwt, y = hwt - pred)) +
  geom_point() +
  geom_smooth(method="lm", level = 0.99)
```




```
# TODO add OLS method to this
# get weights of each response

# fit a linear model on the residuals
tidy_cats_res <- tidy_cats %>%
  add_predictions(lm_cats) %>%
  mutate(res = hwt - pred) %>%
  select(bwt, hwt, res)
lm_cats_res <- lm(data = tidy_cats_res, res ~ hwt)

cat_weights <- tidy_cats_res %>%
  add_predictions(lm_cats_res) %>%
  mutate(res_var = (res-pred)^2) %>%
  mutate(weight = 1 / res_var) %>%
  pull(weight)

lm_cats_weights <- lm(data = tidy_cats, hwt ~ bwt, weights = cat_weights)

tidy_cats %>%
  gather_predictions(lm_cats, lm_cats_weights) %>%
  ggplot(aes(x = bwt, y = hwt - pred, colour = model)) +
  geom_point(aes(y = hwt-pred, colour = model)) +
  geom_smooth(method="lm")
```

When this occurs, there a few ways to remedy it. You could transform the response Y using a function such as $logY$ or $\sqrt{Y}$. If we have a good idea of the variance of each response, we could fit our model using *weighted least*

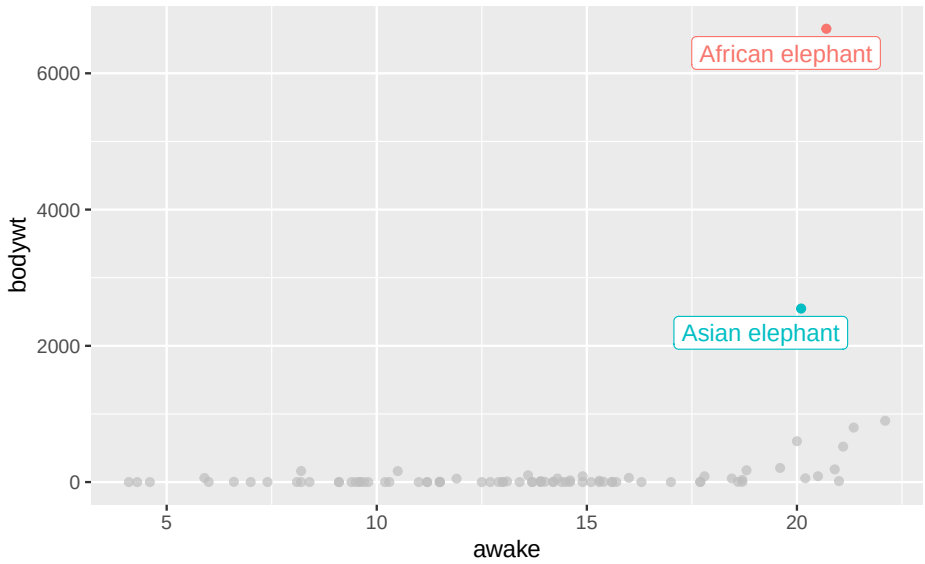
squares, which weights proportional to the inverse of the expected variance of an observation.

3.3.6.4 4. Outliers

An *outlier* is a point for which y_i is far from the value predicted by the model. These can arise for a variety of reasons, such as incorrect recording of an observation during data collection.

For the `msleep` dataset below, which contains data on mammal sleep durations, I've highlighted two observations that most would consider outliers. This is for the `African elephant` and `Asian elephant` mammals, who's bodyweights are far and away from the rest of mammals. There are others that could be considered outliers as well. Identifying outliers is an often arbitrary process.

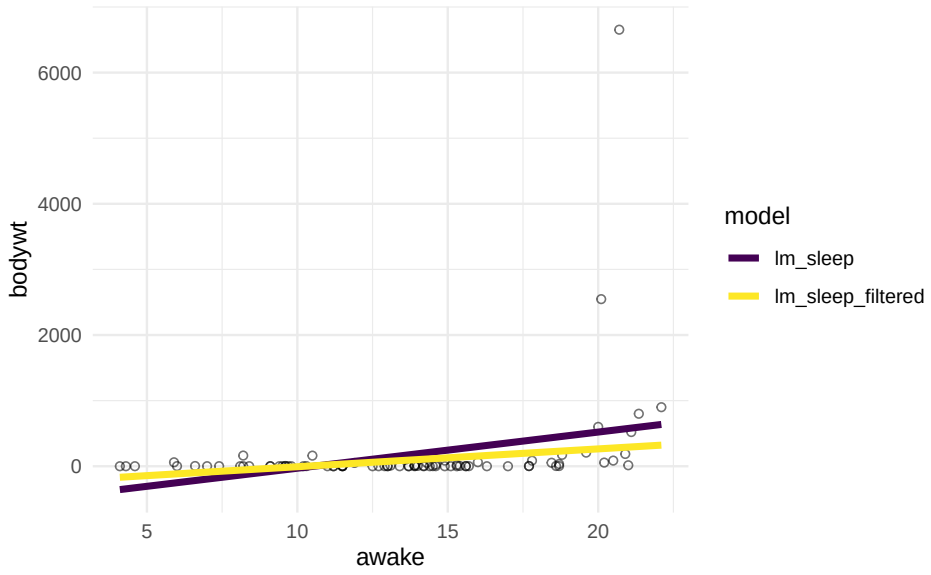
```
msleep %>%
  ggplot(aes(x = awake, y = bodywt, colour = name)) +
  geom_point() +
  gghighlight::gghighlight(bodywt>2000)
```



Let's fit a linear model to predict body weight from how long the animal is awake.

Notice how the data that maintains the elephant observations significantly affects the slope, drawing the regression line away from the majority of observations.

```
lm_sleep <- lm(data = msleep, bodywt ~ awake)
lm_sleep_filtered <- lm(data = msleep %>% filter(name != 'African elephant'), bodywt ~ awake)
msleep %>%
  gather_predictions(lm_sleep, lm_sleep_filtered) %>%
  ggplot(aes(x = awake, y = bodywt)) +
  geom_point(pch=21, alpha = 1/3) +
  geom_line(aes(y = pred, colour = model), size = 1.5) +
  scale_colour_viridis_d() +
  theme_minimal()
```



The model excluding the elephant observations has a significantly higher R^2, which indicates a better fit.

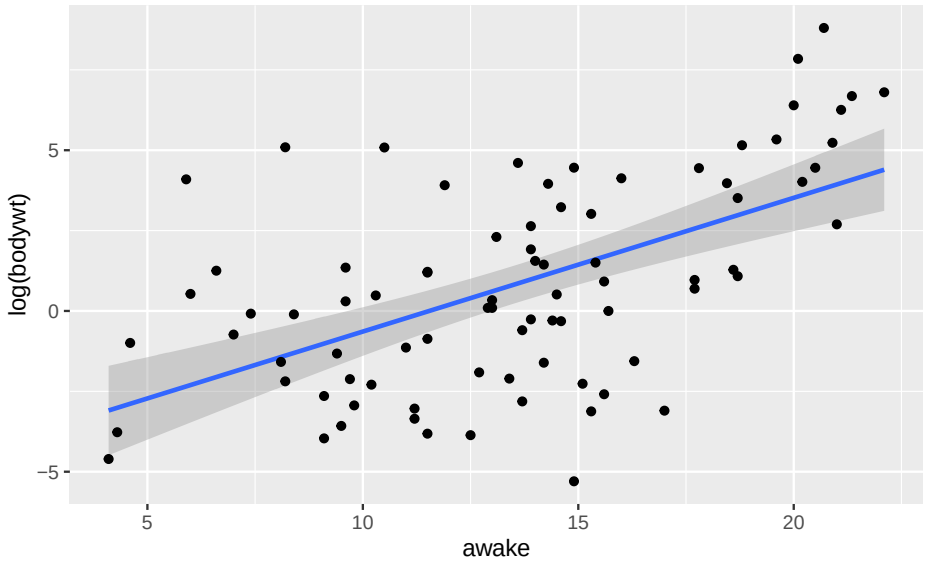
```
models <- list(with_outliers = lm_sleep,
               without_outliers = lm_sleep_filtered)
purrr::map_df(models, broom::glance, .id = "model") %>%
  select(model, r.squared, statistic, p.value, df)
```

```
## # A tibble: 2 x 5
##   model            r.squared statistic  p.value    df
##   <chr>                <dbl>     <dbl>    <dbl> <int>
## 1 with_outliers       0.0973      8.73 0.00409      2
## 2 without_outliers    0.142      13.3  0.000473     2
```

Another way of handling an outlier is transforming the predictor. Upon inspection of the scatterplot, it becomes clear that the relationship between `bodywt` and `awake` is not linear. If we take the same dataset and apply a `log` function

to response variable `bodywt`, we see that the outliers no longer exists.

```
msleep %>%
  ggplot(aes(x = awake, y = log(bodywt))) +
  geom_smooth(method = "lm") +
  geom_point()
```



The model that uses `log(bodywt)` as the response also has better performance than both models above.

```
lm_sleep_log <- lm(data = msleep, log(bodywt) ~ awake)
lm_sleep_log %>% broom::glance() %>%
  select(r.squared, statistic, p.value, df)
```

```
## # A tibble: 1 x 4
##   r.squared statistic      p.value    df
##       <dbl>     <dbl>        <dbl> <int>
## 1     0.324      38.7 0.0000000202     2
```

3.3.6.5 5. High Leverage Points