Chapter 1

A Taxonomy of Problems

The term "data science" has been overused in recent years, and it has become something of a buzzword as a result[1]. However, I think it can best be described as:

> **data science**: Any endeavor in which statistics, machine learning, data analysis, computer science, and information science intersect with domain knowledge.

Data science is about using the machinery of statistics and computer science to solve real-world problems. In the clinical domain, that means incorporating methods from epidemiology, biostatistics, computer science, and machine learning with insights gained from the clinical research literature and the practical experiences of physicians, nurses, hospital administrators, operational teams, and biomedical researchers.

1.1 Project Examples

Whenever I teach, I ask students to provide some examples of projects for which they think data science could be useful. The following are real examples. They provide a broad representation of most of the types of problems clinicians and health system operations/population health teams are interested in.

[1] See also: "artificial intelligence", "machine learning", "deep learning".

1. *Unnecessary ER trips.* "Given a number of factors (types of admissions a person has had in the past, number of admissions/re-admissions, social determinants, etc.) can we predict who is going to show up at the emergency room unnecessarily"

2. *Good/poor candidates for program.* "determine if patients are good or poor candidates for one of our specialty care model bundle programs"

3. *Predicting unplanned admissions.* "predicting unplanned inpatient admissions based on many different variables (e.g. chronic conditions, engagement with primary care, etc.) and how these inputs interact with each other"

4. *Recommending an intervention.* "...stratification/prioritization of care management or other interventions or for clinical decision support...a tool would recommend an appropriate intervention based on the profile of the patient"

5. *Recommending a diagnosis.* "Based on unstructured chat conversations and also structured questions/forms/data...map out possible care pathways. For example, if someone says they have stomach pain, gives their zip code, insurance, pain tolerance and symptoms, and is logged in so we have past history, ask a few more questions and then we could determine they are 45% likely to have ulcer vs. constipation vs. food poisoning vs. appendicitis."

6. *Predicting the amount paid by patients.* "Patient bill estimates - learning from claims data typical amount paid by patients for appointment reasons/types (e.g. estimate of additional services/care administered, and associated cost, based on patient details such as age, gender, etc.)"

7. *Identifying patient subtypes.* "identify cohorts within a population with chronic conditions based on their differences in longitudinal care across the continuum of settings (inpatient, ambulatory, primary care, specialty care, etc.)"

8. *Which conversations are similar?* "using previous chat histories to train (a chatbot) and become more effective/efficient for different, future patient chat experiences"

9. *Predictors of COVID-19 outcomes.* "Get baseline diabetes control marker (HbA1C) and acute glycemic control (inpatient glucose values) and see if either is a stronger predictor of COVID-19 outcomes (ICU, intubation, death)."

10. *Factors influencing mortality in myelofibrosis.* "We see lots of patients who are ineligible for clinical trials based on comorbidities and underlying organ dysfunction. However it is unclear how these factors affect OS. I would like to extract comorbidity data and baseline laboratory factors in patients with myelofibrosis to see how these factors affect mortality, if controlled for such important factors such as treatment, age, sex, insurance, number of comorbidities, and clinical risk score (DIPSS)."

11. *Non-adherence and difficult-to-treat asthma.* "We want to see whether non-adherence to prescribed inhaled corticosteroids plays a major role in poorly controlled asthma. Difficult-to-treat asthma can be evaluated by the number of ED visits, hospitalizations, prescriptions of prednisone and prescriptions of biological therapies. Using EPIC [we] can obtain medicine reconciliation information, of prescriptions sent, what proportion of those prescriptions were dispensed by Pharmacy. Question is can we find associations between the percentage of prescriptions filled and difficult-to-treat asthma."

12. *Impact of diabetes and hyperglycemia on progression-free survival.* "Aim: Assess the impact of diabetes and hyperglycemia on first-line systemic therapy response (progression-free survival) in patients with advanced non-small cell lung cancer. Diabetes- defined by presence of diagnosis codes coding for diabetes. Hyperglycemia- random glucose >200 ng/dL. Covariates of interest- age, sex, other treatments (RT, surgery), malignancy characteristics (stage, histology), smoking history, ecog (performance status), comorbidities, medications (steroids, anti-hyperglycemics)"

13. *Effect of statin use on MACE.* "Retrospective cohort study in elderly patients with CAD taking statins...exposed group are patients on a high-intensity statin; control group are patients on a moderate- or low-intensity statin. Participants matched based on age, gender, LDL category, and Elixhauser index category...The primary efficacy outcome

would be the time-to-first-event of 3-point MACE[2]."

14. *Clustering patients with NAFLD.* "We wanted to understand non-alcoholic fatty liver disease (NAFLD) better, so we developed a cohort of NAFLD patients using EMR-based criteria and then clustered them based on co-morbidities, medications, vital signs, and lab values to identify NAFLD subtypes. We then characterized the phenotypes and outcomes of the different subtypes."

1.2 Abstracting the Problem

All of these examples describe situations where we want to use data to answer questions of clinical or operational importance. While the details differ in each scenario, the important thing to notice here is that many of the tasks themselves are structurally similar.

For example, all of the items except $7-8$ and 14 describe situations where we want to associate information about a patient with a particular outcome or recommendation. Using information about a patient to estimate the size of a bill (#6) may appear to be a very different problem than uncovering factors influencing myelofibrosis mortality (#10), but the structure of the two problems is similar: the patient features are used as input, and the output is whatever quantity you care about (e.g. the cost to the patient in dollars or the probability of mortality by a certain timepoint).

Learning to see these types of similarities will give you a tremendous amount of power when attacking new problems in clinical data science. It will allow you to confidently deploy methods you used to solve one problem on a wide range of other problems. Each new method you learn then multiplies your capacity to solve problems, rather than adding to it.

Question 1.1

How are items $7-8$ and 14 different from the rest?

[2]MACE stands for "Major Adverse Cardiac Event". The 3-point MACE is a composite of nonfatal stroke, nonfatal myocardial infarction, and cardiovascular death.

Question 1.2

How are items $1-6$ similar to items $9-13$ and how are they different?

Question 1.3

How do items $1-3$ differ from items $4-5$ and how are they similar?

Question 1.4

How do items $1-3$ differ from item 6? How is item 6 different from all of the other items?

Question 1.5

How do items $9-11$ differ from items $12-13$?

1.3 Terms and Contrasts

The basic ways in which clinical data science problems vary can be characterized using a few broad conceptual distinctions. These draw from both traditional clinical disciplines, like epidemiology, as well as machine learning/statistics.

1.3.1 Guidance vs. Understanding

Before beginning any study, it is important to carefully consider the study's goal and how the findings from the study will be used. This will help guide you in choosing appropriate methods. For example, in some studies we care mainly about using data to provide **guidance** that will enable us to perform our jobs better in the future. We may want to predict whether a patient is likely to experience an adverse outcome, or we may want to learn the type of patient who is most likely to benefit from a particular treatment. In these cases, we want the data to guide us in making better choices.

Now, contrast this with a study whose primary goal is scientific **understanding**. In this case, we care more about using data to improve our understanding of a phenomenon than in operationalizing those findings. For example, we may be interested in whether a particular genetic variant affects a phenotype, or we may want to establish a causal link between a particular treatment and an outcome.

The distinction is fuzzy and often imperfect, and the same kinds of methods can often be used in both cases. Depending on the goal, however, one may be willing to make certain compromises. For example, complex, "black box" predictive models (e.g. deep learning models) may be appropriate when the goal is guidance, but offer little in the way of understanding. Conversely, regression models have become the de facto standard for clinical trials and causal inference, but may not lead to optimal predictive ability. In situations where the primary goal is a rigorous understanding of causal relationships, that may not matter as much.

1.3.2 Observational Study vs. Experiment

In **experimental studies**, the investigator manipulates some aspect of the subjects' experience and studies its effect on the outcome of interest. For example, here is the NIH's definition of a **clinical trial**:

> A research study in which one or more human subjects are prospectively assigned to one or more interventions (which may include placebo or other control) to evaluate the effects of those interventions on health-related biomedical or behavioral outcomes.

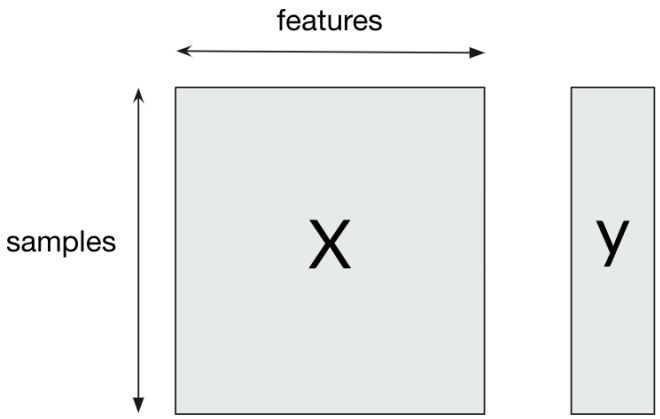
A clinical trial, therefore, is an experiment, because we control the intervention and monitor the effect of that intervention on one or more outcomes. Usually experimental studies employ some type of **randomization** to ensure that comparisons between different intervention groups are fair.

An **observational study**, in contrast, makes no attempt to interfere with its subjects. Instead, these individuals are simply observed, and inferences are made about the associations between different parameters and the outcome(s). Observational study designs and analytic plans are carefully designed to

minimize the effects of different sources of bias that can creep in due to lack of randomization. Although they're not usually referred to using this terminology, virtually all "big data" and machine learning oriented studies in healthcare are observational studies, because they use large datasets that were collected for other purposes.

1.3.3 Types of Machine Learning

This distinction, most often found in discussions of machine learning, refers to the way in which training data is applied to solve a problem. In **supervised learning**, the training data consist of pairs of input features and labels, and the algorithm learns to predict the value of the label from the input features. The general setup for supervised learning looks like this:



In **unsupervised learning**, only the input features are present (i.e. no y) and the algorithm learns to recognize patterns, clusters, or other structure in the inputs. Although they're almost never referred to using this terminology, clinical studies that examine the effect between one or more exposures and an outcome are examples of supervised learning. Studies that attempt to uncover groups, or clusters, of similar patients or samples are examples of unsupervised learning.

There are also two other types of machine learning. In **semi-supervised learning**, a small amount of labeled data is used to create a much larger,

weakly-labeled set of training data that is then fed to a supervised learning algorithm. In **reinforcement learning**, an algorithm is trained with a reward system which provides feedback on the quality of the action the system performs in a given situation instead of (as in supervised learning) simply providing the "right answer".

Chapter 2

The Basics of Classification

Classification is a form of supervised learning in which our goal is to learn a mapping between some features, x, and an output, y. In classification, the output, y, is a category. In **binary classification** (by far the most common), there are only two categories: yes or no, usually represented as "0" (no) or "1" (yes). In **multi-class classification**, there are more than two categories.

To learn an appropriate mapping, we feed **training data** to a **learning algorithm**. Different algorithms learn different types of mappings.

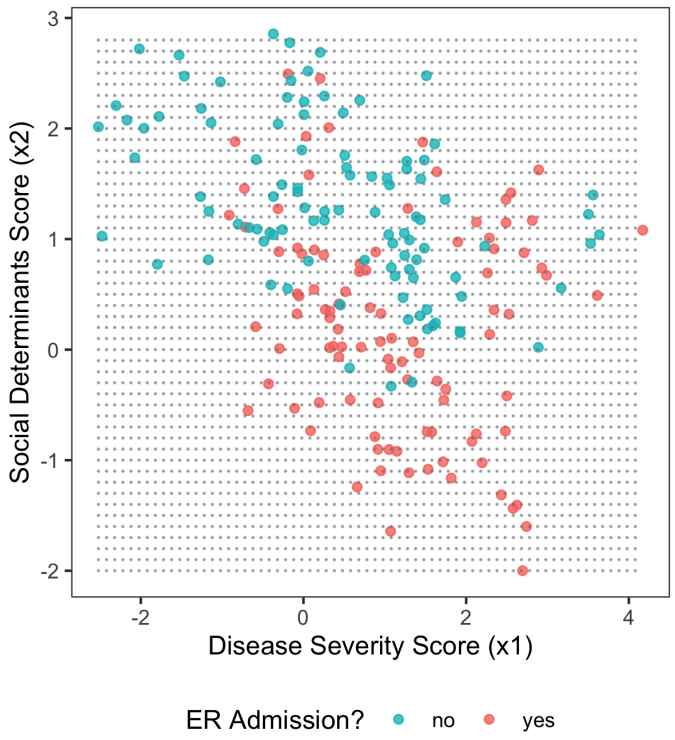
2.1 Definitions

- **Training data:** The data used, along with an appropriate learning algorithm, to create the mapping between input and output. It is composed of **training examples**, a.k.a. **samples**, each consisting of one or more input features and a single output.
- **Test data:** An independent dataset, not used in model training, on which the performance of a trained supervised learning model is evaluated.
- **Feature:** Also known as a **predictor**, or **covariate**, one of the inputs to a supervised learning algorithm.
- **Output:** Also known as the **outcome**, or **label**, the thing you are trying to predict.

- **Feature space:** Envisioning each feature as having its own axis that is orthogonal to all of the other features' axes, the multidimensional space spanned by those axes (or rather: unit vectors in the directions of those axes)
- **Extrapolation:** Making predictions outside the region of the feature space occupied by the training data. This will often lead to errors.

2.2 Visualizing the Classification Problem

Imagine we want to predict whether a patient will be readmitted to the emergency room (ER) within 30 days of hospital discharge. We gather data on two predictors: a disease severity score (x_1), which characterizes the severity of illness, and a social determinants score (x_2), which characterizes the patient's socioeconomic status. We have data on 200 patients.

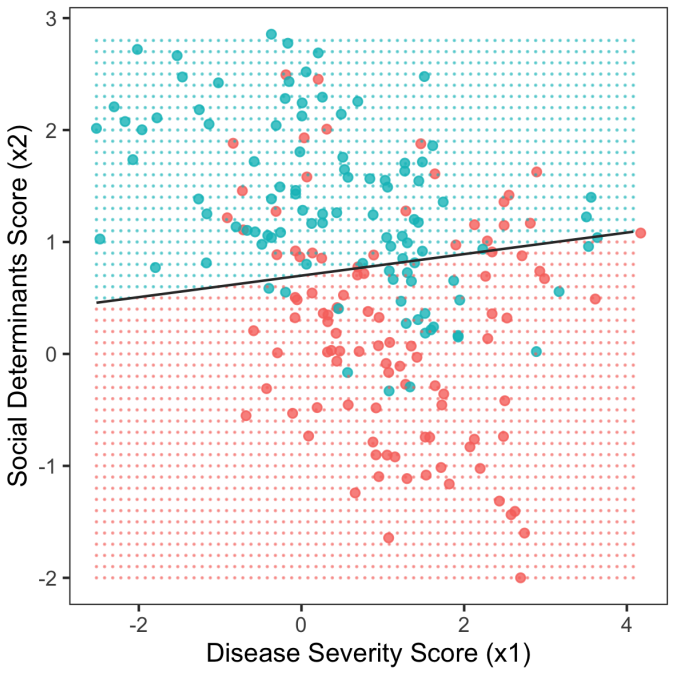


In this figure, the color refers to whether a patient was readmitted (blue = "no", red = "yes"). The location of each point is governed by the patient's disease severity score (x_1, horizontal axis) and social determinants score (x_2, vertical axis). Our goal in classification is to draw a **decision boundary** through this space, on one side of which we will predict that the patient is readmitted, and on the other side not.

2.3 Three Classification Algorithms

2.3.1 Logistic Regression

The simplest decision boundary is, arguably, a line. The logistic regression algorithm simply draws a line[1] through the feature space that divides the positive and negative training examples.



[1] In a higher-dimensional feature space, the decision boundary for logistic regression is a **hyperplane**.

The output of a fitted logistic regression model from R looks like this:

```
Call:
glm(formula = y ~ x1 + x2, family = "binomial", data = df)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.88232  -0.90614  -0.05965   0.86579   2.28489

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.9780     0.2945   3.321 0.000897 ***
x1            0.1344     0.1372   0.980 0.327272
x2           -1.3981     0.2316  -6.035 1.59e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 277.26  on 199  degrees of freedom
Residual deviance: 209.54  on 197  degrees of freedom
AIC: 215.54

Number of Fisher Scoring iterations: 4
```

The equation of the line (or, in higher dimensions, hyperplane) that forms the decision boundary in logistic regression can be obtained by setting the linear sum of coefficients of this model equal to zero.

$$0.9780 + 0.1344x_1 - 1.3981x_2 = 0$$
$$\implies x_2 = \frac{0.9780 + 0.1344x_1}{1.3981}$$

At any point, (x_1, x_2), in the feature space, the model's predicted probability of a positive outcome (i.e. probability of an ER readmission) is related to the coefficients by this equation

$$\log \frac{P[Y = 1]}{1 - P[Y = 1]} = 0.9780 + 0.1344x_1 - 1.3981x_2$$

The decision boundary occurs when $P[Y = 1] = 0.5$ (total uncertainty, e.g. a coin toss). Another way to write this equation is:

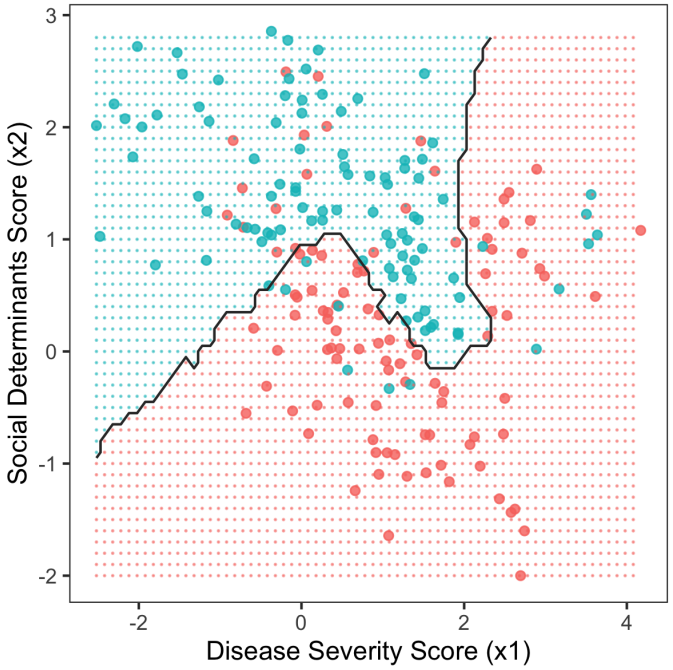
$$P[Y = 1] = \frac{1}{1 + \exp(-(0.9780 + 0.1344x_1 - 1.3981x_2))}$$

The functional form on the right, $1/(1+\exp(-z))$, is called the **logistic function**; this is how logistic regression got its name. We will learn much more about the math behind logistic regression in subsequent chapters.

2.3.2 K Nearest Neighbors (KNN)

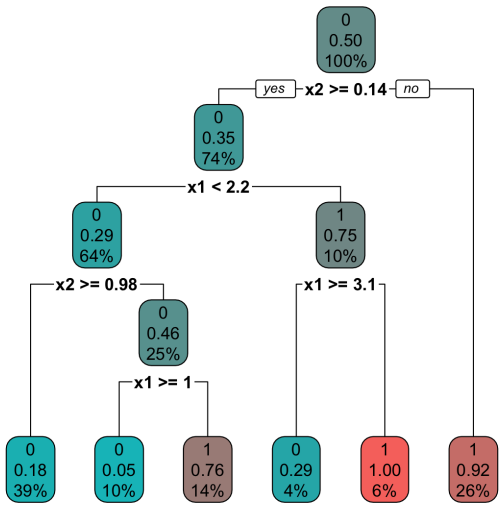
Another – completely different – approach to classification is to start with no assumptions about the shape of the decision boundary. To make a prediction about a new patient, we simply identify the K nearest neighbors to that patient from our training set and allow them to vote on whether or not the new patient will be readmitted. The parameter K must be set independently and is called a **hyperparameter**.

Here is the decision boundary for KNN with $K = 15$:

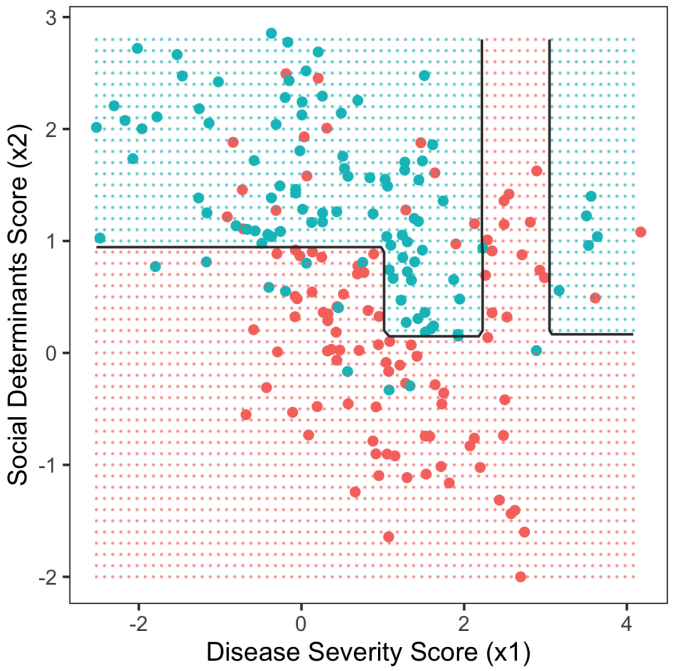


2.3.3 Decision Tree

Finally, we may choose to use our training data to build a decision tree, which will allow us to make predictions on new patients using a series of simple yes/no questions. There are different decision tree learning algorithms, but here is the tree produced by a famous one called CART:



And here is the decision boundary produced by this tree:



Question 2.1

How can you tell, just by looking at these images, which feature (x_1 or x_2) impacts the outcome the most? Which one is it?

Question 2.2

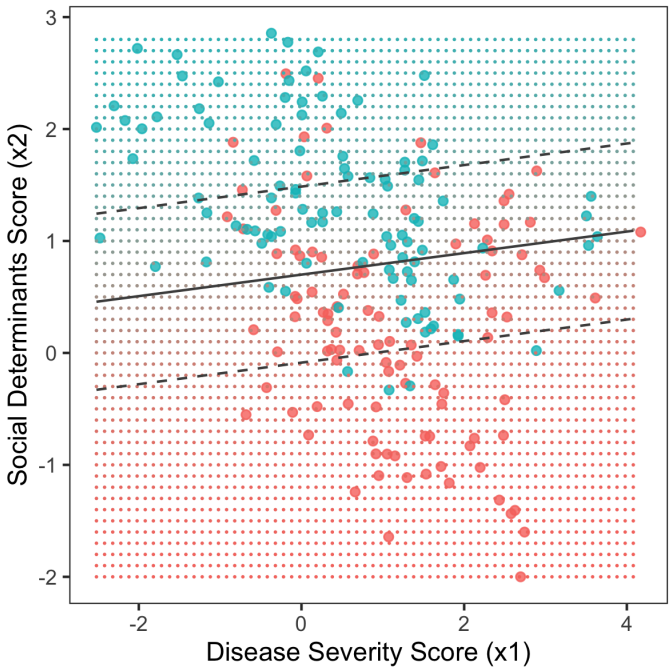
There are six rectangular regions in the picture of the decision tree decision boundary. Each corresponds to one of the six leaves of the tree. Identify all six and which leaves they correspond to on the decision tree.

2.4 Classification with Probabilities

We can think of classification as simply drawing a decision boundary, but underlying each algorithm is a quantitative assessment of each point in the feature space. Each algorithm is, in its own way, able to provide a degree of

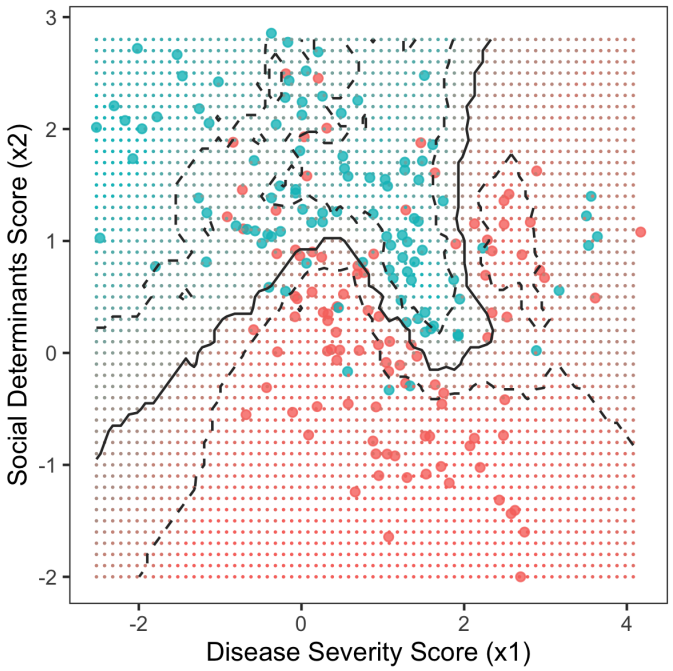
certainty, or **probability**[2], that a point belongs to the positive outcome class.

For example, here is the feature space of the example we just saw, colored by the probability, according to logistic regression, that a sample at each point should be classified as positive (i.e. the patient will be readmitted to the ER):

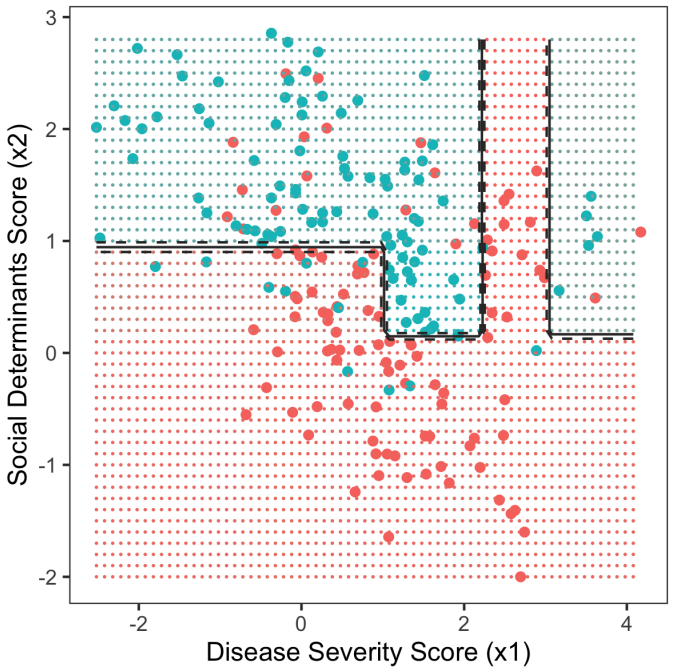


The solid line is the decision boundary, and the dashed lines indicate where the probability of a positive outcome (ER readmission) is 25% (top line) and 75% (bottom line). You can see that the color of the background gets purer red or purer blue the further you get from the decision boundary, but that near the decision boundary, the color is rather murky. That murkiness reflects the algorithm's uncertainty about the outcome. At the decision boundary, it is maximally uncertain. There the probability of a positive outcome is 50%: a coin toss. Here is a similar plot for KNN ($K = 15$):

[2]Pedantic footnote: this is a Bayesian definition of probability, as opposed to a frequentist definition. More on that later.



You can see that the shapes of the 25% and 75% probability lines have much more complex shapes than for logistic regression, but the story is the same: you have regions of pure blue or red, where the algorithm is certain, and you have a murky region near the decision boundary. Now, finally, here is the same plot for the decision tree:



The color of the background in the regions corresponding to the six leaves of the tree is the same throughout each region. That's because the probability in each rectangular region (corresponding to each leaf of the tree) is constant. It equals the number of red dots in that region divided by the total number of dots.

Question 2.3

What are the advantages and disadvantages of each algorithm?

1. Logistic regression?
2. KNN ($K = 15$)?
3. Decision tree?

Question 2.4

What makes a good classification algorithm? Consider issues of accuracy, generalizability, and speed (both to train the algorithm and to use it to make predictions on new samples).

Chapter 3

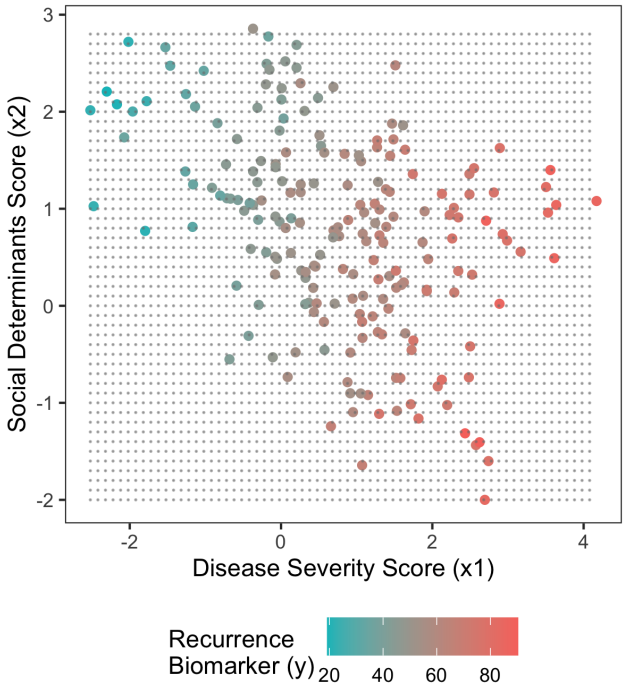
The Basics of Regression

Classification is a form of supervised learning in which the outcome is a category. **Regression** is another form of supervised learning in which the outcome is a numeric value. For example, it may be a lab value, physical characteristic (height, weight, etc.), or numeric measurement (e.g. oxygen saturation).

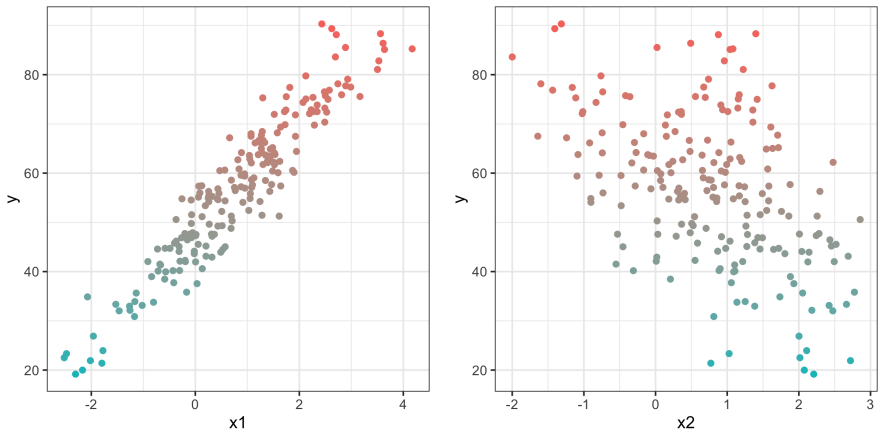
3.1 Visualizing the Regression Problem

Let's consider the same setup from Section 2.2 but this time with a quantitative outcome: a "recurrence biomarker" that indicates the likelihood of recurrence of disease.

Again, we have data on two predictors: a disease severity score (x_1), which characterizes the severity of the illness for which the patient was originally treated, and a social determinants score (x_2), which characterizes a patient's socioeconomic status. We have measurements of x_1 and x_2 on the same 200 patients as in Section 2.2.



This is a plot of the data in a single plane. The color represents the value of the recurrence biomarker – the height of the point above the plane. We want to design a model that will predict the value of the biomarker (y) based on the values of the two predictors, x_1 and x_2. These plots show the **univariate** relationship of each predictor with the outcome.



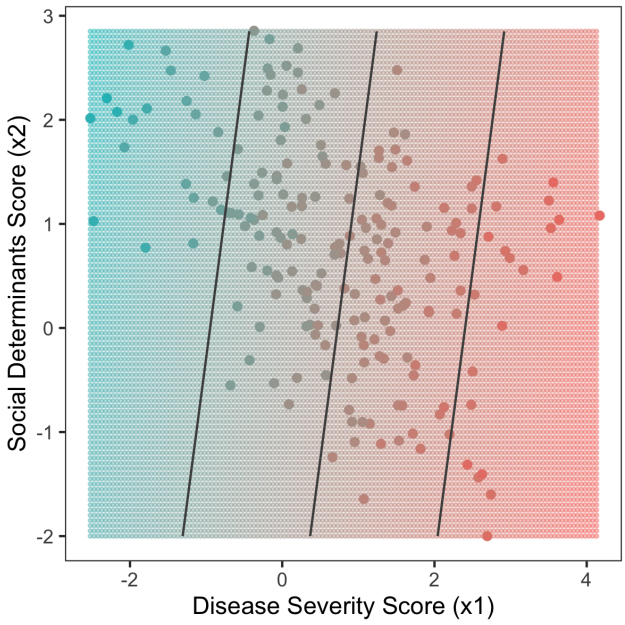
Question 3.1

Which of the two predictors, x_1 or x_2, appears to more strongly influence the value of the recurrence biomarker? Explain your reasoning using evidence from the preceding three plots.

3.2 Three Regression Algorithms

3.2.1 Linear Regression

The regression analogue of logistic regression is **linear regression**[1]. Linear regression creates a hyperplane that slices through the cloud of training datapoints such that it passes as close as possible, on average, to the data. This is, of course, easiest to see when the feature space is two-dimensional, as it is here:



[1]The terminology here is confusing. When we learn about generalized linear models in Chapter 12, you'll see why logistic regression has the word "regression" in its name even though it's a classification algorithm.

The three lines shown here sit on the hyperplane learned by the linear regression model. They are located at heights corresponding to the 25th, 50th, and 75th percentiles of the outcome, y (the biomarker value). The plane tilts downward toward the upper left corner of the $x_1 \times x_2$ grid and upward toward the bottom right corner. It may be helpful to visualize grabbing the $x_1 \times x_2$ plane and rotating/translating it so that it passes through the middle of the training data. Here is a summary of the trained linear regression model:

```
Call:
lm(formula = y ~ x1 + x2, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-11.9218  -3.1032   0.2891   2.8316  12.5813

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  49.8600     0.5370  92.844  < 2e-16 ***
x1           10.4372     0.2855  36.555  < 2e-16 ***
x2           -1.8824     0.3609  -5.215 4.63e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.769 on 197 degrees of freedom
Multiple R-squared:  0.9026,    Adjusted R-squared:  0.9016
F-statistic: 912.4 on 2 and 197 DF,  p-value: < 2.2e-16
```

At each point (x_1, x_2) in the feature space, the model's predicted value of the recurrence biomarker, $\hat{y}$, is

$$\hat{y} = 49.8600 + 10.4372x_1 - 1.8824x_2$$

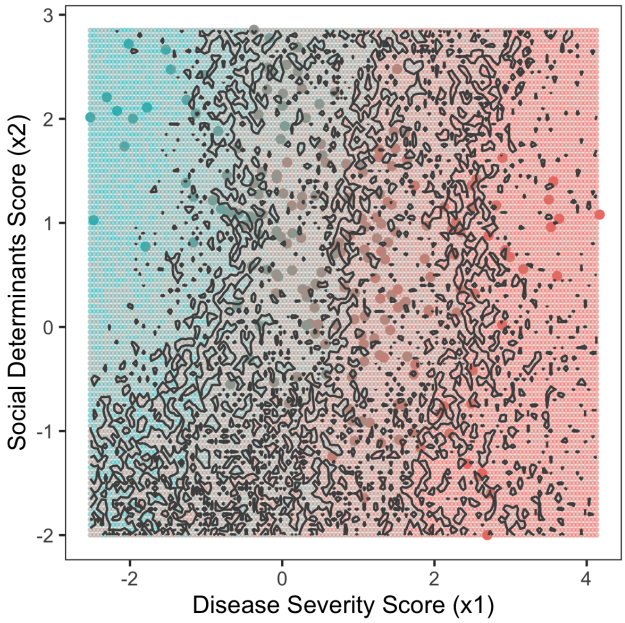
Question 3.2

Compare and contrast the output from the linear regression model with the output from the logistic regression model in Chapter 2. What looks the same? What looks different? What is being predicted in each case?

3.2.2 K Nearest Neighbors (KNN)

Regression using KNN works very similarly to KNN for classification. In classification, we allow the nearest K points to vote on the label of a new test

point. In regression, we **interpolate** between the values of the surrounding points to come up with the value of y for a test point. Typically this is done just by averaging the y values of the nearest K points, but you can also do something more sophisticated, like weight their contributions by distance to the test point. Here is a contour plot of the regression surface produced by KNN ($K = 15$) for our example:

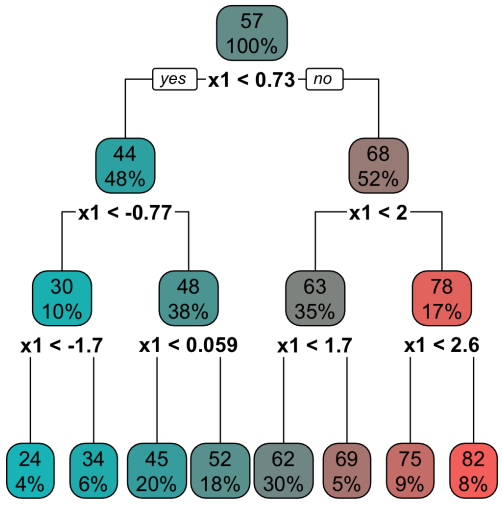


The contours are again drawn at the 25th, 50th, and 75th percentiles of the outcome, y. This looks like a bit of a mess compared to the linear regression plot, but at the same time, the KNN algorithm is able to capture arbitrarily complex relationships between x_1, x_2, and y that can be missed by other regression algorithms.

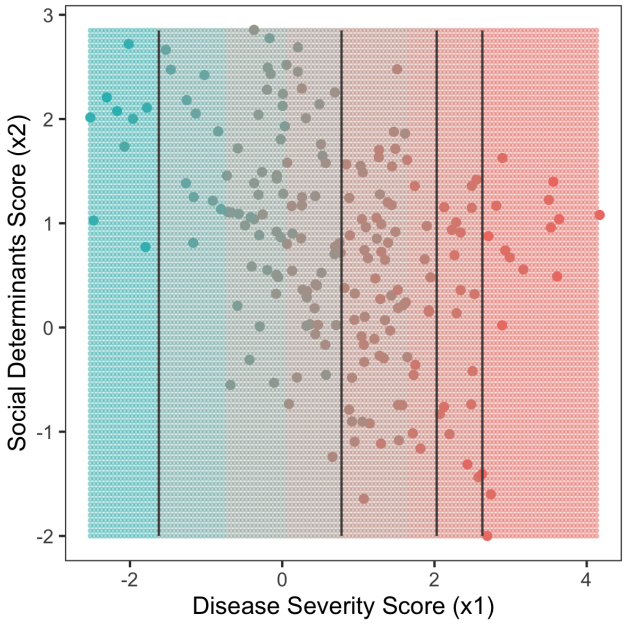
3.2.3 Decision Tree

Decision tree regression is similar to decision tree classification except that the output at each leaf is not a class label or the probability of membership in the positive training class (both of which are shown on the tree in Section 2.3.3),

but a numeric value. That value corresponds to the mean outcome value for the points in that leaf.



The predicted biomarker values for a decision tree trained on this dataset (created using the `rpart` package in R with default parameters) are shown here:



You can see that the decision tree always chooses to split on x_1, the disease severity score, rather than x_2. Revisit Question 3.1 to remind yourself of why this is. The regression surface produced by the decision tree looks like a set of stairs climbing higher and higher as one moves from left to right across the $x_1 \times x_2$ plane. The predicted value of y, the recurrence biomarker, is constant within each stair.

Question 3.3

Compare this decision tree with the decision tree for the classification problem in Chapter 2. What is the same? What is different?

Question 3.4

This **regression tree** has eight leaves. What region of the feature space does each leaf correspond to?

Question 3.5

What are the advantages and disadvantages of each of these three regression algorithms (linear regression, KNN, regression tree)?

Chapter 4

Probability Distributions

Many of the methods we will examine in these workshops depend on basic concepts from probability theory. For example, linear and logistic regression are members of a class of supervised learning algorithms called **generalized linear models** (see Chapter 12) which make assumptions about the type of probability distribution followed by the outcome variable. Decision trees use a concept called **entropy** (see Chapter 7), whose mathematical formulation depends on the probability distribution underlying the outcome. Many **hypothesis tests** (see Chapter 6) likewise rely on probabilistic assumptions about the data. Probability is everywhere.

The following sections review some key probability concepts – in an extremely hand-wavey and non-rigorous way – and the properties of some of the most common probability distributions you will encounter in machine learning and statistics.

4.1 Definitions

A **probability distribution** is just a mathematical function that provides the relative likelihoods of various possible outcomes of an observation. We call the quantity that is being observed a **random variable**. Probability distributions can be discrete or continuous. The random variable involved can be a number, a vector of numbers, a category/class, etc. The **sample space** is the set of all

possible outcomes. The integral (or sum) of the probability distribution over the entire sample space is 1.0. You will often hear probability distributions for continuous random variables referred to as **probability densities**.

Probability distributions are grouped into families that are characterized by their overall shapes. These families contain **parameters** that, when varied, produce different distributions. Specific probability distributions from within a single family can often look quite different.

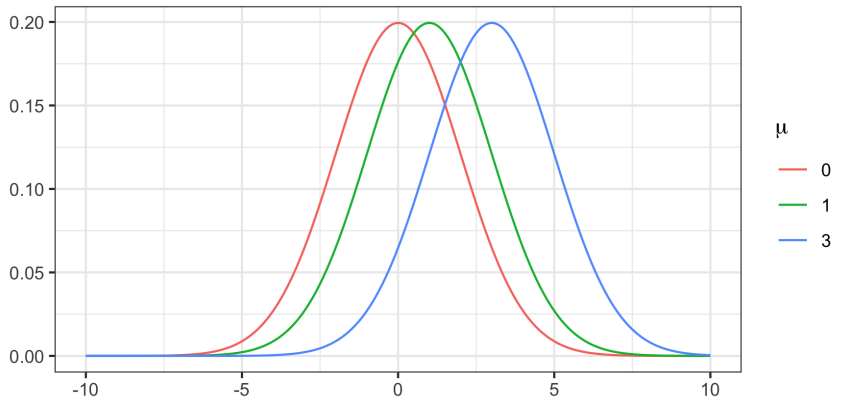
We use the notation $E[x|\theta]$ to refer to the **expected value**, or mean, of a distribution, given its parameter(s), θ. There can be more than one parameter, and it will not always be called θ; this is just an example. We use the notation $\text{var}(x|\theta)$ to refer to the **variance**, or spread, of a distribution around its mean.

4.2 Normal Distribution

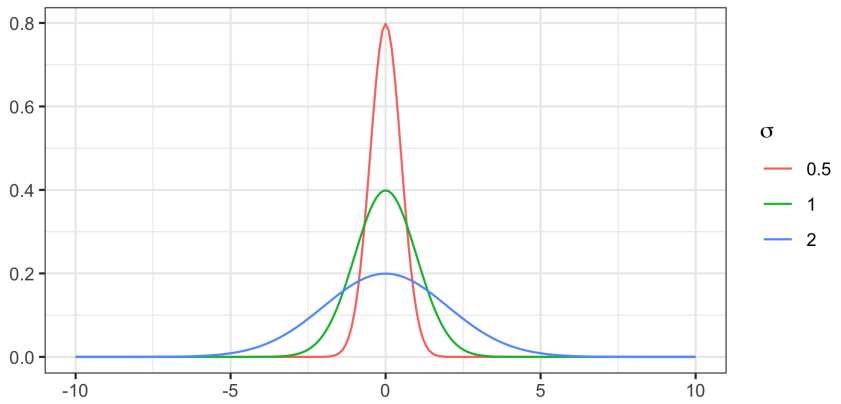
Also called the **Gaussian distribution**, the normal distribution is probably the most well-known continuous probability distribution. It has the following properties:

$$p(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad E[x|\mu,\sigma] = \mu \qquad \text{var}(x|\mu,\sigma) = \sigma^2$$

where $x \in \mathbb{R}$. We will abbreviate the normal distribution as $\mathcal{N}(\mu,\sigma)$. The value of μ changes the position of the center of the normal distribution.



The value of σ changes the width of the normal distribution.



Question 4.1

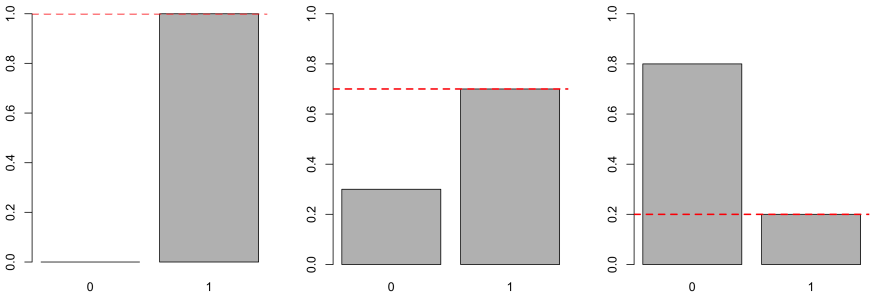
List 5 random variables from medicine or biology that should follow normal distributions.

4.3 Bernoulli Distribution

The **Bernoulli distribution** is a discrete probability distribution with the following properties:

$$p(x|\mu) = \mu^x(1-\mu)^{1-x} \qquad E[x|\mu] = \mu \qquad \text{var}(x|\mu) = \mu(1-\mu)$$

where $x \in \{0, 1\}$. It is used to model events where the outcome is yes/no. Think of it as a weighted coin, with μ the probability that the coin comes up "heads" on a single toss. Here are three Bernoulli distributions with (from left to right) $\mu = 1.0, 0.7, 0.2$. The number along the bottom is x, which can only be 0 or 1.



The **categorical distribution** is a generalization of the Bernoulli distribution to an outcome with more than two levels. The categorical distribution looks like this:

$$p(x|\phi_1, \ldots, \phi_K) = \phi_1^{\mathbb{I}(x=1)} \phi_2^{\mathbb{I}(x=2)} \cdots \phi_K^{\mathbb{I}(x=K)}$$

where $\sum_{k=1}^{K} \phi_k = 1$. The term $\mathbb{I}(x = j)$ is an **indicator**. It equals 1 if $x = j$ and 0 otherwise. For example, $\mathbb{I}(x = 2)$ is 1 if $x = 2$ and 0 otherwise.

Question 4.2

List 5 random variables from medicine or biology that should follow Bernoulli distributions.

4.4 Binomial Distribution

The **binomial distribution** models the number of positive outcomes, x, out of n independent[1] Bernoulli trials, each of which is positive with probability μ. This distribution has the following properties, with $x \in \{0, \ldots, n\}$:

$$p(x|n, \mu) = \binom{n}{x} \mu^x (1-\mu)^{n-x} \qquad E[x|\mu] = n\mu \qquad \text{var}(x|\mu) = n\mu(1-\mu)$$

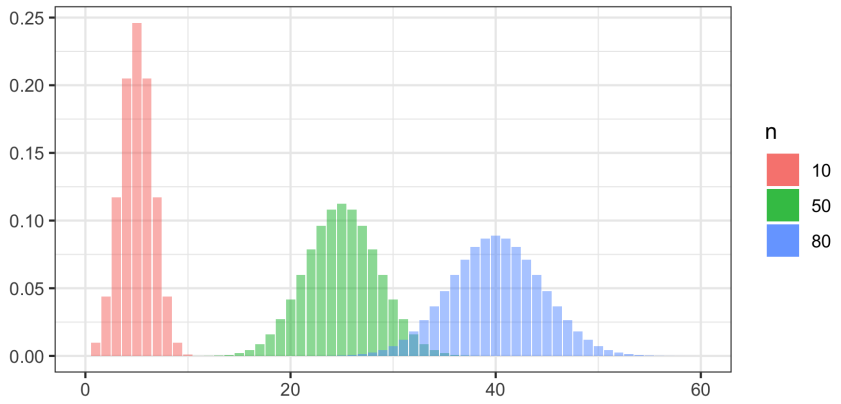
where the notation $\binom{n}{x}$ is defined as:

$$\binom{n}{x} = \frac{n!}{x!(n-x)!}.$$

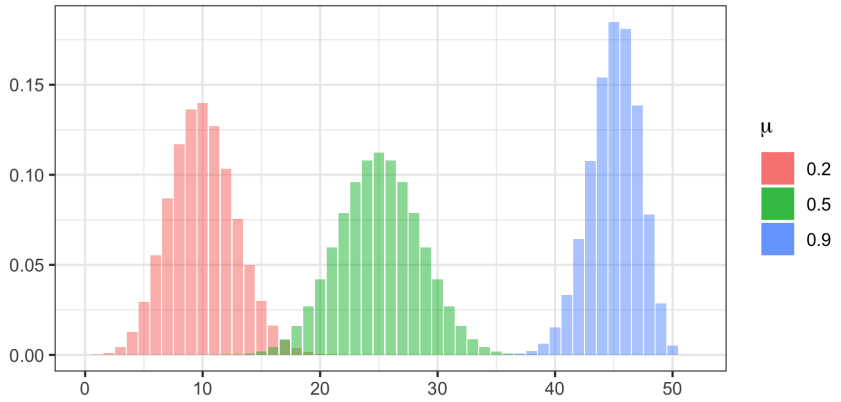
[1]The word **independent** just means that the outcome of one trial does not influence the outcome of any other trial.

This notation denotes the number of ways it is possible to choose x things out of a group of n things, where the ordering doesn't matter. The exclamation point denotes the **factorial function**: $x! = x(x-1)(x-2)\cdots(2)(1)$.

The shape of the binomial distribution is governed by the values of n and μ. Here, we vary n but keep μ constant at 0.5:



And here we vary μ but keep n constant at 50:



Question 4.3

List 5 random variables from medicine or biology that should follow binomial distributions.

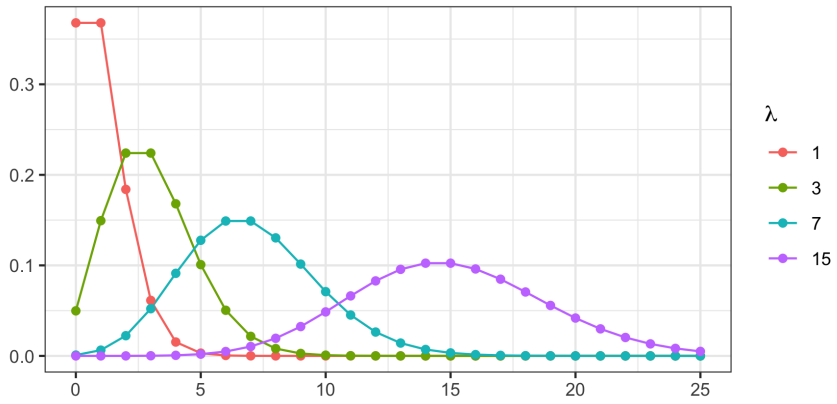
4.5 Poisson Distribution

The **Poisson distribution** is a probability distribution that is often used to model discrete quantitative data, such as counts. It has the following properties:

$$p(x|\lambda) = \frac{e^{-\lambda}\lambda^x}{x!} \qquad E[x|\lambda] = \lambda \qquad \text{var}(x|\lambda) = \lambda$$

where $x \in \{0, 1, 2, \dots\}$. Below are four examples of Poisson distributions. If events of a particular type occur continuously and independently at a constant rate (**Poisson process**), the number of events within a time window of fixed width will be distributed according to the Poisson distribution, with rate parameter λ proportional to the width of the window.

Situations where the population size, n, is large, the probability of an individual event, p, is small, but the expected number of events, np, is moderate (say five or more) can generally be modeled using a Poisson distribution with $\lambda = np$.



Question 4.4

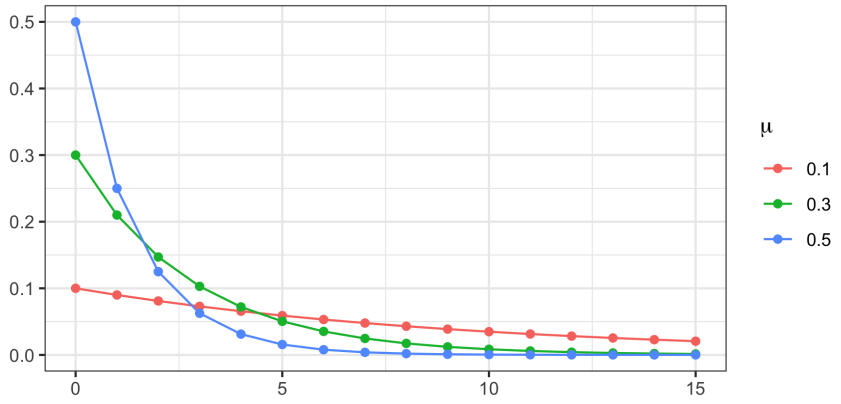
List 5 random variables from medicine or biology that should follow Poisson distributions.

4.6 Geometric

The **geometric distribution** models the number of failures in a sequence of Bernoulli trials before the first success. It has the following properties:

$$p(x|\mu) = (1-\mu)^x \mu \qquad E[x|\mu] = \frac{1-\mu}{\mu} \qquad \text{var}(x|\mu) = \frac{1-\mu}{\mu^2}$$

for $x \in \{0, 1, 2, \dots\}$, where μ refers to the probability (in the Bernoulli trial) that the trial is a success. Some examples of geometric distributions with different μ are shown below:



Question 4.5

List 5 random variables from medicine or biology that should follow geometric distributions.

4.7 Exponential

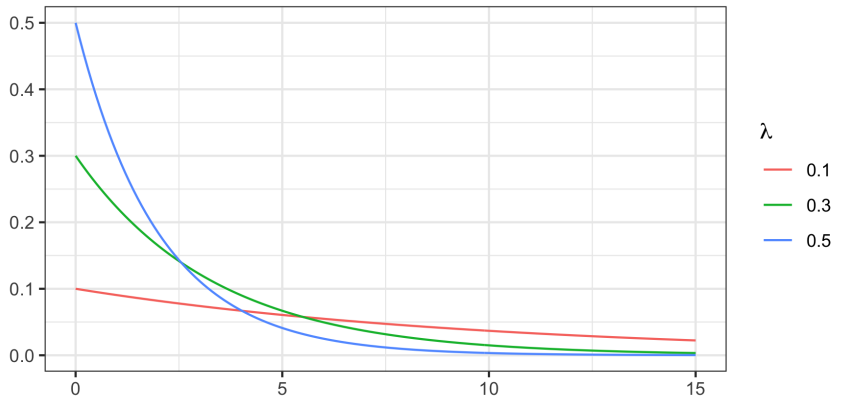
The **exponential distribution** is a continuous probability distribution that models waiting times between events that happen independently and continuously at a constant rate (Poisson process), as well as many other random

variables[2]. It has the following properties:

$$p(x|\lambda) = \lambda e^{-\lambda x} \qquad E[x|\lambda] = \frac{1}{\lambda} \qquad \text{var}(x|\lambda) = \frac{1}{\lambda^2}$$

where $x \in \mathbb{R}^+$ (x is a positive real number, or zero). The exponential distribution is the continuous analogue of the geometric distribution. It is memoryless, which means that the distribution of a waiting time until an event does not depend on how much time has elapsed already.

Here are some different exponential distributions. Compare them to the geometric distribution, above.



Question 4.6

List 5 random variables from medicine or biology that should follow exponential distributions.

4.8 Chi-Squared Distribution

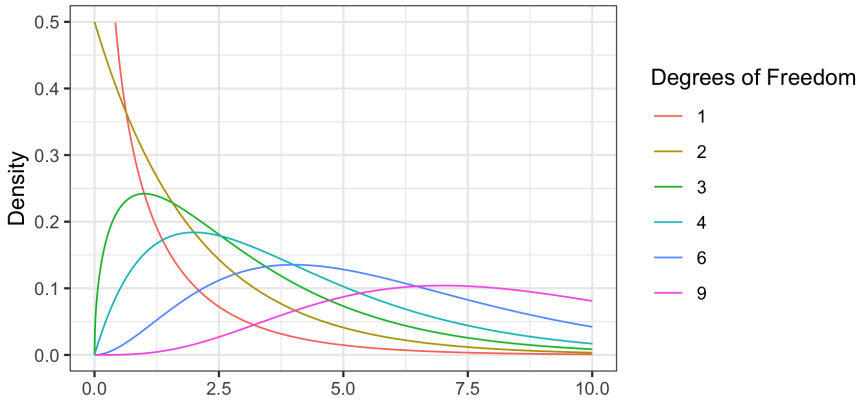
How this distribution arises:

[2]For example, in an epidemiologic model of an infectious process like COVID-19 community spread, exponential waiting times are often used to model transitions between the susceptible, exposed, infectious, and recovered compartments in the model.

1. If $Z \sim \mathcal{N}(0,1)$, the distribution of $U = Z^2$ is called the chi-squared distribution with one degree of freedom.

2. If $U_1, U_2, \ldots, U_k$ are independent χ^2_1 random variables, their sum, $V = \sum_{i=1}^{k} U_i$ follows χ^2_k, a chi-squared distribution with k degrees of freedom.

You'll often see the chi-squared distribution used as the sampling distribution for the sample variance in a variety of statistical hypothesis tests. It looks like this:



The parameter k, the **degrees of freedom**, controls the shape of the chi-squared distribution. The actual formula for the chi-squared distribution looks a bit intimidating, but I'm including it here so you can compare it to the other distributions we've seen:

$$p(x|k) = \frac{1}{2^{k/2}\Gamma(k/2)} x^{k/2-1} e^{-x/2}$$

$$E[x|k] = k \qquad \text{var}(x|k) = 2k$$

The gamma function shown in the denominator of the probability density,

$$\Gamma(z) = \int_0^\infty x^{z-1} e^{-x} dx,$$

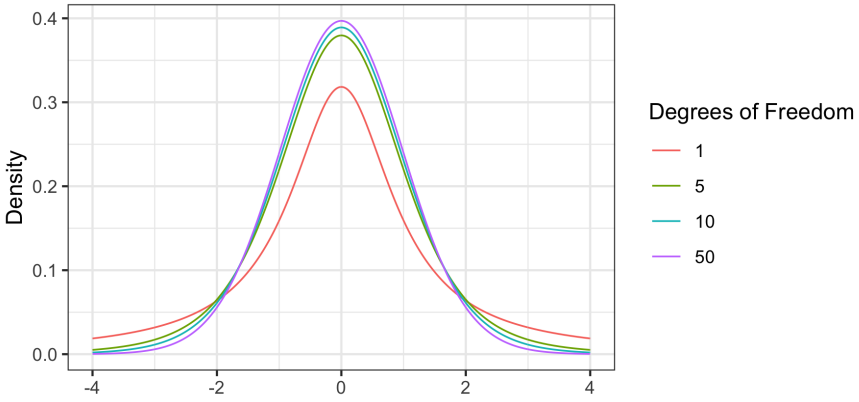
is a generalization of the factorial function to complex numbers. For any positive integer n, $\Gamma(n) = (n-1)!$.

4.9 Student's T Distribution

If $Z \sim \mathcal{N}(0,1)$ and $U \sim \chi^2_k$ and Z and U are independent,

$$T = \frac{Z}{\sqrt{U/k}} \sim t_k$$

or in words, the statistic T follows a t-distribution with k degrees of freedom. The T distribution plays an important role in a family of statistical hypothesis tests called T-tests.



Again, the functional form of the T distribution is a bit intimidating, but I'm including it for completeness:

$$p(x|k) = \frac{\Gamma\left(\frac{k+1}{2}\right)}{\sqrt{k\pi}\,\Gamma\left(\frac{k}{2}\right)}\left(1 + \frac{x^2}{k}\right)^{-\frac{k+1}{2}}$$

$$E[x|k] = 0 \quad \text{for } k > 1; \text{ otherwise undefined}$$

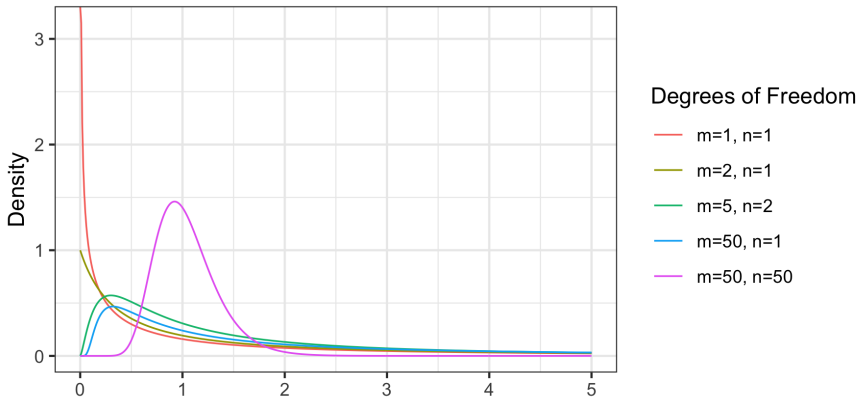
$$\text{var}(x|k) = \begin{cases} \frac{k}{k-2} & k > 2 \\ \infty & 1 < k \leq 2 \\ \text{undefined} & \text{otherwise} \end{cases}$$

4.10 F Distribution

If U and V are independent χ^2 random variables with m and n degrees of freedom,

$$W = \frac{U/m}{V/n} \sim F_{m,n}$$

or in words, the statistic W follows an F distribution with m and n degrees of freedom. I'm not writing out the functional form of the F distribution here because it's too awful-looking, but graphically it looks like this:



Note that if $T \sim t_k$, then $T^2 \sim F_{1,k}$. The F-distribution plays an important role in a class of statistical analysis techniques called **ANalysis Of VAriance**, or **ANOVA**.

Question 4.7

For each of the following experimental conditions, which distribution (from those listed above) provides the best model for how the data $x^{(1)}, \ldots, x^{(n)}$ are generated?

(a) You are observing several patients' skin in a clinical study to see how long it takes them to develop a rash. You take a picture each day. Let $x^{(i)}$ be the number of days of *no rash* before the rash occurs.

Patient ID (i)	$x^{(i)}$
1	4
2	1
3	0
4	2
5	2
6	4
7	3
8	1
9	0
10	1

(b) Same situation as above except that instead of taking a picture each day, the patient texts you at the moment he/she observes a rash. The data look like this, where $x^{(i)}$ is the time (in days) at which patient i develops a rash:

Patient ID (i)	$x^{(i)}$
1	2.25
2	3.43
3	0.68
4	0.04
5	3.78
6	5.65
7	2.88
8	3.88
9	2.83
10	1.87

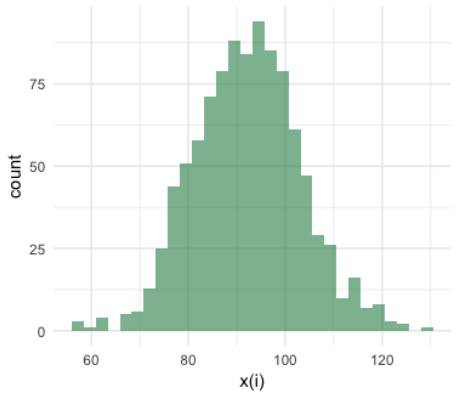
(c) Imagine you are Ladislaus Bortkiewicz, and you are modeling the number of persons killed by mule or horse kicks in the Prussian army per year. You have data from the late 1800s over the course of 20 years. Let $x^{(i)}$ be the number of people killed in year i.

Year (i)	$x^{(i)}$	Year (i)	$x^{(i)}$
1	8	11	9
2	10	12	7
3	5	13	10
4	3	14	12
5	10	15	8
6	8	16	7
7	7	17	8
8	2	18	8
9	6	19	10
10	11	20	7

(d) Every year, 10 scientists go to the same geographic area (same Lyme prevalence) and they each collect 40 ticks. They test each tick for Lyme disease and record the number of ticks that have Lyme. Let $x^{(i)}$ be the number of ticks with Lyme in the ith scientist's bunch.

Scientist ID (i)	$x^{(i)}$
1	8
2	9
3	14
4	15
5	12
6	7
7	6
8	8
9	8
10	14

(e) You have waist circumference data on 1045 men aged 70 and above (see Dey's 2002 paper in the Journal of the American Geriatric Society). It looks like this:


75
50
25
0
count
60
80
100
120
x(i)

Chapter 5

The Basics of Maximum Likelihood Estimation

Beneath our discussions of classification, regression, and probability distributions in Chapters 2, 3, and 4 lies the tricky problem of **model fitting**. We've seen what classification and regression models look like, but we still haven't addressed how to fit these models using training data.

Linear and logistic regression models are fit using a technique called **maximum likelihood (ML) estimation**, in which the model parameters are adjusted to maximize the joint probability of the observed data, or likelihood, given the model.

For example, consider the five different datasets from Question 4.7. In each case, you have some data and an assumption about which probability distribution the data are drawn from. The job of maximum likelihood estimation is to use the data to identify the correct distributional parameters, such as μ and σ (in the case of the normal distribution) or λ (in the case of the Poisson distribution). This process is a type of **statistical inference**.

5.1 The Likelihood and Log-Likelihood

Let $p(x|\theta)$ be the probability distribution that governs our data. Here, θ stands in for all of the parameters we want to fit.

If we draw independent[1] samples from $p(x|\theta)$, the **joint probability density function** for all n observations is:

$$p(x^{(1)}, x^{(2)}, \ldots, x^{(n)}|\theta) = \prod_{i=1}^{n} p(x^{(i)}|\theta).$$

Since the data are known but the parameter(s) θ are unknown, we will view this quantity as a function of θ. This is just a change in notation:

$$\mathcal{L}(\theta) = \prod_{i=1}^{n} p(x^{(i)}|\theta).$$

The higher the joint probability of the data (the more "likely" the data are) given θ, the higher the value of this function. We call $\mathcal{L}(\theta)$ the **likelihood**[2]. Frequently we will want to work with the logarithm of the likelihood, which we call the **log-likelihood**, because it has some nice properties, including allowing us to manipulate sums instead of products[3]:

$$\log \mathcal{L}(\theta) = \sum_{i=1}^{n} \log p(x^{(i)}|\theta).$$

In maximum likelihood estimation, we seek to find the θ for which the likelihood (or log-likelihood) is maximized. We do this by taking derivatives

[1] Independent sampling just means that the values of different samples do not depend on each other. When the samples are drawn independently from the same distribution, their joint probability density is just the product of the individual probability densities (which are all the same).

[2] The distributions we have discussed so far are from a broad family of probability distributions called the **exponential family**. One of the properties of this family is that the log-likelihood is concave. Practically speaking, this means that if we maximize the log-likelihood by setting derivatives equal to zero, we are guaranteed to (a) get only one solution, and (b) find a maximum (not a minimum or an inflection point).

[3] Note that if the function $f(z)$ has a maximum at z', the function $\log f(z)$ will also have a maximum at z', because the logarithmic function is monotonically increasing. So we will get the same parameter estimate(s) either way.

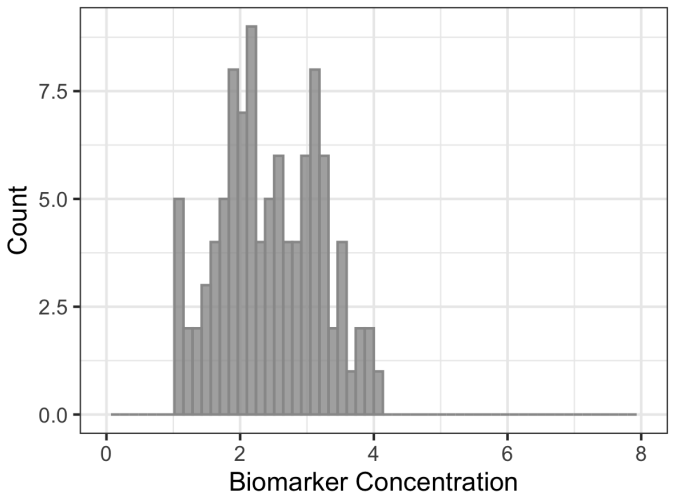
of the log-likelihood with respect to the various parameters and setting them equal to zero. The best-fit parameter estimates obtained in this way are called the **maximum likelihood estimates (MLEs)**.

Question 5.1

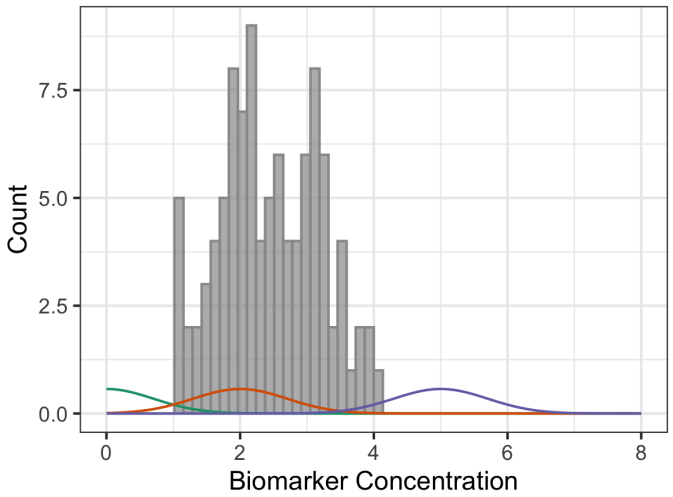
What are some reasons why we might want to fit data to a probability distribution?

5.2 Example: Fitting Data to a Normal Distribution

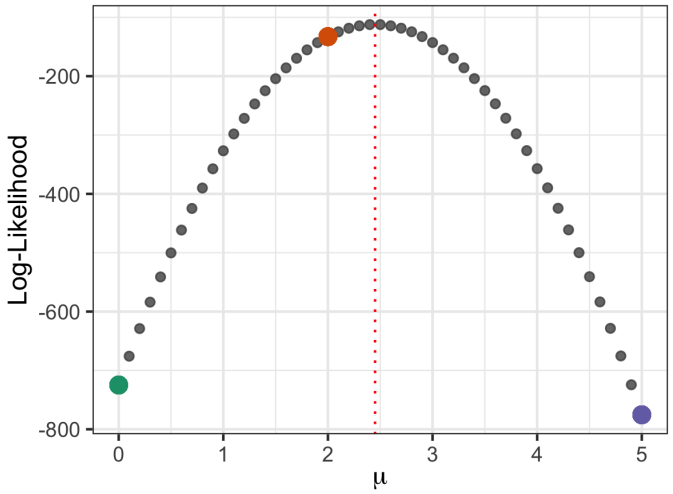
Imagine you have some data from a lab test that measures the concentration of a particular biomarker. You have data from 100 different subjects. A histogram of the raw data looks like this:



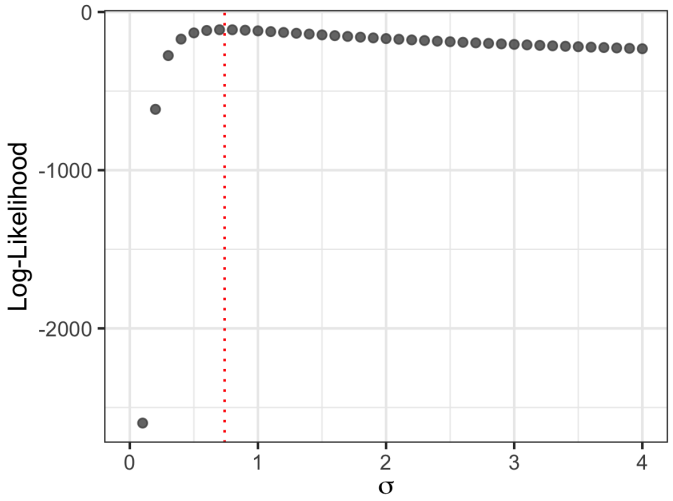
You want to find the normal distribution that best describes these data so you can create a reference distribution for this lab test. To do this, think about trying out several distributions with different values of μ and σ and choosing the one that maximizes the log-likelihood. For example, here are three different normal distributions with different values of μ and $\sigma = 0.7$:



Here is what happens to the log-likelihood as you vary μ. The log-likelihoods of the three distributions shown in the plot above are shown as dots with their corresponding colors, and the maximum likelihood estimate is shown as a vertical dotted line.



Now, here's what happens to the log-likelihood when we vary σ, keeping μ fixed at its maximum likelihood estimate from the graph above. Again, the maximum likelihood estimate is shown as a vertical dotted line.



For the record, I simulated these data from a normal distribution with $\mu = 2.5$ and $\sigma = 0.75$. The maximum likelihood estimates obtained from this dataset are $\hat{\mu} = 2.45$ and $\hat{\sigma} = 0.74$.

5.3 Analytical Calculations of MLEs

In some simple cases, the MLEs can be calculated analytically. We will now go through a bunch of examples of how to find the MLEs of the probability distributions we saw in Chapter 4.

5.3.1 Bernoulli Distribution

The Bernoulli distribution is described in Section 4.3. Our goal is to find the parameter, μ, of this distribution, given some observed data, $x^{(1)}, \ldots, x^{(n)}$. The data will consist of a list of 1s and 0s, since Bernoulli random variables can only take the values 0 or 1.

To find $\hat{\mu}$, our MLE for μ, we first write down the log-likelihood:

$$\begin{aligned}
\log \mathcal{L}(\mu) &= \sum_{i=1}^{n} \log p(x^{(i)}|\mu) \\
&= \sum_{i=1}^{n} \log \left(\mu^{x^{(i)}} (1-\mu)^{1-x^{(i)}} \right) \\
&= \sum_{i=1}^{n} \left[x^{(i)} \log(\mu) + (1-x^{(i)}) \log(1-\mu) \right]
\end{aligned}$$

Then we take the derivative of the log-likelihood with respect to μ:

$$\frac{d}{d\mu} \log \mathcal{L}(\mu) = \sum_{i=1}^{n} \left[\frac{x^{(i)}}{\mu} - \frac{1-x^{(i)}}{1-\mu} \right]$$

The MLE of μ will occur when the likelihood is maximized, which happens when the first derivative equals zero. So to solve for $\hat{\mu}$, we set the derivative equal to zero and rearrange:

$$\begin{aligned}
\sum_{i=1}^{n} \left[\frac{x^{(i)}}{\hat{\mu}} - \frac{1-x^{(i)}}{1-\hat{\mu}} \right] = 0 \implies (1-\hat{\mu}) \sum_{i=1}^{n} x^{(i)} &= \hat{\mu} \sum_{i=1}^{n} (1-x^{(i)}) \\
\implies \boxed{\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}}
\end{aligned}$$

We see that the MLE, $\hat{\mu}$, is simply the sum of our data – i.e. the number of data points where the outcome is 1 – divided by the total number of observations.

This makes sense: if you want to know the probability that a coin will come up heads, a good way to estimate it is to flip the coin a bunch of times and calculate the fraction of observations in which the coin comes up heads.

5.3.2 Binomial Distribution

The binomial distribution is described in Section 4.4. We will make one notational change from that section, which is to call the number of Bernoulli trials m instead of n, since we are using n to refer to the number of data samples. To keep things simple, we will assume that m is a known quantity.

As before, we first write down the log-likelihood:

$$\begin{aligned}\log \mathcal{L}(\mu) &= \sum_{i=1}^{n} \log p(x^{(i)}|m,\mu) \\ &= \sum_{i=1}^{n} \log \left[\binom{m}{x} \mu^x (1-\mu)^{m-x} \right] \\ &= \sum_{i=1}^{n} \left[\log(m!) - \log(x!) - \log((m-x)!) + x^{(i)} \log(\mu) + (m - x^{(i)}) \log(1-\mu) \right]\end{aligned}$$

Then we take the derivative of the log-likelihood with respect to μ:

$$\frac{d}{d\mu} \log \mathcal{L}(\mu) = \sum_{i=1}^{n} \left[\frac{x^{(i)}}{\mu} - \frac{m - x^{(i)}}{1-\mu} \right]$$

We set this equal to zero and solve for $\hat{\mu}$ (the maximum likelihood estimate of μ):

$$\begin{aligned}\sum_{i=1}^{n} \left[\frac{x^{(i)}}{\hat{\mu}} - \frac{m - x^{(i)}}{1-\hat{\mu}} \right] = 0 \implies (1-\hat{\mu}) \sum_{i=1}^{n} x^{(i)} &= \hat{\mu} \sum_{i=1}^{n} (m - x^{(i)}) \\ \implies \boxed{\hat{\mu} = \frac{1}{nm} \sum_{i=1}^{n} x^{(i)}}\end{aligned}$$

Question 5.2

Interpret the MLE for the parameter, μ, of a binomial distribution, assuming fixed m (number of trials). Does the MLE for μ make intuitive sense to you? Think through a few of your examples from Question 4.3.

5.3.3 Normal Distribution

The normal distribution is described in Section 4.2. We will follow the same procedure as in the previous two sections, except that now we have two parameters to solve for, μ and σ, instead of one. First, we write down the

log-likelihood:

$$\begin{aligned}\log \mathcal{L}(\mu, \sigma) &= \sum_{i=1}^{n} \log p(x^{(i)} | \mu, \sigma) \\ &= \sum_{i=1}^{n} \log \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x^{(i)}-\mu)^2}{2\sigma^2}} \right) \\ &= -\frac{n}{2} \log(2\pi) - \frac{n}{2} \log \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} \left(x^{(i)} - \mu \right)^2\end{aligned}$$

To find the MLE for μ, we take the derivative of the log-likelihood with respect to μ:

$$\frac{\partial}{\partial \mu} \log \mathcal{L}(\mu, \sigma) = \frac{1}{\sigma^2} \sum_{i=1}^{n} \left(x^{(i)} - \mu \right)$$

We set this equal to zero and solve for $\hat{\mu}$ (the maximum likelihood estimate of μ):

$$\frac{1}{\sigma^2} \sum_{i=1}^{n} \left(x^{(i)} - \mu \right) = 0 \implies \boxed{\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}}$$

To find the MLE for σ, we then take the derivative of the log-likelihood with respect to σ:

$$\frac{\partial}{\partial \sigma} \log \mathcal{L}(\mu, \sigma) = -\frac{n}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^{n} \left(x^{(i)} - \mu \right)^2$$

We set this equal to zero and solve for $\hat{\sigma}$ (the maximum likelihood estimate of σ)[4]. Note that the answer depends on our previously calculated MLE for μ:

$$-\frac{n}{\hat{\sigma}} + \frac{1}{\hat{\sigma}^3} \sum_{i=1}^{n} \left(x^{(i)} - \mu \right)^2 = 0 \implies \boxed{\hat{\sigma} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left(x^{(i)} - \hat{\mu} \right)^2}}$$

[4]One detail: it turns out this estimate is biased because it depends on the MLE for μ. An unbiased version has $n - 1$ in the denominator instead of n. The effect of this is minimal unless n is small.

Question 5.3

Interpret the MLEs for the parameters, μ and σ, of a normal distribution. Do these results make intuitive sense to you? Think through a few of your examples from Question 4.1.

5.3.4 Poisson Distribution

The Poisson distribution is described in Section 4.5. To find the MLE for λ, its mean, we first (as usual) write down the log-likelihood:

$$\begin{aligned}
\log \mathcal{L}(\lambda) &= \sum_{i=1}^{n} \log p(x^{(i)}|\lambda) \\
&= \sum_{i=1}^{n} \log \left(\frac{e^{-\lambda} \lambda^{x^{(i)}}}{x^{(i)}!} \right) \\
&= \sum_{i=1}^{n} \left[-\lambda + x^{(i)} \log(\lambda) - \log(x^{(i)}!) \right]
\end{aligned}$$

Now we take the derivative of the log-likelihood with respect to λ:

$$\frac{d}{d\lambda} \log \mathcal{L}(\lambda) = \sum_{i=1}^{n} \left[-1 + \frac{x^{(i)}}{\lambda} \right]$$

We set this equal to zero and solve for $\hat{\lambda}$ (the maximum likelihood estimate of λ):

$$\sum_{i=1}^{n} \left[-1 + \frac{x^{(i)}}{\hat{\lambda}} \right] = 0 \implies \boxed{\hat{\lambda} = \frac{1}{n} \sum_{i=1}^{n} x^{(i)}}$$

Question 5.4

Interpret the MLE for the parameter, λ, of a Poisson distribution. Does this result make intuitive sense to you? Think through a few of your examples from Question 4.4.

5.3.5 Geometric Distribution

The geometric distribution is described in Section 4.6. To find the MLE for μ, we first write down the log-likelihood:

$$\begin{aligned}\log \mathcal{L}(\mu) &= \sum_{i=1}^{n} \log p(x^{(i)}|\mu) \\ &= \sum_{i=1}^{n} \log \left((1-\mu)^{x^{(i)}} \mu \right) \\ &= \sum_{i=1}^{n} \left[x^{(i)} \log(1-\mu) + \log(\mu) \right]\end{aligned}$$

Now we take the derivative of the log-likelihood with respect to μ:

$$\frac{d}{d\mu} \log \mathcal{L}(\mu) = \sum_{i=1}^{n} \left[-\frac{x^{(i)}}{1-\mu} + \frac{1}{\mu} \right]$$

We set this equal to zero and solve for $\hat{\mu}$ (the maximum likelihood estimate of μ):

$$\begin{aligned}\sum_{i=1}^{n} \left[-\frac{x^{(i)}}{1-\hat{\mu}} + \frac{1}{\hat{\mu}} \right] = 0 \implies \frac{n}{\hat{\mu}} &= \frac{1}{1-\hat{\mu}} \sum_{i=1}^{n} x^{(i)} \\ \implies \boxed{\hat{\mu} = \frac{n}{\sum_{i=1}^{n} (x^{(i)} + 1)}}\end{aligned}$$

Question 5.5

Interpret the MLE for the parameter, μ, of a geometric distribution. Does this result make intuitive sense to you? Think through a few of your examples from Question 4.5.

5.3.6 Exponential Distribution

The exponential distribution is described in Section 4.7. To find the MLE for λ, we first write down the log-likelihood:

$$\begin{aligned}\log \mathcal{L}(\lambda) &= \sum_{i=1}^{n} \log p(x^{(i)}|\lambda) \\ &= \sum_{i=1}^{n} \log \left(\lambda e^{-\lambda x^{(i)}}\right) \\ &= \sum_{i=1}^{n} \left[\log(\lambda) - \lambda x^{(i)}\right]\end{aligned}$$

Now we take the derivative of the log-likelihood with respect to λ:

$$\frac{d}{d\lambda} \log \mathcal{L}(\lambda) = \sum_{i=1}^{n} \left[\frac{1}{\lambda} - x^{(i)}\right]$$

We set this equal to zero and solve for $\hat{\lambda}$ (the maximum likelihood estimate of λ):

$$\sum_{i=1}^{n} \left[\frac{1}{\hat{\lambda}} - x^{(i)}\right] = 0 \implies \boxed{\hat{\lambda} = \frac{n}{\sum_{i=1}^{n} x^{(i)}}}$$

Question 5.6

Interpret the MLE for the parameter, λ, of an exponential distribution. Does this result make intuitive sense to you? Think through a few of your examples from Question 4.6.

5.4 Summary of MLEs for Common Distributions

The table below contains a summary of the MLEs of various parameters from some common probability distributions.

Distribution	Parameter	ML Estimate	Domain of $x^{(i)}$
Univariate Normal	μ	$\frac{1}{n}\sum_{i=1}^{n} x^{(i)}$	$\mathbb{R}$
	σ	$\frac{1}{n}\sum_{i=1}^{n}\left(x^{(i)}-\hat{\mu}\right)^2$	$\mathbb{R}$
Multivariate Normal	μ	$\frac{1}{n}\sum_{i=1}^{n} x^{(i)}$	$\mathbb{R}^m$
	$\mathbf{\Sigma}$	$\frac{1}{n}\sum_{i=1}^{n}(x^{(i)}-\hat{\mu})(x^{(i)}-\hat{\mu})^T$	$\mathbb{R}^m$
Bernoulli	μ	$\frac{1}{n}\sum_{i=1}^{n} x^{(i)}$	$\{0,1\}$
Binomial (fixed m)	μ	$\frac{1}{nm}\sum_{i=1}^{n} x^{(i)}$	$\{0,1,\ldots,m\}$
Poisson	λ	$\frac{1}{n}\sum_{i=1}^{n} x^{(i)}$	$\{0,1,\ldots\}$
Geometric	μ	$\frac{n}{\sum_{i=1}^{n}(x^{(i)}+1)}$	$\{0,1,\ldots\}$
Exponential	λ	$\frac{n}{\sum_{i=1}^{n} x^{(i)}}$	$\mathbb{R}^+$

Question 5.7

In Question 4.7, we examined several examples of experimental conditions and datasets and discussed which probability distribution best modeled each one. Using the formulas above and the actual datasets from Question 4.7, calculate the MLEs for the parameter(s) of your chosen probability distributions.

Chapter 6

Introduction to Hypothesis Testing

Hypothesis testing is a central idea underpinning much of the analysis in the clinical and biomedical research literature[1]. There are multiple approaches to hypothesis testing, but the most common is **null hypothesis testing**, which was developed by the statistician R.A. Fisher. In null hypothesis testing, one creates a model of how the data should look under default conditions and then quantifies the observed data's deviation from that model using a **test statistic**. If the test statistic is large enough, it means there is evidence that the default position is incorrect.

The statisticians Jerzy Neyman and Karl Pearson developed a different approach to hypothesis testing based on the idea of **model comparison**. In their approach, one sets up different models and then quantifies each model's fit to the data; the hypothesis test is used to see whether one model's fit to the data is significantly better than another's. We see the Neyman-Pearson philosophy reflected in techniques such as power calculations and likelihood ratio tests.

Most of the basic hypothesis tests we learn in introductory biostatistics courses (T-tests, chi-squared tests, etc.) follow Fisher's approach. We will

[1] I should state that there is still a lot of controversy around the whole idea of hypothesis testing and whether p-values should be used at all, etc.

focus on null hypothesis testing in this chapter and explore other ideas in subsequent chapters.

6.1 Basic Steps of a Hypothesis Test

1. *State the **null hypothesis***. The null hypothesis corresponds to the default, or baseline, position; for our example, the null hypothesis might be, "The events 'has mutation' and 'has cancer' are statistically independent." The **alternative hypothesis** is the hypothesis that is contrary to the null; for our example, it might be, "The events 'has mutation' and 'has cancer' are not statistically independent."

2. *List statistical assumptions*. All hypothesis tests make one or more assumptions about the data, and it's important to state them clearly. For example, **parametric** hypothesis tests assume the data follow a particular probability distribution under the null, while **nonparametric** tests do not make this assumption.

3. *Decide on an appropriate test and test statistic*. The **test statistic** quantifies the degree of deviation of the observed data from what one would expect under the null hypothesis[2].

4. *Derive the distribution of the test statistic under the null*. This is called the **null distribution**.

5. *Select a significance level under which you'll reject the null*. The **significance level**, usually written as α, is the probability of a type I error. A type I error is committed when one rejects the null even though it is true (false positive result).

6. *Compute the observed value of the test statistic from the data.*

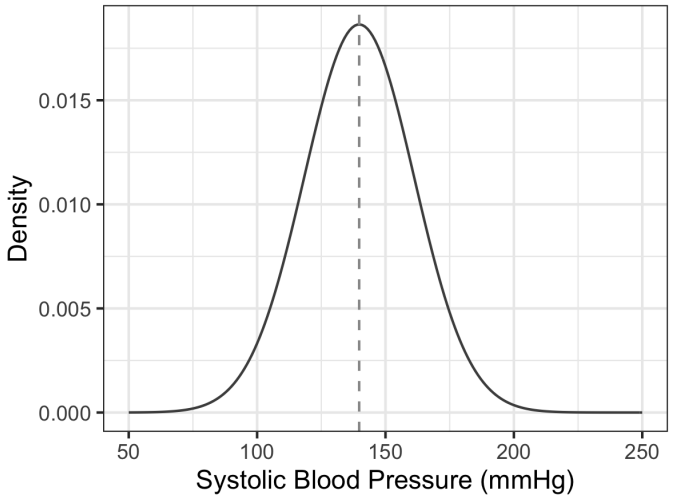
7. *Decide whether or not to reject the null hypothesis.*

[2]Some definitions: A **statistic** is just some quantity that summarizes a set of data, or gives some information about the value of a parameter. A **sufficient statistic** is a statistic that gives the maximum amount of information about a parameter that can possibly be obtained from the sample data.

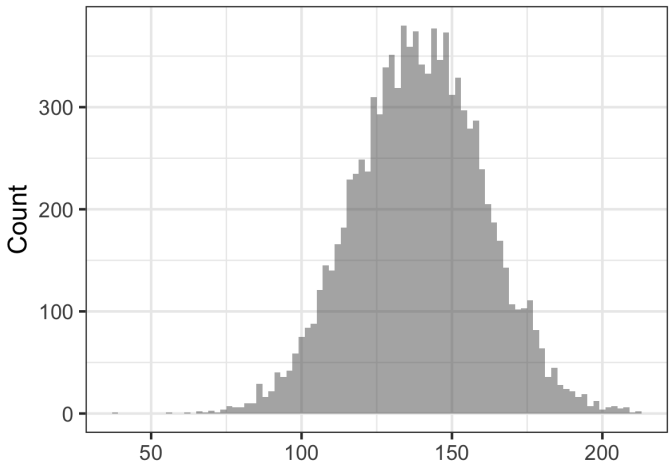
6.2 The Z-Test

A **Z-test** is a hypothesis test for which the null distribution is normal with known mean and standard deviation (i.e. known parameters μ and σ). It is most commonly used to compare the mean of a set of samples, $\overline{x}$, with a known population mean. It also appears in other contexts, such as significance tests of regression coefficients in generalized linear models (Chapter 12).

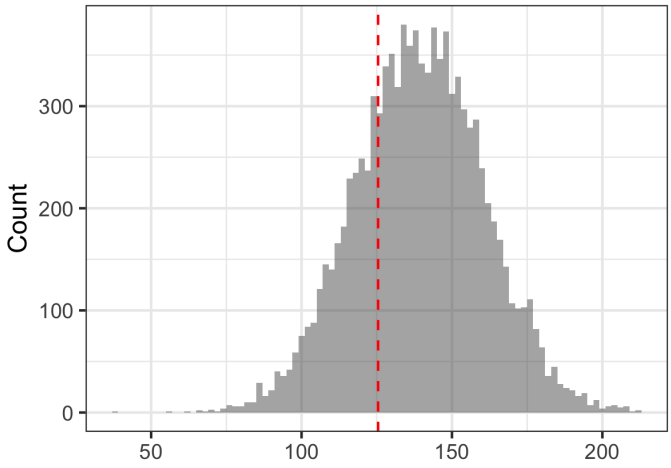
Example: SBP in an Appalachian Town The distribution of systolic blood pressure (SBP) among Caucasian males ages 55-64 in the United States is roughly normal with mean 139.75 mmHg and standard deviation 21.40 mmHg (Source: Int. J. Epidemiol. 2: 294-301, 1973). The following graph shows a normal distribution with those parameters.



Here is a histogram of 10,000 data samples drawn independently from that distribution (i.e., what we would expect if we sampled the SBPs of 10,000 men from the United States at large):



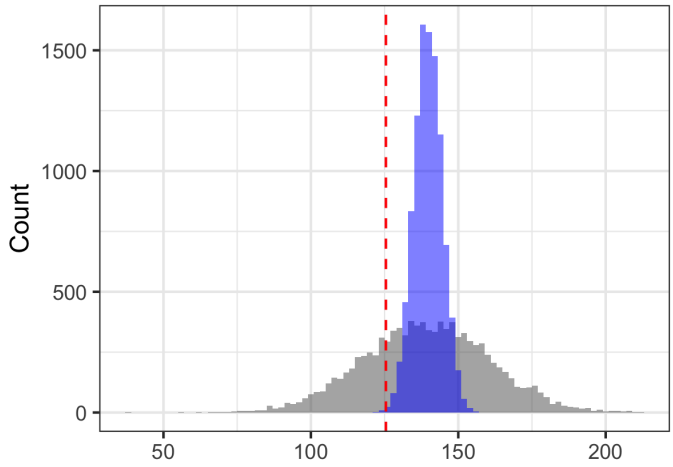
Now, assume some researchers find a small community in rural Appalachia and measure the SBP of 20 Caucasian males ages 55-64 there. Their mean SBP is 125.45 mmHg, illustrated by the red dashed line in the graph below.



At first glance, this may not appear that unusual. After all, the red line is sort of near the center of the gray distribution, right? This analysis is flawed, however, because our 125.45 mmHg value isn't for one man - it's an average

over 20 men. The distribution of the **sample mean**, $\overline{x}$, is different from that of each individual sample.

To see this, imagine taking 20 samples from the gray distribution, taking their mean, and recording that value. Now repeat that process 10,000 times. If you do that, you get the **distribution of the sample mean**, which is skinnier than the gray distribution:



It turns out that the distribution of the sample mean will have the same mean, μ_0, as the population distribution, but its standard deviation will be $\sigma/\sqrt{n}$, where n is the number of samples over which the mean is taken.

Question 6.1

If $n = 1$, what is the standard deviation of the sample mean? If $n = \infty$, what is the standard deviation of the sample mean?

Question 6.2

The sample mean for our 20 sampled Appalachian men is shown as a vertical red dashed line in the figure above. Now that you know what the distribution of the sample mean looks like, do you think the observation from your Appalachian town is "weird"?

Let's conduct a hypothesis test to evaluate whether we have evidence that the mean SBP among men in this town is different from that of the general U.S. population.

1. *State the **null hypothesis***. Here the null hypothesis is going to be our default position: that there is no difference. Let μ_c be the true mean SBP for men in the community and μ_0 be the mean for the general population.

$$H_0 : \mu_c = \mu_0$$
$$H_a : \mu_c \neq \mu_0$$

2. *List statistical assumptions*. We make two assumptions. First, we assume that the SBPs of the different men in the sample are statistically independent. Second, we assume that under the null, SBP will follow a normal distribution with mean 139.75 and standard deviation 21.40, the same as the general population of men aged 55-64.

3. *Decide on an appropriate test and test statistic*. Our test statistic in this case is going to be the **Z-statistic**, which measures the deviation of the sample mean from the population mean in units of the standard deviation of the sample mean, $\sigma/\sqrt{n}$:

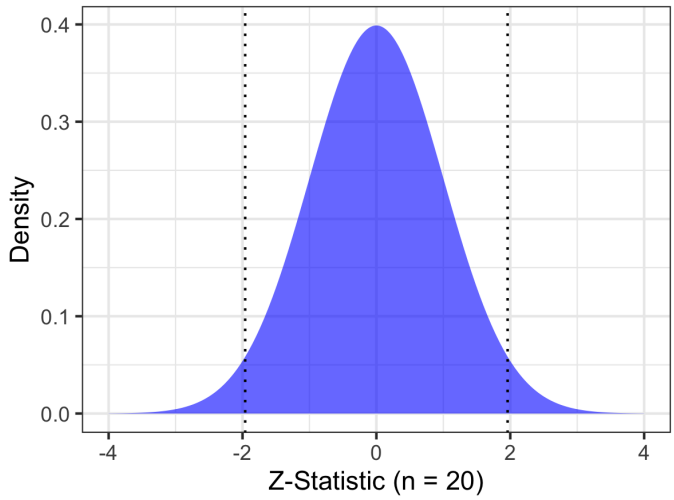
$$Z = \frac{\overline{x} - \mu_0}{\sigma/\sqrt{n}} \qquad \text{where} \qquad \overline{x} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)}$$

   In our case, $n = 20$ because $\overline{x}$, our sample mean, is an average of 20 samples.

4. *Derive the distribution of the test statistic under the null*. The Z-statistic follows a **standard normal** distribution under the null, which is a normal distribution with $\mu = 0$ and $\sigma = 1$. To see this, remember that the distribution of $\overline{x}$ under the null is $\mathcal{N}(\mu_0, \sigma/\sqrt{n})$. When you calculate the Z-statistic, you shift that distribution by a distance μ_0 so it is centered at zero, then adjust its width (standard deviation) to 1.0 by dividing by $\sigma/\sqrt{n}$.

5. *Select a significance level under which you'll reject the null*. For the purposes of this example, we will choose $\alpha = 0.05$ (5% chance of a type I error).

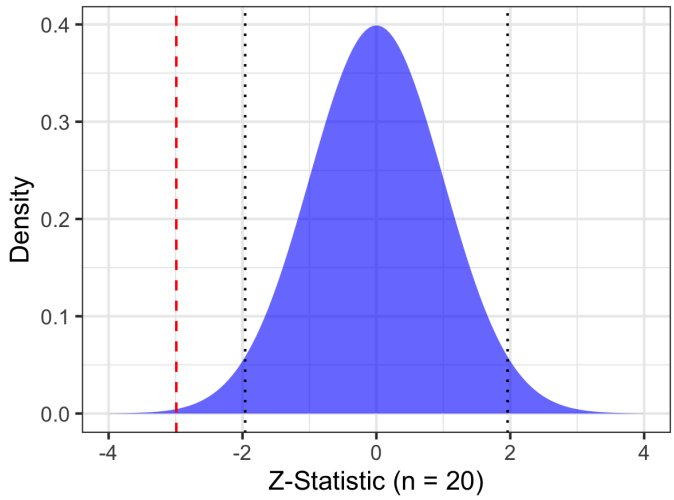
The null distribution of the Z-statistic is shown below. The vertical dotted black lines are situated at the **critical values** that produce $\alpha = 0.05$ (the area under the null distribution that is outside those lines is 0.05).



6. *Compute the observed value of the test statistic from the data.* The observed value of the test statistic is:

$$Z = \frac{\overline{x} - \mu_0}{\sigma/\sqrt{n}} = \frac{125.45 - 139.75}{21.40/\sqrt{20}} = -2.99.$$

7. *Decide whether or not to reject the null hypothesis.* The value of our test statistic falls outside the region contained by the critical values (the **acceptance region**), so we reject the null at this value of α.



Question 6.3

As α gets smaller, are you more or less likely to reject the null for the same value of the test statistic? Hint: What does making α smaller do to the positions of the two black dotted lines in the figure, above?

6.3 Definitions

- **Type I Error:** When a hypothesis test rejects the null even though the null is true (also called a **false positive**). The type I error rate is usually denoted by α.
- **Type II Error:** When a hypothesis test fails to reject the null even though it is false (also called a **false negative**). The type II error rate is usually denoted by β.
- **P-value:** The probability of obtaining a test statistic at least as extreme as the one that was actually obtained, assuming the null is true. A p-value can be **one-sided** or **two-sided**. The difference lies in the definition of "extreme". In a one-sided test, we find the probability that the test statistic is at least as extreme *in the same direction* as the one we observed. In a two-sided test, we find the probability that the test statistic is at

least as extreme *in either direction* (positive or negative deviation). In most cases, this has the practical effect of doubling the p-value.

- **Power:** The probability that a hypothesis test will reject the null when the null is false (that the test will detect a true effect if the effect is there). Usually denoted $1 - \beta$.

6.4 Pearson's Chi-Squared Test

Imagine you have data on two discrete variables for n different subjects. You want to test whether the value of one covariate is independent of the value of the other. To do this, you can arrange your data in a **contingency table** where the rows and columns correspond to the values of the two variables. **Pearson's chi-squared test** can then be used to assess the independence of row and column values.

Example: Association of Genotype and Disease Imagine you want to test whether a person's genotype at a particular locus is associated with whether or not he/she has Disease X. You find 100 people with the disease and 100 healthy controls ($n = 200$) and genotype them:

	AA	Aa	aa	
X	52	43	5	100
Control	67	27	6	100
	119	70	11	200

Let's conduct a hypothesis test to examine this result.

1. *State the **null hypothesis***. We consider the genotype at this locus, G, to be a random variable (see Chapter 4) with three possible outcomes: AA, Aa, and aa. We likewise consider the patient's disease status, D, to be a

random variable with two possible outcomes: disease or no disease. We state our null hypothesis mathematically as:

$$H_0 : G \perp\!\!\!\perp D$$
$$H_a : G \not\perp\!\!\!\perp D$$

where the symbol $\perp\!\!\!\perp$ refers to statistical independence of G and D. We encountered statistical independence in our discussion of maximum likelihood in Chapter 5. Mathematically, statistical independence means that the joint probability of observing a particular value for G and a particular value for D is simply equal to the product of their individual probabilities:

$$P(G = g, D = d) = P(G = g)P(D = d)$$

Under these conditions, the expected values of the cells of our table are:

Under scenario of independence (E):

	AA	Aa	aa	
X	59.5	35.0	5.5	100
Control	59.5	35.0	5.5	100
	119	70	11	200

For example, consider the cell $G = AA, D = X$. Assuming the total number of patients is fixed at $n = 200$ and G and D are independent, the expected number of people in that cell is:

$$P(G = AA, D = X) \cdot n = \left(\frac{119}{200}\right)\left(\frac{100}{200}\right) \cdot 200$$
$$= \mathbf{59.5}$$

Our task now is to decide whether our observed table counts are different enough from what we expect under the null to cause us to reject the null.

2. *List statistical assumptions*. We assume that the data are sampled ran-

domly and independently from a fixed population where each member of the population has an equal probability of selection[3].

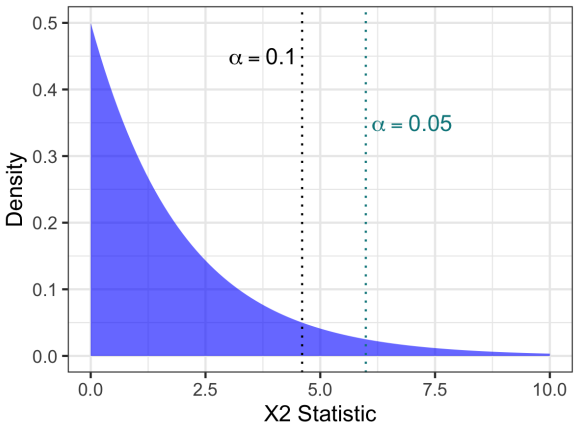
3. *Decide on an appropriate test and test statistic.* The chi-squared test works by calculating expected counts in all $r \times c$ cells of the table (r = number of rows, c = number of columns) and then measuring the data's deviation from those expected counts. The **chi-squared test statistic** has the form

$$X^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

where O refers to "observed count" and E to "expected count". The expected counts are those that assume statistical independence of rows and columns (blue table, above).

4. *Derive the distribution of the test statistic under the null.* Under the null, the X^2 test statistic follows a chi-squared distribution (Section 4.8) with $(r-1)(c-1)$ degrees of freedom. In the case of our genotype example, there are $r = 2$ rows and $c = 3$ columns, thus 2 degrees of freedom.

5. *Select a significance level under which you'll reject the null.* The χ^2 distribution with 2 degrees of freedom is shown below. Two vertical lines are shown at different significance levels: $\alpha = 0.05$ and $\alpha = 0.1$.

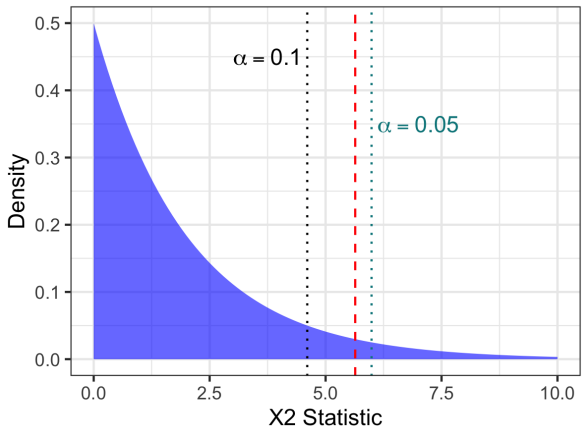


[3]A further assumption of the chi-squared test is that expected counts for each cell must be sufficiently high. A common rule is 5 or more in all cells of a 2×2 table, and 5 or more in 80% of cells in larger tables, but no cells with zero counts.

6. *Compute the observed value of the test statistic from the data.*

Question 6.4

Using the formula in step 4, above, compute the actual value of the chi-squared test statistic for this example. Hint: You should end up with a value that corresponds to the position of the red dashed line in the figure below.



7. *Decide whether or not to reject the null hypothesis.* Based on our calculated value of the test statistic, we will reject the null at $\alpha = 0.1$ and fail to reject the null at $\alpha = 0.05$.

Although it looks much different from the Z-test, the chi-squared test follows the same formalism: defining a null hypothesis, figuring out what the data should look like under the null, quantifying the deviation of the observed data from what's expected using a test statistic, and deciding if that test statistic presents strong enough evidence to cause us to reject the null.

6.5 Student's T-tests

The final example we will look at today is the **T-test**. Like the Z-test, the T-test (actually a family of tests) deals with situations where you have data that are assumed to be normally distributed under the null hypothesis. However, in

this scenario, the population standard deviation, σ is not known and must be estimated from the data itself.

6.5.1 One Sample T-test

Assume you have a dataset $x^{(1)}, \ldots, x^{(n)}$, of real numbers that you can plausibly assume are normally distributed. You want to test whether the mean of your data is equal to a fixed value, μ_0. Under the null hypothesis that the means are the same, the test statistic

$$T = \frac{\overline{x} - \mu_0}{s/\sqrt{n}}$$

which we call a "T statistic", follows a T-distribution (Section 4.9) with $n-1$ degrees of freedom[4]. Here $\overline{x}$ refers to the sample mean, and s refers to the **sample standard deviation**:

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x^{(i)} - \overline{x})^2}$$

Question 6.5

Compare the formula for the sample standard deviation to the maximum likelihood estimate of the parameter, σ, of a normal distribution (Section 5.3.3). What is the same/different? Note in particular the use of $n-1$ in the denominator, rather than n. This arises because the MLE for σ, $\hat{\sigma}$, is a **biased** estimate of the population standard deviation (more on this later). For large n, however, the two are nearly identical.

[4] A one-sample T-test looks a lot like a Z-test. However, because we use s to estimate the population standard deviation from data, we must account for variation in our estimate. It turns out that the sample variance, s^2, follows a chi-squared distribution with $n-1$ degrees of freedom, where n is the sample size. In this case, by the definition of the T-distribution (Section 4.9), the T statistic follows a Student's T-distribution with $n-1$ degrees of freedom. As the number of samples, n, grows, the sample standard deviation approaches the population standard deviation and the T-test becomes a Z-test. But when n is small, the T-test is quite a bit more conservative.

6.5.2 Two Independent Samples, Equal Variance

Assume you have a dataset $x^{(1)}, \ldots, x^{(n)}$ and another dataset $y^{(1)}, \ldots, y^{(m)}$. You assume that both are drawn from normal distributions with equal variance but potentially different means. You want to test whether the means are equal.

The same basic machinery for the one-sample T-test can be deployed in this context with a slightly different test statistic. The test statistic

$$T = \frac{\overline{x} - \overline{y}}{s_p \sqrt{\frac{1}{n} + \frac{1}{m}}}$$

where

$$\begin{aligned} s_p^2 &= \frac{(n-1)s_x^2 + (m-1)s_y^2}{m+n-2} \\ s_x^2 &= \frac{1}{n-1} \sum_{i=1}^{n} (x^{(i)} - \overline{x})^2 \\ s_y^2 &= \frac{1}{m-1} \sum_{i=1}^{m} (y^{(i)} - \overline{y})^2 \end{aligned}$$

follows a t-distribution with $m + n - 2$ degrees of freedom.

6.5.3 Two Independent Samples, Unequal Variance

Sometimes you have two independent samples but cannot assume the variances are equal. Again, similar machinery can be deployed. In this case, you can use **Welch's T-test**, which uses the test statistic

$$T = \frac{\overline{x} - \overline{y}}{s_{xy}}$$

where

$$s_{xy} = \sqrt{\frac{s_x^2}{n} + \frac{s_y^2}{m}}.$$

This test statistic approximately follows a t-distribution with degrees of freedom given by the Welch-Sattherwaite Equation

$$\text{d.f.} = \frac{\left(\frac{s_x^2}{n} + \frac{s_y^2}{m}\right)^2}{\frac{(s_x^2/n)^2}{n-1} + \frac{(s_y^2/m)^2}{m-1}}$$

6.5.4 Matched Pairs

Assume you have a data set of matched pairs. This could be a set of measurements of the same individuals taken at two different points in time, for example, or paired measurements taken from individuals with similar characteristics. You want to test whether the second set of values have changed relative to the first set of values.

To do this, you can use a one-sample T-test on the *differences* of the individual pairs. If no change has occurred, you would expect the mean of those differences to be zero. If we define $x^{(i)}$ as the difference of the paired observations for sample i and $\overline{x}$ as $\frac{1}{n}\sum_{i=1}^{n} x^{(i)}$, the sample mean of those differences, then

$$T = \frac{\overline{x}}{s/\sqrt{n}}$$

follows a T-distribution with $n-1$ degrees of freedom.

Question 6.6

Here are some sample data. They come from a study that looked at the effect of ozone, a component of smog, on the weight gain of rats. (Original source: Biometrika 63: 421-434, 1976, reproduced in Rice's *Mathematical Statistics and Data Analysis*, p. 465.) A group of 22 seventy-day-old rats were kept in an environment containing ozone for 7 days, and their weight gains were recorded. Another group of 23 rats of a similar age were kept in an ozone-free environment for a similar time and their weight gains were also recorded. Here are the data for the control group:

	group	original_weight	weight_gain
1	control	340.8	41.0
2	control	389.1	25.9
3	control	355.2	13.1
4	control	421.8	-16.9
5	control	377.1	15.4
6	control	404.3	22.4
7	control	321.2	29.4
8	control	447.5	26.0
9	control	305.9	38.4
10	control	335.9	21.9
11	control	386.3	27.3
12	control	377.0	17.4
13	control	357.2	27.4
14	control	441.7	17.7
15	control	383.7	21.4
16	control	373.7	26.6
17	control	336.0	24.9
18	control	419.4	18.3
19	control	287.1	28.5
20	control	602.8	21.8
21	control	325.4	19.2
22	control	452.4	26.0
23	control	398.9	22.7
Mean	control	384.4	22.4
St.Dev.	control	65.5	10.8

And here are the data for the ozone group:

	group	original_weight	weight_gain
1	ozone	437.4	10.1
2	ozone	275.9	7.3
3	ozone	296.3	-9.9
4	ozone	295.9	17.9
5	ozone	379.7	6.6
6	ozone	274.1	39.9
7	ozone	360.0	-14.7
8	ozone	331.9	-9.0
9	ozone	531.8	6.1
10	ozone	350.5	14.3
11	ozone	345.7	6.8
12	ozone	268.1	-12.9
13	ozone	339.9	12.1
14	ozone	352.4	-15.9
15	ozone	435.8	44.1
16	ozone	476.9	20.4
17	ozone	462.5	15.5
18	ozone	368.0	28.2
19	ozone	504.3	14.0
20	ozone	188.0	15.7
21	ozone	466.9	54.6
22	ozone	288.8	-9.0
Mean	ozone	365.0	11.0
St.Dev.	ozone	88.6	19.0

(a) Imagine that the population weight distribution of rats is known to be normal with $\mu = 350$ (grams) and unknown σ. How would you test the hypothesis that the mean of the control group is equal to the population mean? How would you test the hypothesis that the mean of the ozone group is equal to the population mean?

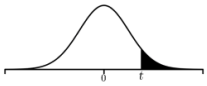
(b) How would you test the hypothesis that the mean original weights of the ozone and control groups are equal? Do not assume equal variance.

(c) How would you test the hypothesis that the mean weight gain in the ozone group is equal to the mean weight gain in the control group? Do not assume equal variance.

(d) How would your approach in part (c) change if you assumed the weight gains in the two groups had equal variance?

Plug in the relevant numbers from the tables above to perform each hypothesis test with $\alpha = 0.05$. The following table of critical values for the T-distribution[a] may help you:

Critical Values for Student's t-Distribution.



	Upper Tail Probability: $\Pr(T > t)$									
df	0.2	0.1	0.05	0.04	0.03	0.025	0.02	0.01	0.005	0.0005
1	1.376	3.078	6.314	7.916	10.579	12.706	15.895	31.821	63.657	636.619
2	1.061	1.886	2.920	3.320	3.896	4.303	4.849	6.965	9.925	31.599
3	0.978	1.638	2.353	2.605	2.951	3.182	3.482	4.541	5.841	12.924
4	0.941	1.533	2.132	2.333	2.601	2.776	2.999	3.747	4.604	8.610
5	0.920	1.476	2.015	2.191	2.422	2.571	2.757	3.365	4.032	6.869
6	0.906	1.440	1.943	2.104	2.313	2.447	2.612	3.143	3.707	5.959
7	0.896	1.415	1.895	2.046	2.241	2.365	2.517	2.998	3.499	5.408
8	0.889	1.397	1.860	2.004	2.189	2.306	2.449	2.896	3.355	5.041
9	0.883	1.383	1.833	1.973	2.150	2.262	2.398	2.821	3.250	4.781
10	0.879	1.372	1.812	1.948	2.120	2.228	2.359	2.764	3.169	4.587
11	0.876	1.363	1.796	1.928	2.096	2.201	2.328	2.718	3.106	4.437
12	0.873	1.356	1.782	1.912	2.076	2.179	2.303	2.681	3.055	4.318
13	0.870	1.350	1.771	1.899	2.060	2.160	2.282	2.650	3.012	4.221
14	0.868	1.345	1.761	1.887	2.046	2.145	2.264	2.624	2.977	4.140
15	0.866	1.341	1.753	1.878	2.034	2.131	2.249	2.602	2.947	4.073
16	0.865	1.337	1.746	1.869	2.024	2.120	2.235	2.583	2.921	4.015
17	0.863	1.333	1.740	1.862	2.015	2.110	2.224	2.567	2.898	3.965
18	0.862	1.330	1.734	1.855	2.007	2.101	2.214	2.552	2.878	3.922
19	0.861	1.328	1.729	1.850	2.000	2.093	2.205	2.539	2.861	3.883
20	0.860	1.325	1.725	1.844	1.994	2.086	2.197	2.528	2.845	3.850
21	0.859	1.323	1.721	1.840	1.988	2.080	2.189	2.518	2.831	3.819
22	0.858	1.321	1.717	1.835	1.983	2.074	2.183	2.508	2.819	3.792
23	0.858	1.319	1.714	1.832	1.978	2.069	2.177	2.500	2.807	3.768

Answers: (a) One-sample T-test of control group original weights vs. null of $\mu_0 = 350$; T-statistic is 2.5165, 22 d.f., two-sided p-value is 0.01964, reject null at $\alpha = 0.05$. One-sample T-test of ozone group original weights vs. null of $\mu_0 = 350$; T-statistic is 0.7961, 21 d.f., two-sided p-value is 0.4349, fail to reject null at $\alpha = 0.05$. (b) Welch's two-sample T-test of control vs. ozone group original weights; T-statistic is 0.8293, d.f is estimated using the Welch-Satterwaite equation at 38.619, two-sided p-value is 0.4120, fail to reject null at $\alpha = 0.05$. (c) Welch's two-sample T-test of control vs. ozone group weight gains; T-statistic is 2.4629, d.f. is estimated using the Welch-Satterwaite equation at 32.918, two-sided p-value is 0.01918, reject null at $\alpha = 0.05$. (d) You would use Pearson's two-sample T-test, which assumes equal variances; T-statistic is 2.4919, d.f. is 43, two-sided p-value is 0.01664, reject null at $\alpha = 0.05$.

[a] Borrowed with gratitude from https://www.stat.purdue.edu/ lfindsen/stat503/t-Dist.pdf

Chapter 7

Building a Decision Tree

Decision trees were developed as an alternative to neural networks in the 1970s. We already saw them in Chapters 2 and 3 as examples of supervised learning algorithms. Now we're going to get into a bit more detail about how these trees are learned from data.

Decision trees can be adapted to solve supervised learning problems with different types of outcomes. We will focus on **classification trees** in this chapter, but the same tree building principles can be applied to solve regression problems (see Chapter 3). Similar modifications can also allow us to construct **survival trees**, which model survival outcomes (represented as Kaplan-Meier curves; see Chapter 11). Decision trees can also handle count outcomes, modeling them using Poisson distributions (see Chapter 4).

7.1 Example: The Wisconsin Breast Cancer Dataset

The Wisconsin breast cancer dataset[1] contains information about imaging features of a fine needle aspirate (FNA) of a breast mass from 569 different subjects. Ten real-valued features are computed for each cell nucleus in the image:

(a) radius (mean of distances from center to points on the perimeter)

[1]You can download the dataset from the UCI Machine Learning Repository here: `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`.

(b) texture (standard deviation of gray-scale values)

(c) perimeter

(d) area

(e) smoothness (local variation in radius lengths)

(f) compactness (perimeter^2 / area - 1.0)

(g) concavity (severity of concave portions of the contour)

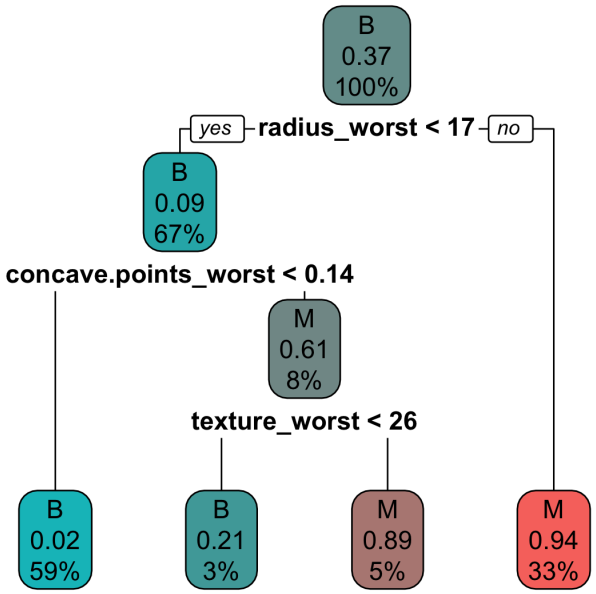
(h) concave points (number of concave portions of the contour)

(i) symmetry

(j) fractal dimension, a statistical index of complexity quantifying how detail in a pattern changes with the scale on which it is measured

The mean, standard deviation, and worst value of each feature are then computed, creating a total of 30 features. Each image is also labeled by its true diagnosis: *B* (for benign) or *M* (for malignant).

Here is a decision tree for this dataset, built using the `rpart` package in R with default parameters:



Question 7.1

What is the most important feature, as identified by the decision tree learning algorithm, for determining whether a breast mass is benign or malignant? What two other features are considered important by the tree? Which features are ignored completely?

Question 7.2

Looking at the decision tree for the Wisconsin Breast Cancer dataset, what do you think the advantages of a decision tree are for this problem over other classification methods, such as logistic regression and KNN?

Although there are several tree building algorithms, all of them are conceptually similar. They all try to reduce "impurity", or "uncertainty", in the outcome variable by intelligently splitting on the predictors. Trees are built recursively from root to leaves. Each internal node of the tree "contains" a subset of the overall dataset (this is why tree building is often called **recursive partitioning**, and why the R package is named `rpart`). At each stage, the algorithm will consider each of the existing leaf nodes and choose the split variable that maximally reduces uncertainty in the outcome. To understand how trees are built computationally, all we need to do is look at the math behind this idea.

7.2 The ID3 Algorithm

One of the earliest approaches to building decision trees was the **ID3 algorithm**[2]. The ID3 Algorithm uses the concepts of entropy and information gain to build trees.

7.2.1 Entropy

Entropy, usually abbreviated H, is a measure of the uncertainty in the value of a random variable. It is the number of bits (on average) required to describe

[2]Quinlan, J.R. "Induction of decision trees", Machine Learning, num. 1, pp. 81- 106, 1986.

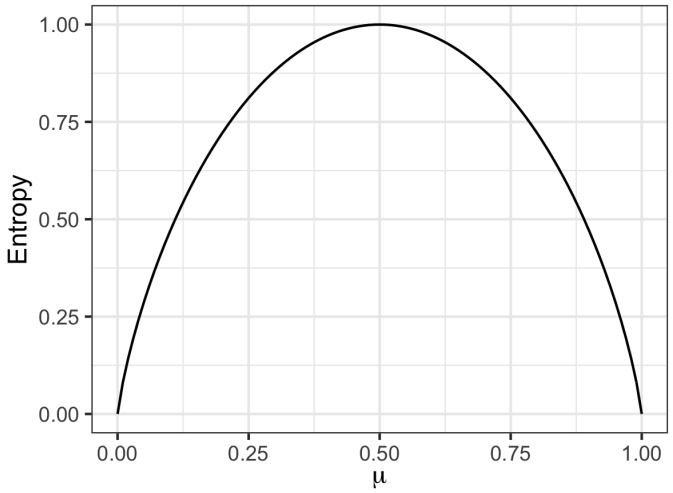
the outcome of the random variable. Here is the formula for the entropy of the discrete probability distribution governing the outcome of a random variable, X:

$$H(X) = -\sum_{x} P(X = x) \log_2 \left(P(X = x)\right)$$

For a Bernoulli random variable (see Chapter 4, Section 4.3), there are only two possible outcomes: 0 and 1. The entropy of this random variable is given by:

$$H_{\text{Bernoulli}} = -\mu \log_2(\mu) - (1 - \mu) \log_2(1 - \mu)$$

where μ is the sole parameter of the Bernoulli distribution: the probability of a positive outcome. Here is what the entropy of a Bernoulli distribution looks like as a function of μ:



Question 7.3

At which value(s) of μ are we maximally uncertain about the outcome? At which value(s) of μ are we completely certain about the outcome? This should make intuitive sense if you consider the meaning of μ.

7.2.2 Information Gain

The goal of a decision tree is to reduce uncertainty about the outcome with every split. Let Y be the outcome variable of a training set. Let X be one of

the predictors. **Information gain** is defined as:

$$\begin{aligned}\text{Gain}(Y, X) &= H(Y) - \sum_{x \in \text{Values(X)}} \frac{|Y(X = x)|}{|Y|} H(Y(X = x)) \\ &= H(Y) - H(Y|X)\end{aligned}$$

Since entropy is a measure of uncertainty, or impurity, in the outcome, information gain is the reduction in that uncertainty achieved by conditioning on the predictor, X. The tree will choose split variables for which the information gain is maximized.

Question 7.4

Say you have a dataset where the outcome is $Y = [0, 1, 0, 1, 0, 1]$ and there are two predictors: $X = [0, 0, 1, 1, 2, 2]$, and $Z = [1, 2, 1, 2, 1, 2]$. Intuitively, which predictor would make the better splitting variable and why? Calculate $Gain(Y, X)$ and $Gain(Y, Z)$. Which value is higher?

7.2.3 Using Entropy to Build a Decision Tree

By understanding the concepts of entropy and information gain, we arrive naturally at the ID3 Algorithm:

1. Start with a single node: the root of the tree.
2. At each current leaf node:
   (a) Compute the information gain for each feature.
   (b) Split on the one with the highest gain.
3. Return to Step 2. Stop when either the class distributions at all leaf nodes are entirely pure or there are no more variables left to split on.

7.3 Example: Happiness Dataset

Let's build a decision tree for a simple example, using the ID3 algorithm. Assume we surveyed 10 people and asked them whether they were happy or

unhappy. We also asked whether they had friends (yes/no), money (poor/enough/rich), and free time (none/some). The data look like this:

Datapoint ID	friends (X_1)	money (X_2)	free time (X_3)	happy (Y)
1	1	1	0	0
2	1	1	1	0
3	0	1	1	0
4	0	0	0	0
5	1	0	0	0
6	0	0	0	0
7	1	2	1	1
8	1	0	1	1
9	0	0	1	1
10	1	0	0	1

$$x_1 = \begin{cases} 0 & \text{no friends} \\ 1 & \text{friends} \end{cases} \qquad x_2 = \begin{cases} 0 & \text{poor} \\ 1 & \text{enough money} \\ 2 & \text{rich} \end{cases} \qquad x_3 = \begin{cases} 0 & \text{no free time} \\ 1 & \text{some free time} \end{cases}$$

Question 7.5

Build a decision tree for this dataset using the ID3 algorithm. To get started, you need to know the entropy of the overall outcome distribution. It is:

$$H(Y) = -\frac{4}{10}\log_2\frac{4}{10} - \frac{6}{10}\log_2\frac{6}{10} = \mathbf{0.971}$$

We will go through the calculations below. As we go, you can start to draw the tree on another page.

(a) Perform the initial split at the tree root to determine which variable to split on first. Update the tree with this information.

$$\frac{|Y(X_1 = 0)|}{|Y|} H[Y(X_1 = 0)] =$$

$$\frac{|Y(X_1 = 1)|}{|Y|} H[Y(X_1 = 1)] =$$

$$\text{Gain}(Y, X_1) =$$

$$\frac{|Y(X_2 = 0)|}{|Y|} H[Y(X_2 = 0)] =$$

$$\frac{|Y(X_2 = 1)|}{|Y|} H[Y(X_2 = 1)] =$$

$$\frac{|Y(X_2 = 2)|}{|Y|} H[Y(X_2 = 2)] =$$

$$\text{Gain}(Y, X_2) =$$

$$\frac{|Y(X_3 = 0)|}{|Y|} H[Y(X_3 = 0)] =$$

$$\frac{|Y(X_3 = 1)|}{|Y|} H[Y(X_3 = 1)] =$$

$$\text{Gain}(Y, X_3) =$$

(b) We see that two of the leaves of our tree are "pure", meaning that all of the training examples that arrive there are of one outcome class. For those two leaves, we're done. However, for the third ($X_2 = 0$, or poor), we need to perform another split. Perform the split at the $X_2 = 0$ node to find which variable to split on next and update the tree with this information.

$$H[Y(X_2 = 0)] =$$

$$\frac{|Y(X_2 = 0, X_1 = 0)|}{|Y(X_2 = 0)|} H[Y(X_2 = 0, X_1 = 0)] =$$

$$\frac{|Y(X_2 = 0, X_1 = 1)|}{|Y(X_2 = 0)|} H[Y(X_2 = 0, X_1 = 1)] =$$

$$\text{Gain}(Y(X_2 = 0), X_1) =$$

$$\frac{|Y(X_2 = 0, X_3 = 0)|}{|Y(X_2 = 0)|} H[Y(X_2 = 0, X_3 = 0)] =$$

$$\frac{|Y(X_2 = 0, X_3 = 1)|}{|Y(X_2 = 0)|} H[Y(X_2 = 0, X_3 = 1)] =$$

$$\text{Gain}(Y(X_2 = 0), X_3) =$$

(c) We need to do one more split on the $X_2 = 0, X_3 = 0$ node. The only variable left to split on is X_1 (friends). Perform this split and add this information to the tree.

7.4 Alternative Splitting Criteria

Information gain is not the only criterion that is used in decision tree algorithms. In fact, the tree built for the Wisconsin Breast Cancer dataset in Section 7.1 used a different criterion, the mean decrease in Gini impurity, or **Gini index**, because it is the default in R's `rpart` package.

The Gini impurity measures how often a randomly chosen element from a set would be incorrectly labeled if it were randomly labeled using the distribution of labels in the subset. It sums up the probability of an item with label i being chosen (p_i) multiplied by the probability $\sum_{j \neq i} p_j = 1 - p_i$ that a mistake is made in classifying it. The formula is:

$$G(p) = \sum_i p_i(1 - p_i) = 1 - \sum_i p_i^2$$

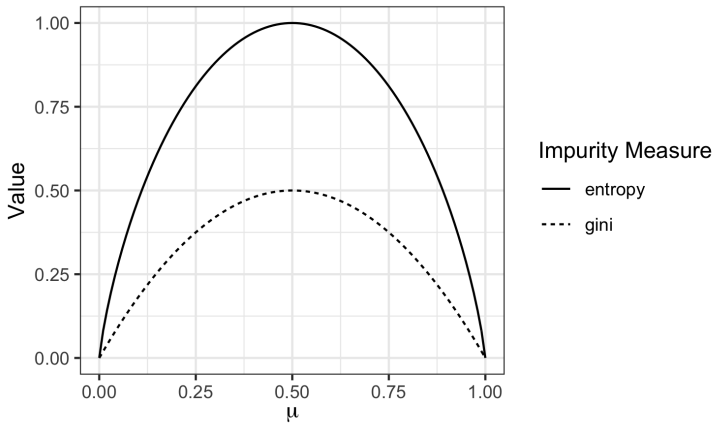
which for a Bernoulli distribution is

$$G_{\text{Bernoulli}} = 1 - \mu^2 - (1 - \mu)^2.$$

The Gini impurity plays a role analogous to entropy. The Gini index is a weighted sum of the Gini impurities within the children of a particular split; if it is low, it means the split was successful in reducing impurity.

Question 7.6

Plots of the Gini impurity vs. entropy for a Bernoulli distribution are shown below. What do you notice about the value(s) of μ for which each is maximized or minimized?



Entropy and Gini impurity are similar and usually, though not always, yield similar trees. The Gini impurity is computationally faster because it does not make use of logarithms as the entropy does. This can make a difference as the number of features grows.

7.5 Decision Tree Regression

So far we've assumed that our outcome is discrete. But what happens if it's numeric? (That is, what if we want to perform regression instead of classification?)

In that case, we use **standard deviation reduction** instead of information gain to decide which variables to split on. The sample standard deviation of an outcome, y, is defined as:

$$S(Y) = \sqrt{\frac{\sum_i (y^{(i)} - \overline{y})}{n-1}}$$

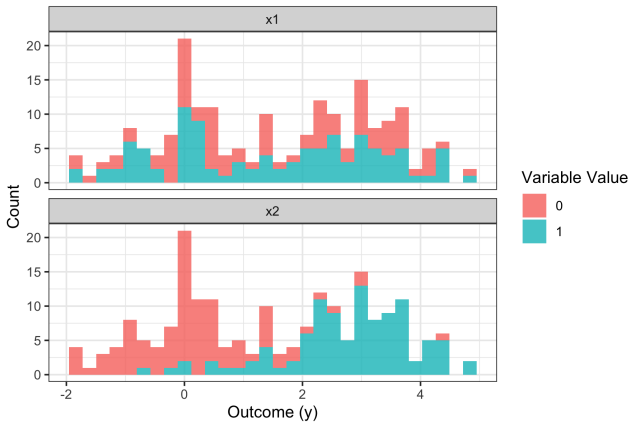
The procedure is identical to the ID3 algorithm except that it uses conditional standard deviation instead of information gain to decide on features. We define

$$S(Y, X) = \sum_x P(X = x)\, S(Y(X = x))$$

and at each current leaf node, we split on the variable where the reduction in standard deviation, $S(Y) - S(Y, X)$, is the highest.

Question 7.7

Imagine you have a dataset with two predictors, x_1 and x_2, each of which is binary (can only be 0 or 1). Here are the distributions of outcome values associated with x_1 and x_2:



Based on the idea of standard deviation reduction, which of these two variables, x_1 or x_2, would make the most sense for a decision tree to split on? What would such a split look like and what would the output value of the tree (the predicted value of y) be for each side of the split?

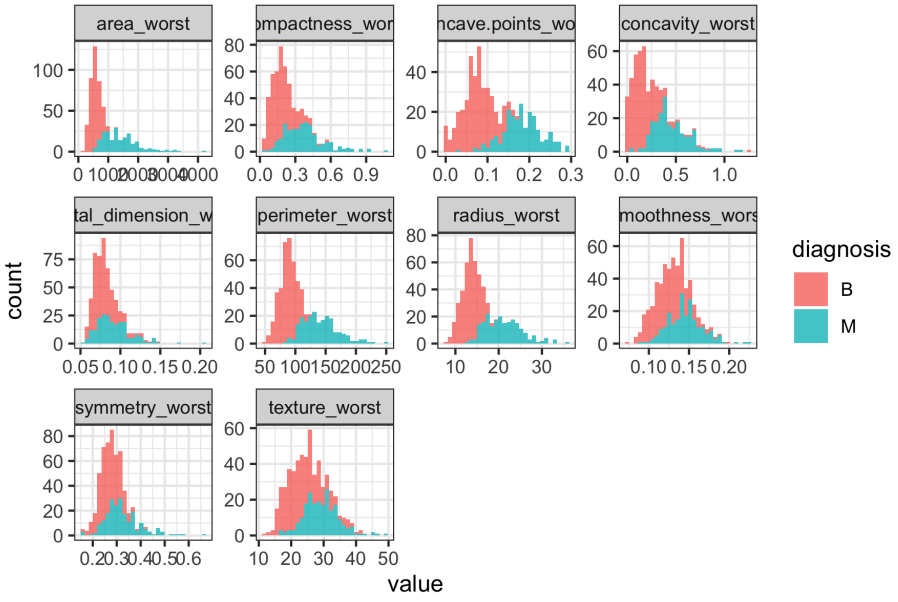
7.6 Numeric Predictors

Although the Happiness Dataset contained only discrete predictors, decision trees can also handle numeric predictors. We've seen this already with the Wisconsin Breast Cancer dataset.

There are different strategies for deciding on an optimal split for a predictor. The most common approach is to consider all possible splits. So for example, if a predictor has values $10, 11, 16, 18, 20$, and 35, the tree building algorithm would consider all $N - 1 = 5$ possible split points. If a dataset has a large number of numeric features or features with lots of different possible values, therefore, it can really slow down the construction of the tree.

Question 7.8

Here are histograms of 10 of the predictors in the Wisconsin Breast Cancer dataset. Only the "worst" variable version of each predictor is shown for clarity. Which variable, and which threshold, appears to show the clearest division of samples into B and M groups? Compare your choice to the first split of the tree in Section 7.1.



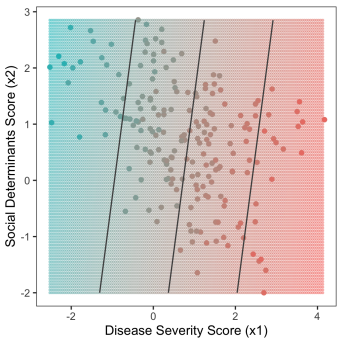
Chapter 8

Interpreting a Linear Regression Model

This chapter is devoted to understanding the structure of linear regression models. We first encountered them in Chapter 3 as "just one example" of a regression model. However, linear regression's overwhelming popularity in the clinical domain means that one cannot do clinical data science without fully understanding these models' structure and how to interpret the output produced by model fitting software.

8.1 Biomarker Example from Chapter 3

In Chapter 3, we saw an example where information about two predictors – a disease severity score (x_1) and a social determinants score (x_2) – was used to predict the level of a disease recurrence biomarker. One of the three supervised learning algorithms we tried was a **linear regression** model (Section 3.2.1). The output from that model is repeated below.




```
Call:
lm(formula = y ~ x1 + x2, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-11.9218  -3.1032   0.2891   2.8316  12.5813

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  49.8600     0.5370  92.844  < 2e-16 ***
x1           10.4372     0.2855  36.555  < 2e-16 ***
x2           -1.8824     0.3609  -5.215 4.63e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.769 on 197 degrees of freedom
Multiple R-squared:  0.9026,    Adjusted R-squared:  0.9016
F-statistic: 912.4 on 2 and 197 DF,  p-value: < 2.2e-16
```

Question 8.1

What are the number of samples, n, and the number of predictors, p, for this dataset?

But what do all these numbers *mean*?

8.2 Understanding the Model Summary

A linear regression model looks like this (see also Chapter 3, Section 3.2.1):

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon$$

where we assume that the error, ε, is normally distributed, $\mathcal{N}(0, \sigma)$. Another way of saying this is that we are assuming the outcome, y, is normally distributed with mean $\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$ and standard deviation σ.

8.2.1 The Call

The first line of the output is just repeating the call you made to the `lm` function in R to fit the model. The `lm` package fits linear regression models using ordinary least squares. However, linear regression models are also a type of generalized linear model (see Chapter 12) and can be fit using maximum likelihood. In R, if you use the `glm` package with `family = "gaussian"`

you should get identical coefficients and error estimates to what you get using the `lm` package.

8.2.2 The Residuals

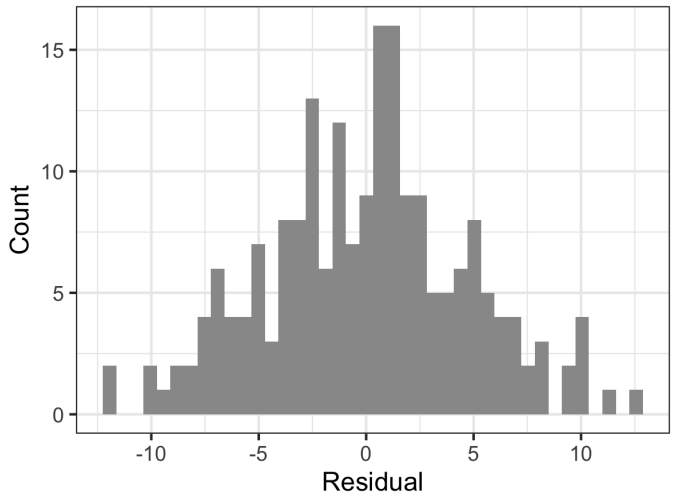
A **residual** is a measure of how much the true outcome value (y) of one datapoint differs from the model's prediction. For a linear regression model, the residual of training point i is:

$$\text{residual}^{(i)} = y^{(i)} - \hat{y}^{(i)}$$

where $\hat{y}^{(i)}$ is the model's prediction:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \cdots + \beta_p x_p^{(i)}.$$

Here is a histogram of the residuals for this model:

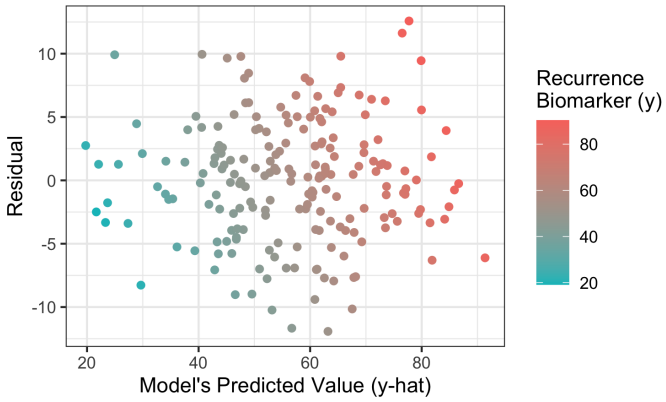


Question 8.2

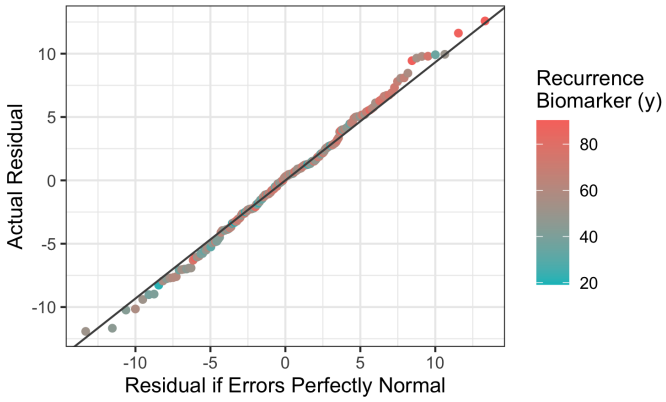
Estimate the maximum, minimum, and mean residuals from this graph. Do they match what is in the model output?

The residuals play an important role in linear regression models because they are what allow us to estimate σ. They also play an important role in

model diagnostics because they enable us to check one of the core model assumptions: the assumption that σ is constant. We can check this assumption by making a plot of the residuals vs. $\hat{y}$:



We can also check the normality of the residuals by making a plot of their values vs. what we would expect if the residuals were perfectly normally distributed with the same mean and standard deviation:



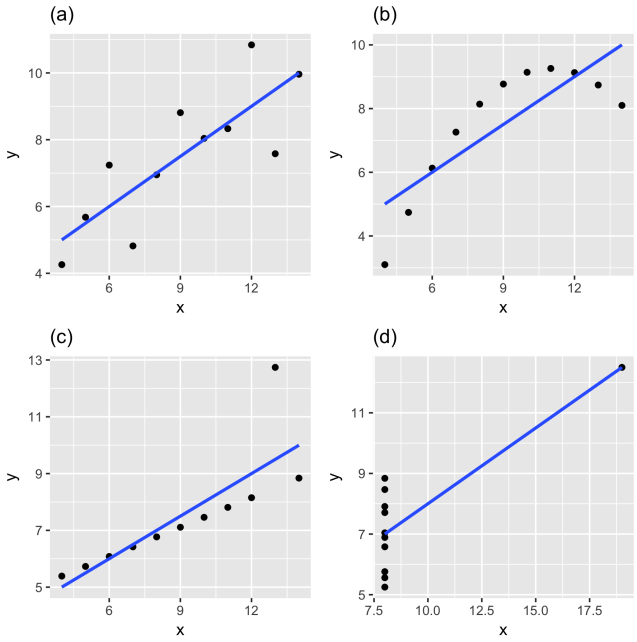
This is not quite a **QQ-plot**, one of the standard diagnostic plots of linear regression, because it's plotting the actual values of the residuals instead of their quantiles. But aside from the axis scales, it's exactly the same. We'll see formal QQ-plots later.

The fact that the points in the second plot lie close to a line is a good indication that the residuals are normally distributed. Another thing to notice

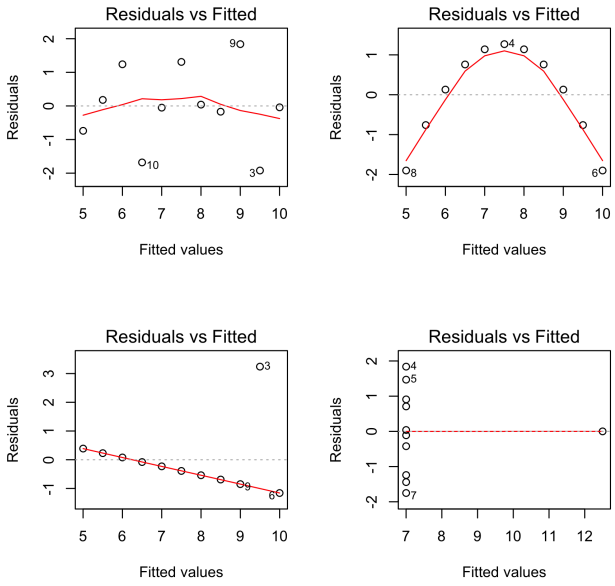
from this plot is that the colors of the points (their y values) are unrelated to their residuals.

Question 8.3

The four plots below show a famous dataset called **Anscombe's quartet**. The regression lines produced by fitting a linear regression model to each dataset are identical, but only one dataset actually fulfills the assumptions of a linear regression model.



We can check these assumptions by examining plots of the residuals vs. fitted values of the model (here the "fitted value" of point i means $\hat{y}^{(i)}$).



Which of the four datasets fulfills the assumption of a linear regression model that the error has constant variance? How can you tell?

8.2.3 Coefficients and Standard Errors

Although residuals are important, the parts of the model output that will be scrutinized, reported in papers, etc. are the coefficients, along with their standard errors and hypothesis tests. The coefficients are what create the regression surface, which captures the model's prediction for every point in the feature space (see Section 3.2.1).

There are actually a few different ways to derive the coefficients in a linear regression model. The most common way uses **least squares**, which adjusts the coefficients $\beta_0, \beta_1, \ldots, \beta_p$ until the sum of the squared residuals is

minimized:

$$\begin{aligned}\text{SSE} &= \sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2 \\ &= \sum_{i=1}^{n}(y^{(i)} - \beta_0 - \beta_1 x_1^{(i)} - \beta_2 x_2^{(i)} - \cdots - \beta_p x_p^{(i)})^2\end{aligned}$$

It turns out that you can find the optimal values of the βs analytically by taking derivatives of this thing and setting them equal to zero. Alas, this requires some matrix multiplication and the taking of matrix inverses, so we will save it for a later chapter. Suffice it to say that the βs are adjusted to minimize the SSE, and the values in the model output are the optimal values.

Question 8.4

Looking at the form of the linear regression model

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \varepsilon$$

what does the value of each of the βs mean? What is β_j telling us about how y varies with the predictor j, all else being equal?

The model's overall estimate of σ, the standard deviation of the error term, is obtained very naturally by averaging over the residuals, taking into account the number of predictors, p:

$$\hat{\sigma}^2 = \frac{1}{n-p-1}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2.$$

Question 8.5

The sum of the squared residuals for our model is 4480.678. There are $n = 200$ datapoints, and the number of predictors, p, is 2. Calculate $\hat{\sigma}$ for this model. Do you see this number anywhere in the model output? What is it called?

The standard errors of the model coefficients likewise require matrix multiplication to fully understand, but they are related to three factors: (1) the value of $\hat{\sigma}$, (2) the spread of the values of the corresponding covariate about

its mean (more spread will decrease the standard error), and (3) correlations between that covariate and other covariates in the model (tighter correlations will increase the standard error).

Question 8.6

The standard errors attempt to capture how much we expect our estimates of the model coefficients to vary if we were to refit the model using a different dataset, provided that the new dataset is similar to (i.e., sampled from the same population distribution as) the one used to fit the model. On average, approximately how much would we expect β_0 (the intercept) to deviate from its fitted value of 49.8600? How much would we expect β_1 and β_2 to deviate from their fitted values?

8.2.4 Hypothesis Tests of Coefficients

Armed with our coefficients and standard errors, we can perform a hypothesis test on each regression coefficient. Our null hypothesis in each case will be that the true value of that coefficient is zero: that is, it has no effect on the outcome. Under the null hypothesis that $\beta_j = 0$ and assuming n, our number of samples, is large enough, the quantity $\hat{\beta}_j / \text{se}(\hat{\beta}_j)$ will be distributed according to a Student's T distribution (see Chapter 4, Section 4.9) with $n - p - 1$ degrees of freedom.

Question 8.7

Sketch the null distributions for the hypothesis tests of our three regression coefficients, β_0, β_1, and β_2. Do you see why the p-values for these tests are so low?

8.2.5 Other Model Output

The software also provides some other output. The quantity **R-squared** is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{n}(y^{(i)} - \overline{y})^2}$$

$$= 1 - \frac{\sum_{i=1}^{n}(y^{(i)} - \hat{\beta}^T x^{(i)})^2}{\sum_{i=1}^{n}(y^{(i)} - \overline{y})^2}$$

or rather, the proportion of total variance in y explained by the model. The **adjusted R-squared** is almost exactly the same except it fixes a source of bias in R^2, namely that R^2 will favor models with more parameters. Adjusted R^2 penalizes models with more parameters. It is defined as:

$$R^2_{\text{adj}} = 1 - \frac{n-1}{n-p-1}\frac{\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2}{\sum_{i=1}^{n}(y^{(i)} - \overline{y})^2}$$

$$= 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

The **F-statistic** is the ratio of two variances: the variance in the outcome, y, that is explained by the model parameters ("sum of squares of regression", or SSR) and the residual, or unexplained variance ("sum of squares of error", or SSE). The corresponding **F-test** tests the null hypothesis that a model with no independent variables (that is, an intercept-only model with $\beta_1 = 0$ and $\beta_2 = 0$) fits the data as well as our model. The F-statistic follows an F distribution with p and $n - p - 1$ degrees of freedom (see Chapter 4, Section 4.10). The p-value reported in the model output is the p-value for this hypothesis test.

8.3 Example: Small Cities Pollution Dataset

The following data come from an early study that examined the possible link between air pollution and mortality. The authors examined 60 cities throughout the United States and recorded the following data:

MORT	Total age-adjusted mortality from all causes, in deaths per 100,000 population
PRECIP	Mean annual precipitation (in inches)
EDUC	Median number of school years completed for persons of age 25 years or older
NONWHITE	Percentage of the 1960 population that is nonwhite
NOX	Relative pollution potential of oxides of nitrogen
SO2	Relative pollution potential of sulfur dioxide

Note: "Relative pollution potential" refers to the product of the tons emitted per day per square kilometer and a factor correcting the SMSA dimensions and exposure.

We want to predict the value of `MORT` (y) using the predictors `PRECIP`, `EDUC`, `NONWHITE`, `NOX`, and `SO2` (x_1, x_2, x_3, x_4 and x_5). Here is the output for a fitted linear regression model:

```
Call:
lm(formula = MORT ~ PRECIP + EDUC + NONWHITE + NOX + SO2, data = d)

Residuals:
   Min     1Q Median     3Q    Max
-91.38 -18.97  -3.56  16.00  91.83

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 995.63646   91.64099  10.865 3.35e-15 ***
PRECIP        1.40734    0.68914   2.042 0.046032 *
EDUC        -14.80139    7.02747  -2.106 0.039849 *
NONWHITE      3.19909    0.62231   5.141 3.89e-06 ***
NOX          -0.10797    0.13502  -0.800 0.427426
SO2           0.35518    0.09096   3.905 0.000264 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 37.09 on 54 degrees of freedom
Multiple R-squared:  0.6746,    Adjusted R-squared:  0.6444
F-statistic: 22.39 on 5 and 54 DF,  p-value: 4.407e-12
```

Question 8.8

Interpret the values of each of these coefficients. Based on the coefficient values and their standard errors, which predictor(s) do you think have the greatest impact on mortality?

Question 8.9

In this model, is the effect of one predictor (say, `PRECIP`) impacted by the value(s) of any of the other predictor(s)? How does this differ from the other regression algorithms we've seen (KNN and decision trees)? What are the advantages and disadvantages of this choice?

Question 8.10

Is a normal distribution the right distribution to model an outcome of age-adjusted mortality (MORT)? Why or why not? Look back at our discussion of the normal distribution in Chapter 4 if you need a refresher.

Chapter 9

Interpreting a Logistic Regression Model

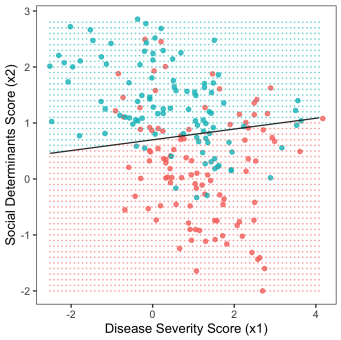
This chapter is similar to Chapter 8 but focuses on logistic regression models. As we saw in Chapter 8, linear regression models are used in situations where the outcome of a supervised learning problem, y, follows a normal distribution, conditional on the values of the predictors. **Logistic regression** models, in contrast, handle situations where the outcome, y, is binary: either 0 or 1. We first encountered these models as examples of classification algorithms in Chapter 2. Because of their popularity in the clinical domain, it's important to understand how these models are fit and how to interpret the summary output produced by software.

Unfortunately, a full understanding of logistic regression requires knowledge of maximum likelihood estimation. We will, therefore, skip over some of the details until we've had more time to explore this topic.

9.1 ER Readmissions Example from Chapter 2

In Chapter 2, we saw an example where information about two predictors – a disease severity score (x_1) and a social determinants score (x_2) – was used to predict a binary outcome: whether a patient would be readmitted to the ER within 30 days of discharge. We tried three different supervised learning

algorithms, one of which was a logistic regression model (Section 2.3.1). The output from that model is repeated below.




```
Call:
glm(formula = y ~ x1 + x2, family = "binomial", data = df)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.88232  -0.90614  -0.05965   0.86579   2.28489

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.9780     0.2945   3.321 0.000897 ***
x1            0.1344     0.1372   0.980 0.327272
x2           -1.3981     0.2316  -6.035 1.59e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 277.26  on 199  degrees of freedom
Residual deviance: 209.54  on 197  degrees of freedom
AIC: 215.54

Number of Fisher Scoring iterations: 4
```

9.2 Understanding the Model Summary

A logistic regression model looks like this (see also Section 2.3.1):

$$\log \frac{\mu}{1-\mu} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p \tag{9.1}$$

where μ is the mean of the Bernoulli distribution (see Section 4.3) governing our binary outcome, y; in other words, it is the probability that $y = 1$.

You will note that there is no independent error term here as there was in linear regression. That's because the variance and mean of a Bernoulli distribution are coupled and depend only on μ (again, see Section 4.3).

Question 9.1

In logistic regression, μ itself is not equal the sum of the predictors; instead, the **logit** of μ is their sum. Based on what you know about μ, why is a logistic

regression model not of the form

$$\mu = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p?$$

We will explore this further in Chapter 12.

Question 9.2

The decision boundary in logistic regression (see picture above) occurs where the sum of the linear predictors, $\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$, is zero. What value of μ does this correspond to? Why does this make sense, intuitively?

9.2.1 The Call

The first line of the output repeats the call you made to the `glm` function in R to fit the model. The `glm` package fits a variety of different generalized linear models using maximum likelihood estimation (see Chapter 5; this will also be discussed in more detail in Chapter 12). The `family = "binomial"` argument tells the function to fit a logistic regression model.

9.2.2 Coefficients and Standard Errors

Logistic regression models, like other GLMs, are fit using maximum likelihood (see Chapter 5 for a brief introduction). We will skip most of the details for now, but you can gain intuition by staring at Equation 9.1. This equation says that the model's predicted value of μ, the probability that the outcome will be positive ($y = 1$), is controlled by the values of the predictors and their coefficients $\beta_0, \ldots, \beta_p$.

By adjusting the values of the βs, the model causes μ to be high in regions of the feature space where $y = 1$ and low where $y = 0$. The values of the βs that do this the best are called the **maximum likelihood estimates**, and they are the coefficients shown in the model output.

As with linear regression, a full understanding of the standard errors requires matrix multiplication. However, they are related to the same factors that drive the standard errors in linear regression: (1) the spread of the values

of the corresponding covariate about its mean (more spread will decrease the standard error) and (2) correlations between that covariate and other covariates in the model (tighter correlations will increase the standard error).

Question 9.3

Looking at the form of the logistic regression model

$$\log \frac{\mu}{1-\mu} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$$

what does the value of each of the βs mean? What is β_j telling us about how y varies with the predictor j, all else being equal?

Question 9.4

The **odds** of something happening are defined as $\mu/(1-\mu)$, where μ is the probability that the thing occurs. In our example model, we are interested in the odds that $y = 1$ (the patient is readmitted). Does a unit increase in x_1 (disease severity score) increase or decrease the odds that a patient will be readmitted? What about x_2 (social determinants score)?

Question 9.5

What are the odds of readmission for a patient with:

(a) $x_1 = 0.1$ and $x_2 = 0.3$?

(b) $x_1 = 0.1$ and $x_2 = -1.3$?

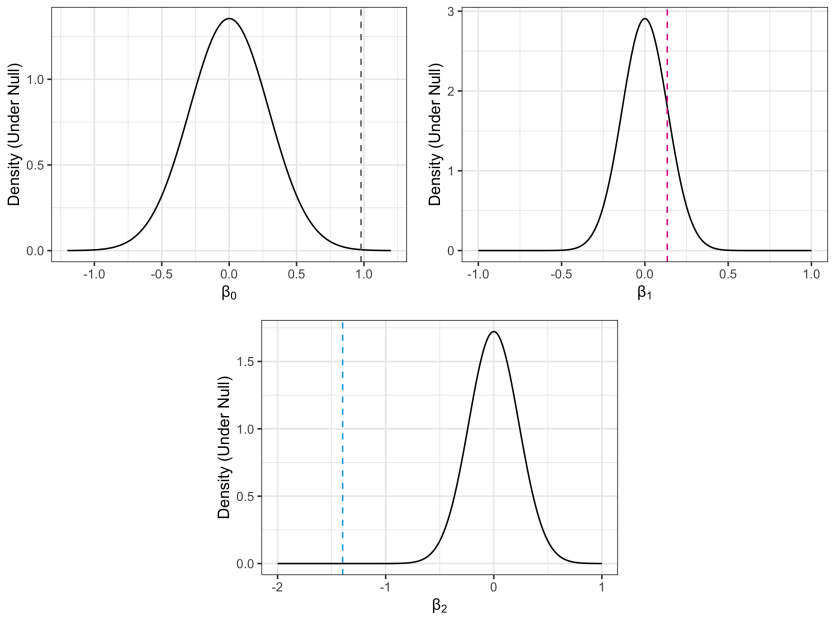
(c) $x_1 = 1.1$ and $x_2 = 0.3$?

9.2.3 Hypothesis Tests of Coefficients

Just as in linear regression, we can use our coefficients and standard errors to perform a hypothesis test on each regression coefficient, β_j, against the null hypothesis that $\beta_j = 0$ (the predictor x_j has no effect on the outcome). In logistic regression, the quantity $\hat{\beta}_j / \text{se}(\hat{\beta}_j)$ will follow a normal distribution under the null.

Question 9.6

Below are the null distributions for the hypothesis tests of our three regression coefficients, β_0, β_1, and β_2. In each graph, the maximum likelihood estimate of the coefficient is shown as a vertical dashed line. Based on these graphs, can you tell why the p-values for β_0 and β_2 are low and the one for β_1 is high? What is the intuition behind this?



9.2.4 Deviance and Deviance Residuals

The **deviance** (called **residual deviance** in the model output) plays a role in GLMs akin to that of the residual standard error in linear regression; it is a measure of the residual variation in the outcome not explained by the model. The **null deviance** is the deviance for a model that only includes an intercept. Under the null hypothesis that all of the βs are zero except the intercept (i.e, a model with no predictors explains the data as well as our model), the difference

$$\text{null deviance} - \text{residual deviance}$$

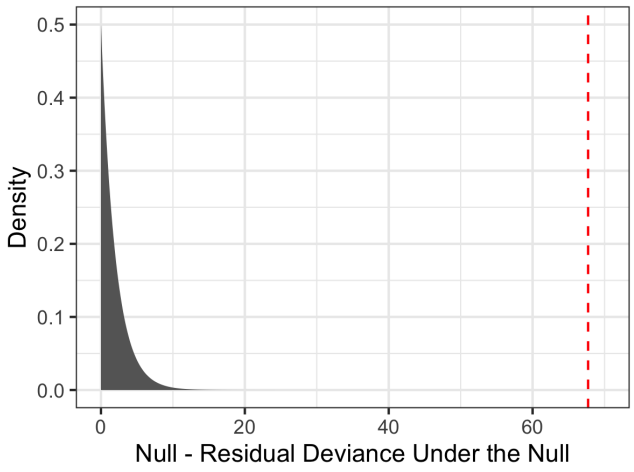
is distributed as χ^2_p, a chi-squared distribution (see Section 4.8) with p degrees of freedom, where p is the number of predictors.

Question 9.7

This test is a hypothesis test of the null hypothesis that a model with no predictors fits our data as well as our model, where goodness of fit is measured by the deviance (lower is better). What is this hypothesis test akin to in the linear regression model output?

Question 9.8

The difference in null and residual deviances in this case is 67.72. It follows a χ^2_2 distribution under the null. A plot of the χ^2_2 distribution and our test statistic is shown below. What do these findings indicate about the p-value of this goodness of fit test and what does it mean?



In the GLM context, there are multiple types of residual (more on this later). **Deviance residuals** quantify the contributions of the individual samples to the deviance. Unfortunately, the output from `glm` is confusing because what `glm` calls a deviance residual in the model summary is actually something called a **working residual**. We will ignore this part of the output until we understand more about the inner workings of GLMs.

9.3 Example: Low Birthweight Dataset

The goal of this study was to identify risk factors associated with giving birth to a low birth weight baby (a baby weighing less than 2500 grams). Infant mortality rates and birth defect rates are very high for low birth weight babies. A woman's behavior during pregnancy (including diet, smoking habits, and receiving prenatal care) can greatly alter the chances of carrying the baby to term and, consequently, of delivering a baby of normal birth weight.

Data were collected on 189 women, 59 of which had low birth weight babies and 130 of which had normal birth weight babies[1].

LOW	Low birth weight (0 = birth weight $\geq$ 2500 g; 1 = birth weight $<$ 2500 g)
AGE	Age of mother in years
LWT	Mother's weight in pounds at last menstrual period
RACE	Race (1 = white, 2 = black, 3 = other)
SMOKE	Smoking status during pregnancy (1 = yes, 0 = no)
PTL	History of premature labor (0 = none, 1 = one, etc.)
HT	History of hypertension (0 = no, 1 = yes)
UI	Presence of uterine irritability (0 = no, 1 = yes)
FTV	Number of physician visits during the first trimester
BWT	Birth weight in grams

We will build a model that predicts the value of LOW based on all of the other covariates except, of course, BWT. (Why not use BWT?)

[1]SOURCE: Hosmer and Lemeshow (2000) *Applied Logistic Regression: Second Edition*. Data were collected at Baystate Medical Center, Springfield, Massachusetts during 1986.

```
Call:
glm(formula = LOW ~ AGE + LWT + RACE + SMOKE + PTL + HT + UI +
    FTV, family = "binomial", data = d)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.8946  -0.8212  -0.5316   0.9818   2.2125

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.480623   1.196888   0.402  0.68801
AGE         -0.029549   0.037031  -0.798  0.42489
LWT         -0.015424   0.006919  -2.229  0.02580 *
RACE2        1.272260   0.527357   2.413  0.01584 *
RACE3        0.880496   0.440778   1.998  0.04576 *
SMOKE        0.938846   0.402147   2.335  0.01957 *
PTL          0.543337   0.345403   1.573  0.11571
HT           1.863303   0.697533   2.671  0.00756 **
UI           0.767648   0.459318   1.671  0.09467 .
FTV          0.065302   0.172394   0.379  0.70484
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 234.67  on 188  degrees of freedom
Residual deviance: 201.28  on 179  degrees of freedom
AIC: 221.28

Number of Fisher Scoring iterations: 4
```

Question 9.9

In this model, is the effect of one predictor (say, `AGE`) impacted by the value(s) of any of the other predictor(s)? How does this differ from the other classification algorithms we've seen (KNN and decision trees)? What are the advantages and disadvantages of this choice?

Question 9.10

Comment on how the variable `RACE` enters into the model here. Does this make sense in light of what that variable means and how it potentially interacts with the other study variables?

Question 9.11

Interpret the values of each of these coefficients. Based on the coefficient values and their standard errors, which predictor(s) do you think have the greatest impact on whether or not a woman has a low birthweight baby?

Chapter 10

A Brief Note on Feature Engineering

All machine learning algorithms and statistical models depend on the concept of a **feature**. A feature is some aspect of a dataset that, the model designer believes, represents the data in a way that is relevant to the problem he/she is trying to solve.

Before any algorithm can be applied, therefore, it is necessary to decide how to represent the data: which features to include and how to extract them from the raw data. This task is called **feature engineering**.

10.1 Study Design vs. Feature Engineering

We have seen a large number of features in Chapters 1–9, but we never stopped to consider them. That's because, in many datasets, the features are chosen at the **study design** stage. The analyst (statistician, data scientist, etc.) has no say in what the features look like or which features are included.

This paradigm is changing as data science increasingly focuses on large, observational datasets, like those from electronic medical records (EMRs). In these types of studies, the raw data were not collected for the study itself, but to fulfill some other purpose. The analyst must choose how to build features

from the raw data and use them in models.

Question 10.1

The examples in Chapters 2 and 3 used the same two features. What were these features? How were they represented? What are some alternatives to this choice of features?

Question 10.2

In Chapter 7, we looked at the Wisconsin Breast Cancer Dataset, which includes 30 different imaging features relevant to predicting whether a tumor is benign or malignant. How were these features represented? What are some alternatives to this choice?

Question 10.3

In Chapters 8 and 9, we looked at two datasets that were collected for the purposes of answering particular questions. Do you agree with these study designers' choice of features? What other features could potentially have been relevant to answering each research question?

10.2 Turning Data into Numbers

A model is just a tool for learning relationships among sets of numbers. The first step in any data science problem, therefore, is deciding how to represent what is often a large, complex, noisy dataset as a set of numbers.

10.2.1 Numbers

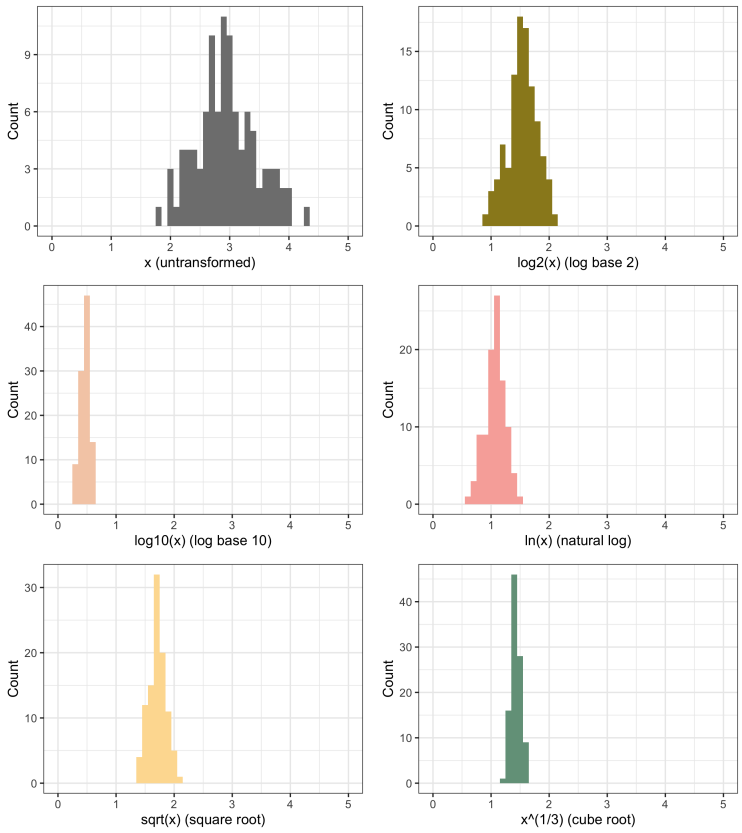
Sometimes you get lucky and the feature you need is already a number, such as a vital sign measurement, lab value, or other biomarker. In that case, more often than not, the feature enters into the model as its raw value.

In some cases, you may also choose to apply a **transformation** to the feature before it enters the model. A transformation is simply the application of a deterministic mathematical function that changes the shape of the distribution

of the feature. Transformations are often used to improve the interpretability of a model and/or to ensure that the model fulfills the assumptions of the statistical inference method(s) being used (e.g., a hypothesis test).

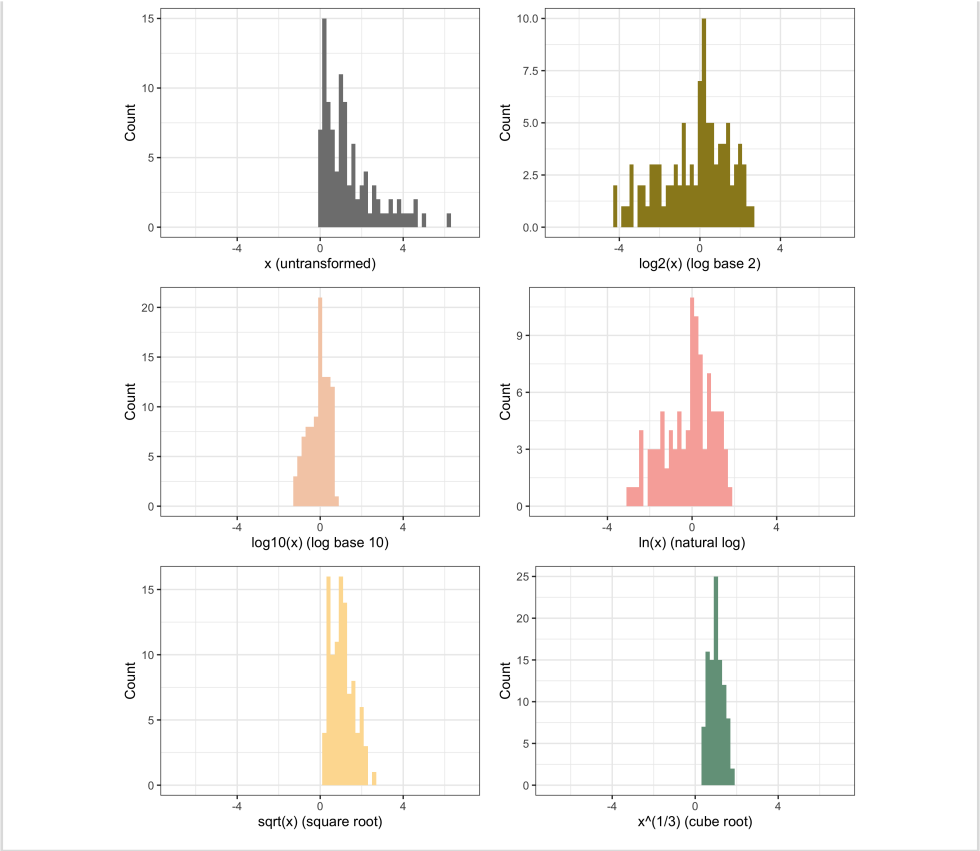
Question 10.4

Here are 100 random samples from a normal distribution with $\mu = 3.0$ and $\sigma = 0.5$ and five different transformations of those samples. What do you notice about the shape and position of the data under the different transformations?



Question 10.5

Here are 100 random samples from an exponential (see Section 4.7) distribution with $\lambda = 0.8$ and the same five transformations of those samples. What do you notice about the shape and position of the data under the different transformations?



Economics, the social sciences, and related disciplines, which are heavily dependent on the use of regression models and hypothesis tests, rely extensively on transformations. In my experience, machine learning folks spend almost no time on them because their primary concern is predictive accuracy, not model interpretation. Machine learning practitioners, however, very frequently **scale and center** their predictors (see footnote in Section 12.5), which is another type of transformation. We will get into more detail on transformations as we continue to learn about regression models.

10.2.2 Binary Variables

For features which are yes/no (e.g., presence/absence of a disease, symptom, physical attribute, etc.) the most common coding scheme is to use "1" for "yes" and "0" for "no". This is useful for interpretation, particularly in regression

models. In a linear regression model using this coding scheme, for example, the model coefficient will be the shift in the mean of the normal distribution representing the outcome, y, when the feature is present.

10.2.3 Categories

Categorical features with $k > 2$ categories are generally represented using **indicator variables**. If a feature, x, has k **levels**, we can use $k - 1$ yes/no indicator variables to represent that feature. For example, assume $k = 3$ and the possible levels of our feature, x, are A, B, and C. We set:

$$x_1 = \begin{cases} 1 & \text{if } x = A \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if } x = B \\ 0 & \text{otherwise} \end{cases}$$

If the value of x is A, $x_1 = 1$ and $x_2 = 0$. If it's B, $x_1 = 0$ and $x_2 = 1$. The value C is called our **reference category** and has $x_1 = 0$ and $x_2 = 0$. In this way, information about all three categories is captured using only two variables. Creating indicator variables is just another way of transforming the value of a feature.

Question 10.6

In Section 9.3, we saw an example of a model that predicts whether or not a mother will give birth to a low birthweight baby. One of the factors considered in that model is the mother's race, which was coded (crudely and probably inaccurately, I might add) as `1 = white, 2 = Black, 3 = other`. You can tell how the feature `RACE` was coded by examining the model output. How many indicator variables were used? Which level of the feature was used as the reference category?

Chapter 11

Survival Data and the Kaplan-Meier Curve

We have already investigated supervised learning models and hypothesis tests in cases where the outcome of interest is a category or number. But what if the outcome is a *time duration*? For example, what if we're comparing the effects of two treatments and our outcome is the time between treatment administration and disease progression?

Data where the outcome is a time duration are very common in clinical data science and are called **time-to-event** data or **survival data**. The field of **survival analysis** develops methods to analyze and interpret such data. We will examine one such method today and many more in subsequent chapters.

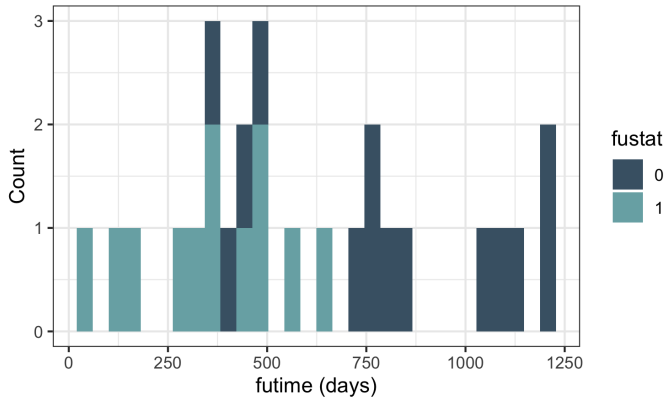
11.1 Example: Ovarian Cancer Survival Dataset

Today we'll examine some data from a study of ovarian cancer[1]. The dataset contains information on 26 women. The variables are:

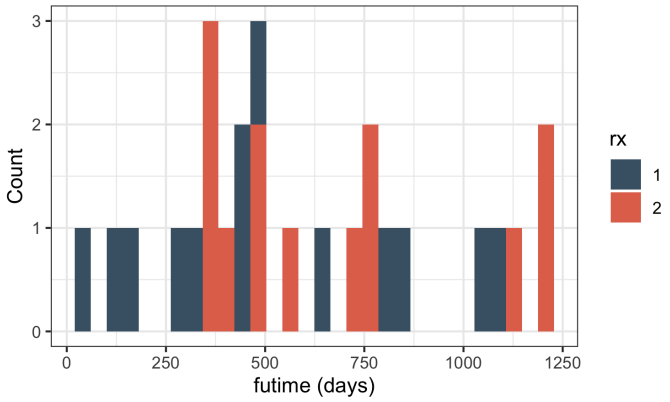
[1]The dataset comes from the `survival` package in R and is labeled `ovarian`. The original study is Edmonson JH *et al*, "Different chemotherapeutic sensitivities and host factors affecting prognosis in advanced ovarian carcinoma versus minimal residual disease", *Cancer Treatment Reports*, 63(2): 241-247; 1979.

- `futime`: The number of days from enrollment in the study until death or censoring, whichever came first
- `fustat`: An indicator of death (1) or censoring (0)
- `age`: The patient's age in years at the time of treatment administration
- `resid.ds`: Residual disease present at the time of treatment administration (1 = no, 2 = yes)
- `rx`: Treatment group (1 = cyclophosphamide, 2 = cyclophosphamide + adriamycin)
- `ecog.ps`: A measure of performance score or functional status at the time of treatment administration, using the Eastern Cooperative Oncology Group's (ECOG) scale. It ranges from 0 (fully functional) to 4 (completely disabled). Level 4 subjects are usually considered too ill to enter a randomized trial such as this. The patients in this dataset are all at Levels 1 and 2.

Here is a histogram of the follow-up times (`futime`) in days, colored according to whether the patient died or was censored (`fustat`):



And here is the same graph colored by treatment group (`rx`):



Now, imagine that we want to study the effect of the treatment group (`rx`) on the outcome of death or no death (1 = death, 0 = no death). We could think of this as a classification problem with only a single feature: treatment group. Unfortunately, this method of analyzing time-dependent data is fraught with problems:

1. How do you choose the time horizon at which to evaluate mortality?
2. How do you handle people who dropped out of the study before that time?

11.2 Definitions

Censoring occurs when the event of interest in a time-to-event analysis is not observed. It is a form of missing data problem (see Chapter **??**) and can be caused by a variety of factors, including inconsistencies in follow-up, the study's ending before all subjects have experienced the event, or a lack of knowledge about when, exactly, the event occurred. The type of censoring represented in the `ovarian` dataset is called **right-censoring**. We will focus on right-censoring today and investigate other types later.

> **Right censoring:** A situation that arises when the event of interest has not occurred by the end of the follow-up period. This may be because (a) the study itself ends, (b) a patient is lost to follow-up

during the study period, or (c) a patient experiences a different event that makes further follow-up impossible[2].

Survival data are generally described using two probabilities, called the survival and hazard.

Survival: Also called the **survival function** or **survival probability** and abbreviated $S(t)$, this is the probability that an individual survives to time t (i.e., does not experience the event by time t).

Hazard: Usually denoted by $h(t)$ or $\lambda(t)$, this is the probability that an individual who has not yet experienced the event at time t experiences it at that exact time. In other words, it is the instantaneous event rate for an individual who has already survived to time t.

We will focus on the survival function now and learn more about the hazard later.

11.3 The Kaplan-Meier Estimator

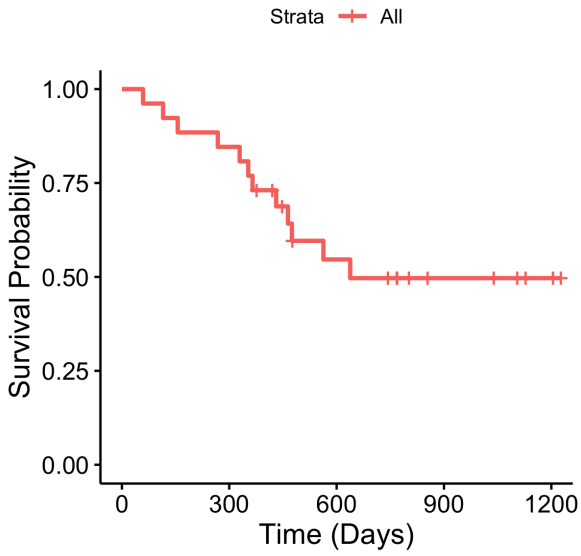
The **Kaplan-Meier estimator** is a nonparametric estimate of the survival function, usually represented graphically by a **Kaplan-Meier curve**[3]. The Kaplan-Meier estimator looks like this:

$$\hat{S}(t) = \prod_{j|t_j \leq t} \frac{n_j - d_j}{n_j}$$

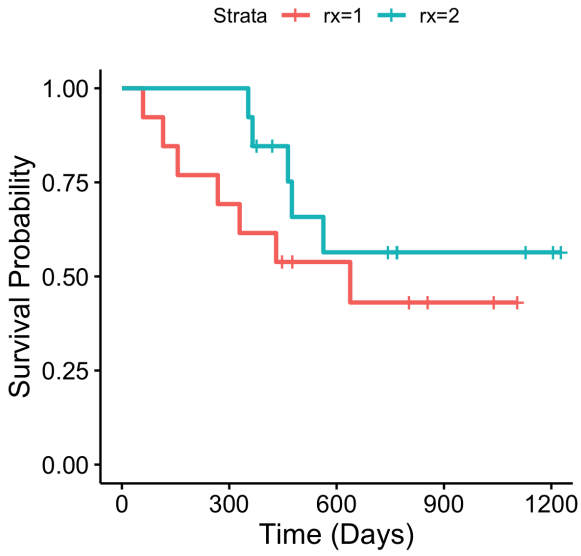
where d_j is the number of subjects who fail at time t_j and n_j is the number of subjects at risk just prior to t_j. Here is a Kaplan-Meier curve for the `ovarian` dataset. The little "+" signs correspond to censoring events.

[2]For more information, please see Clark TG *et al*, "Survival Analysis Part I: Basic Concepts and First Analyses", *British Journal of Cancer*, 89, 232–238; 2003.

[3]It can be shown mathematically that the Kaplan-Meier estimator is the maximum likelihood estimator (see Chapter 5) of the survival function in the case of censoring.



And here are Kaplan-Meier curves for the two treatment groups separately:



Question 11.1

Here are the raw data from treatment group 1 of the `ovarian` dataset. Using these data, fill in the remaining cells of the table below.

	rx	futime	fustat
1	1	59	1
2	1	115	1
3	1	156	1
4	1	268	1
5	1	329	1
6	1	431	1
7	1	448	0
8	1	477	0
9	1	638	1
10	1	803	0
11	1	855	0
12	1	1040	0
13	1	1106	0

j	t_j	n_j	d_j	$\hat{S}(t_j)$	Calculation
0	0	13	0	1.000	$\frac{13-0}{13}$
1	59	13	1	0.923	$\hat{S}(t_0)\left(\frac{13-1}{13}\right)$
2	115	12	1	0.846	$\hat{S}(t_1)\left(\frac{12-1}{12}\right)$
3	156				
4	268				
5	329	9	1	0.615	$\hat{S}(t_4)\left(\frac{9-1}{9}\right)$
6	431	8	1	0.538	$\hat{S}(t_5)\left(\frac{8-1}{8}\right)$
7	448	7	0	0.538	$\hat{S}(t_6)\left(\frac{7-0}{7}\right)$
8	477	6	0	0.538	$\hat{S}(t_7)\left(\frac{6-0}{6}\right)$
9	638	5	1	0.431	$\hat{S}(t_8)\left(\frac{5-1}{5}\right)$
10	803	4	0		
11	855	3	0		
12	1040	2	0		
13	1106	1	0		

Question 11.2

Based solely on the Kaplan-Meier curves for the two treatment groups, which treatment appears to prolong survival more effectively?

11.4 Assumptions of the Kaplan-Meier Estimator

The Kaplan-Meier estimator makes three important assumptions:

1. The probability of censoring is unrelated to the outcome of interest.
2. The survival probabilities are the same for participants recruited at different times during the study (e.g., circumstances that could alter the survival, such as treatments, do not change over calendar time).
3. The events occurred at exactly the times specified.

Question 11.3

What is one way each of these assumptions could be violated?

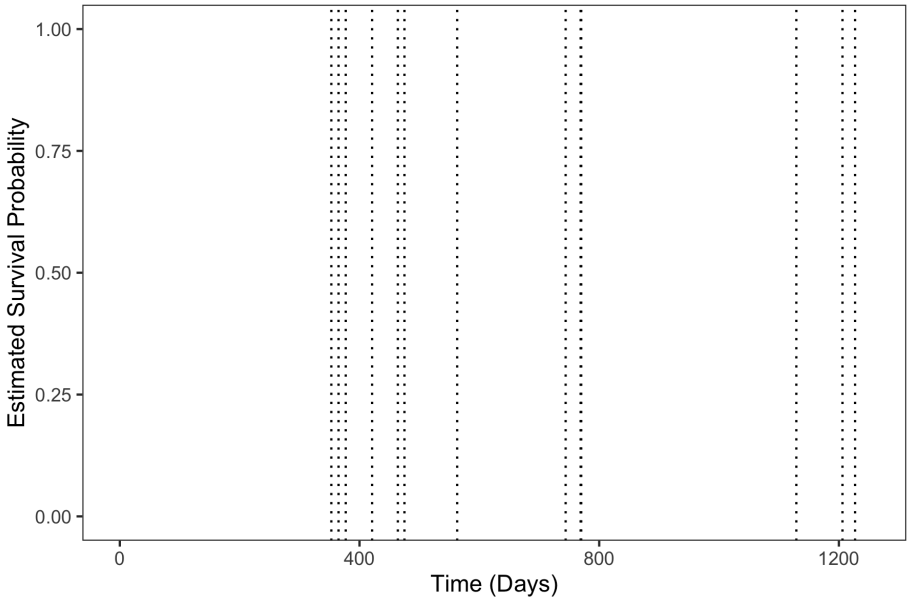
11.5 Comparing Kaplan-Meier Curves

Of course, now the question arises: How do we formally compare two Kaplan-Meier curves? There is a nonparametric hypothesis test for comparing Kaplan-Meier curves called the log-rank test; we will see it in Chapter **??**. There is also an entire family of linear models, called Cox proportional hazards models, that use the Kaplan-Meier curve as their backbone and model the effects of different covariates on this curve. We will see them in Chapter 18.

Question 11.4

Here are the data for treatment group 2 of the `ovarian` dataset. Perform the calculations of $\hat{S}(t_j)$ for $j = 0, \ldots, 13$, starting with $t_0 = 0$. Draw the Kaplan-Meier curve, adding symbols for the censoring events.

	rx	futime	fustat
1	2	353	1
2	2	365	1
3	2	377	0
4	2	421	0
5	2	464	1
6	2	475	1
7	2	563	1
8	2	744	0
9	2	769	0
10	2	770	0
11	2	1129	0
12	2	1206	0
13	2	1227	0



Chapter 12

Generalized Linear Models

Linear and logistic regression, which we have seen already in Chapters 2, 3, 8, and 9, are members of a broader class of supervised learning models called **generalized linear models (GLMs)**. In GLMs, the outcome variable, y, is assumed to follow a probability distribution of a particular type. For example, in linear regression, y follows a normal distribution. In logistic regression, y is binary ($y \in \{0, 1\}$) and follows a Bernoulli distribution[1]. The expected value, or mean, of the outcome distribution is related to a **linear combination** of the predictors, $\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$, via a model-specific **link function**.

GLMs, like maximum likelihood (Chapter 5), are normally considered an advanced topic. However, they provide a nice example of how the same formalism – modeling the response variable using a probability distribution, assuming a certain form for the predictors, optimizing the whole thing using maximum likelihood – can be applied to solve different-looking problems. They are also a good entryway into the sorts of optimization tasks performed by graphical models and deep learning algorithms.

12.1 Model Assumptions

GLMs require us to make several assumptions which affect both our choice of model and our interpretation of model output:

[1] In **grouped** logistic regression, it follows a binomial distribution.

1. We assume that the outcome follows a certain type of distribution (e.g. Bernoulli distribution for a logistic regression model, normal for linear, etc.) conditional on the predictors. This assumption is baked into the model structure. It is, therefore, important to consider whether the outcome distribution you chose actually makes sense for your particular problem. It is generally not advisable to use a linear regression model, for example, when your outcome is a count.

2. We assume that the predictors are fixed and known, and thus have no error associated with their measurements[2].

3. We assume that the predictors enter the model as a linear combination, $\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$. This is why GLMs are referred to as "linear models".

4. We assume that the n samples in our dataset are collected independently, so that the errors of the n sample outcomes are uncorrelated[3].

12.2 Notation for the Predictors

As mentioned above, GLMs assume that the predictors enter the model as a linear combination. A linear combination is an expression constructed from a set of terms by multiplying each term by a constant and adding the results. We denote the number of predictors in the model by p and the vector of predictors by x, where

$$x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix}$$

and we have included a "1" as the first element to allow for an **intercept**. We write $x^{(i)}$ to denote the vector of predictors associated with the ith training example. The coefficients of the linear combination (i.e. the model parameters

[2]Bayesian versions of these models relax this assumption.

[3]Think back to our formulation of the likelihood in Chapter 5 and how it depended on the samples' being independent and identically distributed, or iid.

we are hoping to learn) are denoted by:

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$$

and we often express the linear combination as an **inner product**, or **dot product**, of the two vectors, written as

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

This is just notational shorthand.

Question 12.1

We saw the details of linear and logistic regression models in Chapters 8 and 9 and discussed the limitations of predictors' entering as a linear combination. What are some of those limitations?

Question 12.2

Just to confirm that you understand this notation, write out the form of $\beta^T x$ for a model with (a) one predictor, (b) three predictors. Write both the general form and the form for one training example, $x^{(i)}$.

12.3 Modeling the Outcome

Generalized linear models model the expected value of the outcome, $E[y]$, as a function of this linear combination of predictors.

12.3.1 Linear Regression

In linear regression, we assume that the outcome, y, follows a normal distribution (see Section 4.2), whose mean is controlled by the values of the predictors.

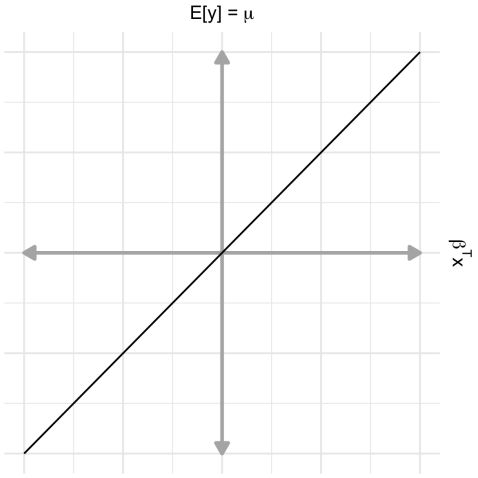
Recall that the normal distribution is a continuous probability distribution with the following properties:

$$p(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}} \qquad E[y] = \mu \qquad \text{var}(y) = \sigma^2$$

where $y \in \mathbb{R}$. Its mean, μ, can be any real number. To link μ to the predictors, therefore, we simply set it equal to $\beta^T x$, like so:

$$E[y] = \mu = \beta^T x \tag{12.1}$$

This is called using the **identity link**. The relationship between $E[y] = \mu$ and $\beta^T x$ is shown below.



Question 12.3

In this model, how much does the mean of the outcome distribution, μ, change as you vary each predictor? For example, if you have $p = 3$ predictors, by how much does μ change as the value of x_2 changes by one unit (for example, from 1 to 2)? How much does μ change as the value of x_2 changes from 3 to 4? What about x_1 and x_3?

12.3.2 Logistic Regression

In logistic regression the outcome, y, is either 0 or 1. We model it using the Bernoulli distribution (see Section 4.3), which is a discrete probability distribution with the following properties:

$$p(y) = \mu^y(1-\mu)^{1-y} \qquad E[y] = \mu \qquad \text{var}(y) = \mu(1-\mu)$$

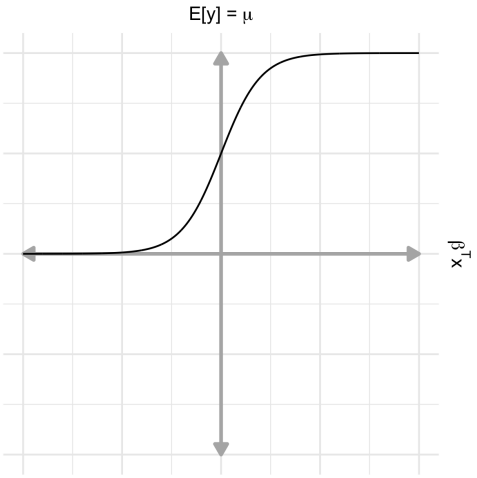
where $y \in \{0, 1\}$. Because μ is a probability, it must be a real number between 0 and 1. No matter how large or small $\beta^T x$ gets, the value of $E[y] = \mu$ cannot be outside this range. We therefore apply the **logistic function**, $f(x) = 1/(1+\exp(-x))$, which has the range $(0, 1)$, to $\beta^T x$ to squash it:

$$E[y] = \mu = \frac{1}{1+\exp\left(-\beta^T x\right)} \tag{12.2}$$

The relationship between $E[y]$ and $\beta^T x$ is shown below. We typically invert the model to write

$$\log \frac{\mu}{1-\mu} = \beta^T x.$$

The function $\log\left(\mu/(1-\mu)\right)$ is called the logit, and we say we use the **logit link**.



Question 12.4

Let's revisit Question 9.1. Now that we've described logistic regression in the framework of GLMs, what more can you say about why the model is not of the form

$$\mu = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p?$$

12.3.3 Poisson Regression

In Poisson regression, the outcome is a count. We model the outcome using a Poisson distribution, which is a discrete probability distribution with the following properties (Section 4.5):

$$p(y) = \frac{e^{-\lambda}\lambda^y}{y!} \qquad E[y] = \lambda \qquad \text{var}(y) = \lambda$$

where $y \in 0, 1, 2, \ldots$. Because λ, the mean of the outcome distribution, is the expected value of a count, it must be a real number greater than or equal to zero. In particular, no matter how small $\beta^T x$ gets, the value of $E[y] = \lambda$ cannot be negative. We therefore exponentiate $\beta^T x$ to ensure that λ is greater than zero:

$$E[y] = \lambda = \exp(\beta^T x) \tag{12.3}$$

The relationship between $E[y]$ and $\beta^T x$ is shown below. We typically invert the model to write

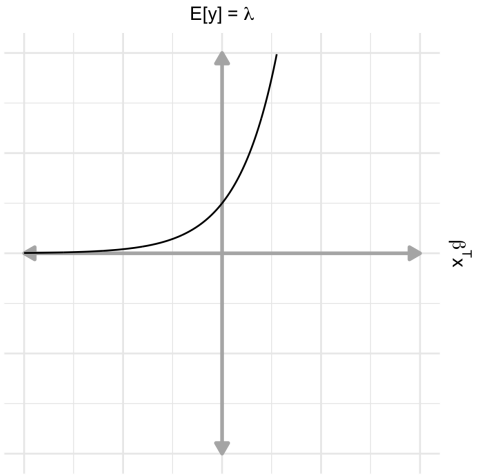
$$\log(\lambda) = \beta^T x$$

which is the standard form of the Poisson regression model. We say these models use the **log link**.

Question 12.5

There are many other generalized linear models. In each case, the mean (expected value) of a probability distribution is related, via a link function, to a linear combination of the predictors.

Knowing this, how would you create a GLM where the outcome follows an exponential distribution (Section 4.7)? Which link would you use?



12.4 Maximum Likelihood for GLMs

GLMs are fit using maximum likelihood estimation (see Chapter 5). A full treatment of MLE for GLMs is outside the scope of these notes, but I've put the start of the calculations for each type of model below. The only difference between these calculations and those in Chapter 5 is that now our parameters of interest, the means of our outcome distributions, are functions of our predictors $x_1, \ldots, x_p$. Our job is to find the coefficients on those predictors, $\beta_0, \ldots, \beta_p$, that provide the best fit between our model and our training data.

12.4.1 Linear Regression

The likelihood for the linear regression model is:

$$\mathcal{L}(\mu^{(1)}, \ldots, \mu^{(n)}, \sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y^{(i)} - \mu^{(i)})^2}{2\sigma^2}\right]$$

where we use $\mu^{(i)}$ to represent the model's estimate of the mean of the outcome at the position of training example i. We can use Equation 12.1 to rewrite this

as a function of the predictors:

$$\mathcal{L}(\beta, \sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(y^{(i)} - \beta^T x^{(i)})^2}{2\sigma^2}\right]$$

Taking the log, we obtain the log-likelihood:

$$\log \mathcal{L}(\beta, \sigma) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y^{(i)} - \beta^T x^{(i)}\right)^2$$

Taking derivatives of the log-likelihood with respect to the βs, we find that we can maximize the likelihood by minimizing the sum-squares: $\sum_{i=1}^{n}\left(y^{(i)} - \beta^T x^{(i)}\right)^2$.

Question 12.6

Take a minute to stare at this result. When most people learn linear regression, they learn that these models are fitted by minimizing the sum of squared residuals (see Chapter 8). Indeed, linear regression models predate GLMs and are typically fit using ordinary least squares, not maximum likelihood. If you fit a linear regression model in R using the `lm` package, you're using OLS. If you use the `glm` package with the argument `family = "gaussian"`, you're using maximum likelihood. However, both methods will produce the same fitted models. Do you see why this is?

12.4.2 Logistic Regression

The likelihood for the logistic regression model is:

$$\mathcal{L}(\mu^{(1)}, \ldots, \mu^{(n)}) = \prod_{i=1}^{n} \mu^{(i)^{y^{(i)}}}(1 - \mu^{(i)})^{1-y^{(i)}}$$

Rewriting this as a function of the predictors, we get:

$$\mathcal{L}(\beta) = \prod_{i=1}^{n}\left(\frac{1}{1+\exp(-\beta^T x^{(i)})}\right)^{y^{(i)}}\left(\frac{\exp(-\beta^T x^{(i)})}{1+\exp(-\beta^T x^{(i)})}\right)^{1-y^{(i)}}$$

Taking the log, we obtain the log-likelihood:

$$\log \mathcal{L}(\beta) = \sum_{i=1}^{n} \left[y^{(i)} \beta^T x^{(i)} - \log \left(1 + \exp(\beta^T x^{(i)}) \right) \right]$$

Again, we will take derivatives of the log-likelihood with respect to the βs to maximize it. However, we cannot solve for the optimal βs analytically in this case. Numerical optimization methods are used to find the maximum likelihood estimates, $\hat{\beta}_0, \hat{\beta}_1$, etc.

12.4.3 Loglinear (Poisson) Regression

The likelihood for the Poisson regression model is:

$$\mathcal{L}(\lambda^{(1)}, \ldots, \lambda^{(n)}) = \prod_{i=1}^{n} \frac{{\lambda^{(i)}}^{y^{(i)}} e^{-\lambda^{(i)}}}{y^{(i)}!}$$

Rewriting this as a function of the predictors, we get:

$$\mathcal{L}(\beta) = \prod_{i=1}^{n} \frac{\exp\left(y^{(i)} \beta^T x^{(i)}\right) e^{-\exp\left(\beta^T x^{(i)}\right)}}{y^{(i)}!}$$

Taking the log, we obtain the log-likelihood:

$$\log \mathcal{L}(\beta) = \sum_{i=1}^{n} \left[y^{(i)} \beta^T x^{(i)} - \exp(\beta^T x^{(i)}) - \log(y^{(i)}!) \right]$$

As with logistic regression, we cannot solve for the optimal βs analytically; numerical optimization methods are used.

Question 12.7

Think of the log-likelihood as measuring the height of a hill. Your data,

$$\left\{ x^{(1)}, \ldots, x^{(n)} \right\}$$

don't change, so we don't care about their effect on the height. What we care about are the parameters, $\beta_0, \ldots, \beta_p$. For each combination of those $p + 1$ parameters, the height changes. We want to find the combination of parameters

that puts us at the top of the hill.

The first derivative of the log-likelihood with respect to one of the parameters, β_j, is

$$\frac{\partial \log \mathcal{L}}{\partial \beta_j}$$

and the vector of all of these first derivatives for $\beta_0, \ldots, \beta_j$ is called the **gradient**. Evaluated at a particular set of parameters, the gradient tells you how steep your hill is in the direction of each of your $p + 1$ parameters. How could you use this information to maximize the likelihood? You don't need to do any math. Just say how you would do it.

Question 12.8

There are many different numerical optimization algorithms that one can use to maximize the likelihood (i.e., find the top of the hill). One of them is called **Fisher scoring**. Examine the output of the logistic regression models in Chapter 9 and the Poisson regression model shown below in Section 12.6. Where do you see the term "Fisher scoring"? What do you think the term "Fisher scoring iterations" refers to?

12.5 Standard Errors and Hypothesis Tests

Here, once again, is the summary output from a logistic regression model of the ER readmissions example from Chapter 2, reprinted again in Section 9.1:

```
Call:
glm(formula = y ~ x1 + x2, family = "binomial", data = df)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.88232  -0.90614  -0.05965   0.86579   2.28489

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)   0.9780     0.2945   3.321 0.000897 ***
x1            0.1344     0.1372   0.980 0.327272
x2           -1.3981     0.2316  -6.035 1.59e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 277.26  on 199  degrees of freedom
Residual deviance: 209.54  on 197  degrees of freedom
AIC: 215.54

Number of Fisher Scoring iterations: 4
```

As we discussed in Chapter 9, the magnitudes of the coefficients in these models matter, but they are only important in relation to:

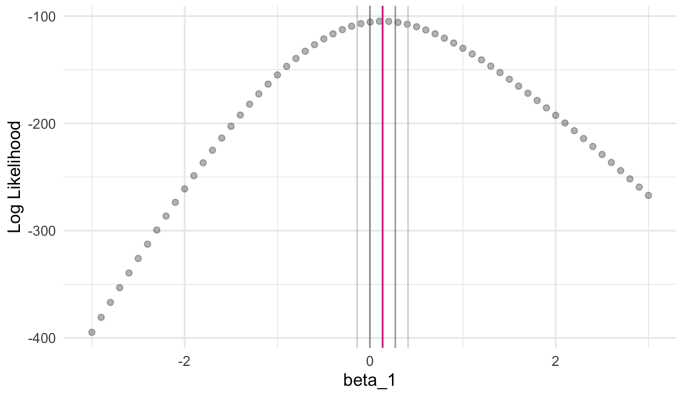
1. The scale on which the predictors are measured.
2. The amount of uncertainty the model has about their values.

For example, if a predictor varies only across a tiny range of values, its model coefficient may be large, since it quantifies the change in the link-function-transformed outcome when the predictor changes by 1.0. However, that doesn't mean that the predictor itself is important to the outcome[4].

Similarly, the model may be highly uncertain about a coefficient's value, owing to factors like a small dataset (small n) or collinearity (correlations) among the predictors. Mathematically, high uncertainty means that the value of the likelihood doesn't change very rapidly as you move away from the maximum likelihood estimate of a coefficient. For example, here is how the

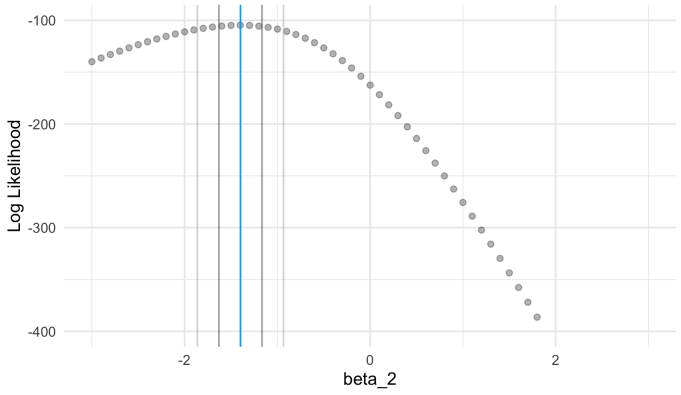
[4]This is one reason many advocate **scaling** and **centering** predictors before fitting a model. Centering means subtracting the mean value of a predictor from all of its individual measurements so that the mean of each centered predictor is zero. Scaling means dividing the values of each predictor by their standard deviation, so that the standard deviation of each predictor is 1.0. This enables the relative magnitudes of the model coefficients to be compared directly.

log-likelihood for the logistic regression example above changes when we vary β_1 (the coefficient of x_1), keeping β_0 (the intercept) and β_2 (the coefficient of x_2) fixed at their MLEs:



The gray vertical lines are related to the **standard error** of the model coefficient, which is in turn related to the "flatness" of the likelihood surface around the MLE. The gray lines are situated at 1 and 2 standard errors away from the MLE in either direction. You can see that in the case of β_1, the gray lines overlap zero. The value zero (no effect) is a plausible estimate of the impact of x_1 on the outcome.

Contrast this with how the log-likelihood varies around the MLE for β_2:



Here the standard error is larger, but the magnitude of the coefficient is also larger, so the range of the gray lines does not overlap zero.

Question 12.9

These findings are reflected in the relative values of the Z-statistic (`z value`) and P-value (`Pr(>z|)`) in the model output for the two coefficients. With that in mind, let's reconsider Question 9.6. How do these likelihood plots and the null distributions shown in Question 9.6 convey the same information?

12.6 Example: Nesting Horseshoe Crabs Dataset

Let's examine some output from a Poisson regression model, which is a type of GLM with which you may not already be familiar.

These data come from a study of nesting horseshoe crabs. Each of the 173 observed female horseshoe crabs had a male crab resident in her nest. The study investigated factors affecting whether the female crab had any other males, called *satellites*, residing nearby. (Source: Agresti, *Categorical Data Analysis*, Table 4.3. Data courtesy of Jane Brockmann, Zoology Department, University of Florida; study described in *Ethology* **102**: 1-21, 1996.)

```
SATELL  Number of satellites
COLOR   Color of the female crab
        (1 = light medium, 2 = medium, 3 = dark medium,
        4 = dark)
SPINE   Spine condition
        (1 = both good, 2 = one work or broken,
        3 = both worn or broken)
WIDTH   Carapace width of the female crab (cm)
WEIGHT  Weight of the female crab (g)
```

The GLM output of this model is:

```
Call:
glm(formula = satell ~ color + spine + width + weight, family = "poisson",
    data = d)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0126  -1.8846  -0.5406   0.9448   4.9602

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.3435447  0.9684204  -0.355  0.72278
color       -0.1849325  0.0665236  -2.780  0.00544 **
spine        0.0399764  0.0568062   0.704  0.48160
width        0.0275251  0.0479425   0.574  0.56588
weight       0.0004725  0.0001649   2.865  0.00417 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 632.79  on 172  degrees of freedom
Residual deviance: 551.85  on 168  degrees of freedom
AIC: 917.15

Number of Fisher Scoring iterations: 6
```

Question 12.10

Comment on how the variables `color` and `spine` are coded here. Does this make sense in light of what those variables mean?

Question 12.11

Interpret the values of each of these coefficients. Based on the coefficient values and their standard errors, which predictor(s) do you think have the greatest impact on the number of male satellites around a nesting female horseshoe crab?

Question 12.12

How could you use a decision tree to model the horseshoe crabs data? What are its advantages and disadvantages relative to Poisson regression (a type of GLM)?

Chapter 13

Random Forests

In Chapter 7, we saw how individual decision trees are learned from training data. These days, decision trees are mainly used as parts of **ensembles**, collections of models whose predictions are combined to produce a final answer.

A **random forest** is an ensemble of decision trees whose predictions are [mostly] uncorrelated. Each tree is built using a subset of the training data and a subset of the features. The trees' predictions are then combined using a voting or weighting scheme. The same basic methodology works for several different supervised learning problems, including classification (Chapter 2), regression (Chapter 3), and survival analysis (Chapter 11). A key advantage of random forests is that they are non-parametric, meaning that they make no distributional or functional assumptions about the relationships between the predictors and the outcome. Another advantage is that the fitting of individual trees happens independently and can be **parallelized**.

Question 13.1

If random forests are so great, why are linear, logistic, and Cox proportional hazards regression models still the standard approaches to predicting continuous, categorical, and survival outcomes in clinical research? What do you think are some of the main drawbacks of random forests and other ensemble methods?

13.1 Building a Random Forest

Assume we have a training dataset of N samples and P predictors. The basic strategy for building a random forest is as follows[1]:

1. For $b = 1, \ldots, B$, where B is the desired number of trees:
   (a) Draw a bootstrap sample of size n from the training data.
   (b) Grow a tree, T_b, on the bootstrap sample by recursively repeating the following steps for each terminal node of the tree until the minimum node size, n_{min}, is reached:
      i. Select p predictors at random.
      ii. Pick the best predictor and split point.
      iii. Split the node into two child nodes.
2. Output the ensemble of trees, $T_1, \ldots, T_B$.

Question 13.2

This algorithm leaves us with many choices. We must choose B, the number of trees; n, the size of each bootstrap sample; p, the number of predictors evaluated for each split; and n_{min}, the minimum node size. What impact does each parameter choice have on the properties of the forest, both in terms of the individual trees and the forest as a whole?

Question 13.3

What does the "best" predictor and split point refer to? How are these choices made for classification and regression models? Think back to our discussion in Chapter 7.

The term **bootstrapping** refers to random sampling with replacement. To create a bootstrap sample of size n from a larger dataset of size N, we simply select n items from among the N, one at a time, being careful to put each

[1] See *Elements of Statistical Learning*, Chapter 15, Algorithm 15.1.

item back between selections. Because we are sampling with replacement, a bootstrap sample will likely contain repeats; this is fine and expected.

The process of averaging model predictions built on different bootstrap samples is called bootstrap aggregating, or **bagging**. We will learn more about why bagging works in Chapter 15.

13.2 Classification Example: Breast Cancer Diagnosis

Let's revisit the classification example for which we built a single decision tree in Chapter 7. The Wisconsin Breast Cancer Dataset contains information about 30 different imaging features of fine needle aspirate (FNA) samples from breast masses in 569 study participants. Here are tabular representations of two trees from a 100-tree random forest built on this dataset:

node_id	left_child	right_child	split_variable	split_point	prediction
1	2	3	area_mean	694.10	
2	4	5	symmetry_worst	0.37	
3	6	7	texture_mean	14.09	
4	8	9	concave.points_mean	0.05	
5	10	11	radius_worst	14.84	
6	12	13	compactness_se	0.03	
7	14	15	concave.points_mean	0.05	
8	16	17	area_se	42.19	
9	18	19	smoothness_worst	0.13	
10	0	0		0.00	B
11	0	0		0.00	M
12	0	0		0.00	B
13	0	0		0.00	M
14	0	0		0.00	M
15	0	0		0.00	M
16	0	0		0.00	B
17	0	0		0.00	M
18	0	0		0.00	B
19	0	0		0.00	M

node_id	left_child	right_child	split_variable	split_point	prediction
1	2	3	perimeter_worst	106.10	
2	4	5	radius_se	0.63	
3	6	7	radius_mean	15.04	
4	8	9	compactness_worst	0.76	
5	10	11	smoothness_se	0.01	
6	12	13	smoothness_mean	0.09	
7	14	15	radius_worst	18.23	
8	16	17	concave.points_worst	0.18	
9	0	0		0.00	M
10	18	19	compactness_se	0.01	
11	0	0		0.00	B
12	0	0		0.00	B
13	0	0		0.00	M
14	0	0		0.00	M
15	0	0		0.00	M
16	0	0		0.00	B
17	0	0		0.00	M
18	0	0		0.00	M
19	0	0		0.00	B

Question 13.4

Draw the two classification trees from the Wisconsin Breast Cancer Dataset random forest that are represented by these tables. Note: if a variable's value is less than the split point at a particular node, the training sample goes to the left.

13.3 Regression Example: Insurance Costs

The following dataset comes from the book *Machine Learning with R*, by Brett Lantz. It's unclear whether it is real or simulated, but it provides insurance cost information on 1338 subjects, as well as information about the following predictors:

(a) age (age of primary beneficiary)

(b) sex (sex of primary beneficiary, labeled "female" or "male")

(c) bmi (body mass index of beneficiary)

(d) children (number of children/dependents covered by beneficiary's health insurance)

(e) smoker (smoking status of beneficiary)

(f) region (the beneficiary's residential area in the U.S.: northeast, southeast, southwest, northwest)

The variable *charges* is the outcome of interest; it is the total individual medical costs (in thousands of dollars) billed by the beneficiary's health insurance. Here are tabular representations of two trees from a 100-tree random forest built on this dataset:

node_id	left_child	right_child	split_variable	split_point	prediction
1	2	3	smoker	yes	13.19
2	4	5	region	NE	8.65
3	6	7	region	NE, SE, SW	31.94
4	8	9	children	2.50	7.85
5	10	11	children	0.50	8.92
6	12	13	bmi	30.01	29.21
7	14	15	bmi	30.30	36.26
8	16	17	bmi	25.19	7.29
9	18	19	children	3.50	11.61
10	0	0		0.00	8.11
11	0	0		0.00	9.55
12	0	0		0.00	20.72
13	0	0		0.00	41.93
14	0	0		0.00	23.15
15	0	0		0.00	44.51
16	0	0		0.00	10.95
17	0	0		0.00	6.82
18	0	0		0.00	12.49
19	0	0		0.00	8.09

node_id	left_child	right_child	split_variable	split_point	prediction
1	2	3	smoker	yes	13.24
2	4	5	bmi	31.30	8.33
3	6	7	region	NE, NW	31.71
4	8	9	sex	2.00	7.56
5	10	11	region	NE, NW, SE	9.24
6	12	13	children	2.50	29.45
7	14	15	bmi	30.10	33.69
8	16	17	children	0.50	7.27
9	18	19	children	1.50	7.82
10	0	0		0.00	8.69
11	0	0		0.00	11.09
12	0	0		0.00	28.39
13	0	0		0.00	33.71
14	0	0		0.00	22.11
15	0	0		0.00	41.00
16	0	0		0.00	6.79
17	0	0		0.00	7.72
18	0	0		0.00	7.34
19	0	0		0.00	9.04

Question 13.5

Draw the two regression trees from the Insurance Cost Dataset random forest that are represented by these tables. Note: if a variable's value is less than the split point at a particular node, the training sample goes to the left. For categorical variables, if the variable's value is one of the categories listed under "split point", the training sample goes to the right.

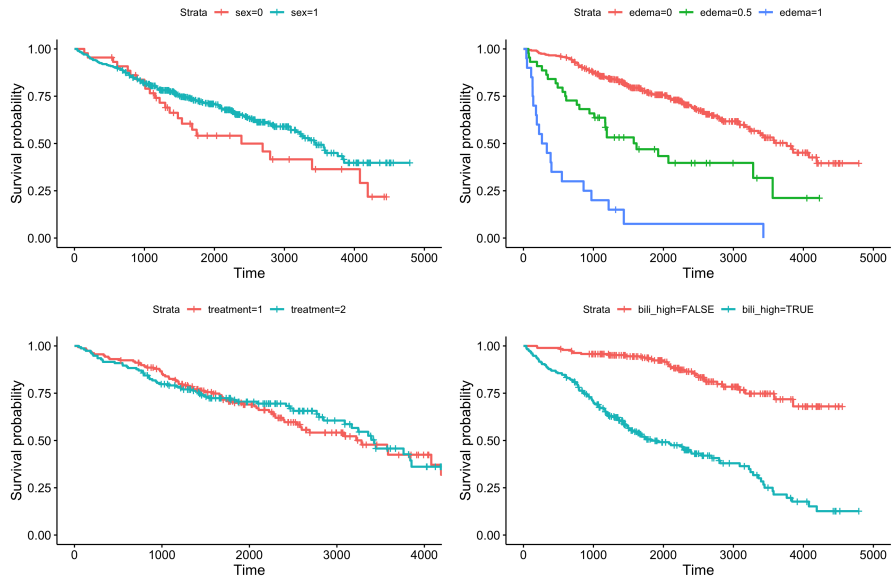
13.4 Model Parameters

13.4.1 Splitting Criteria

We already discussed how individual trees are built in Chapter 7. One of the choices we made then was which **splitting criterion** to use in building the tree. The splitting criterion is some way of deciding which variables are "good" to split on. For classification trees, the most common criteria are Gini index and information gain (the Gini index is by far the most popular). For regression, **variance reduction** (the same idea as standard deviation reduction) is the most common criterion.

Question 13.6

The following data come from a Mayo Clinic trial of the drug D-penicillamine for primary biliary cirrhosis (PBC) of the liver. The trial was conducted between 1974 and 1984. The data shown here are for 418 patients who completed the trial. The dataset comes from the `survival` package in R and contains information on 17 predictors, as well as the follow-up time and outcome (death or censoring) for each patient. Here are Kaplan-Meier curves (see Chapter 11) for four predictors, one of which (bilirubin) I manually binarized. Bilirubin is considered high if it is greater than 1.2 mg/dL.



Say you wanted to build a decision tree to predict survival in PBC. You would want to choose splits for which survival looks very *different* on either side of the split. This is analogous to choosing splits that increase the purity of the outcome (for classification) or reduce the variance of the outcome (regression). Speculate on how you might build such a tree. We will discuss the process of constructing **random survival forests** in much greater detail after we've seen a bit more survival analysis.

13.4.2 Creating Split Points

Each node within a tree signifies a division of one of the predictors into two groups. **Deterministic splitting** means considering all possible splits and identifying the best one. For a numeric predictor, this involves considering all of the values of the predictor represented in the dataset; there may be as many as n possible values, where n is the number of training samples included in the tree. For a categorical predictor, this involves dividing the possible categories into two **complementary groups**. For example, the insurance cost dataset in Section 13.3 contains a predictor `region` with four categories: northeast, southeast, southwest, and northwest. Each split decision must consider all

$2^{k-1} - 1$ possible divisions[2] of those categories, where k is the number of categories. For `region`, the possible splits are:

Group 1	Group 2
NE, SE, SW	NW
NE, SW, NW	SE
NE, SE, NW	SW
SE, SW, NW	NE
NE, SE	NW, SW
NE, SW	NW, SE
NE, NW	SE, SW

Deterministic splitting becomes problematic when the number of possible splits is large. In that case, software packages often employ **random splitting**, in which a predetermined number of possible splits (the exact number is set using a parameter) are randomly chosen from among all the possibilities.

Question 13.7

A known issue with decision trees is their tendency to prefer to split on continuous predictors over discrete predictors. Why do you think this is? It can be avoided, in part, by using random splitting with a fixed number of possible splits.

13.4.3 Bag Size and Number of Predictors

Each tree in a random forest is built using a "bagged" sample of n training examples from the original N examples. Typically around $2/3$ of training examples are used per tree, but most software packages include a parameter that allows the user to set this value. In addition, at each split, a tree considers only a randomly-chosen subset of p predictor variables; the default number is usually $\sqrt{P}$ (for classification problems) or $P/3$ (for regression problems), where P is the total number of predictors. In software, this will also be a

[2]This comes from taking 2^k (total combinations of k categories), subtracting 2 (all in or all out, neither of which is possible), and then dividing the whole thing by 2 (because the ordering of the groups doesn't matter). $(2^k - 2)/2 = 2^{k-1} - 1$

settable parameter. Usually it's fine to leave these parameters at their default values.

Question 13.8

For an ensemble to be more accurate than any of its individual members, the learners comprising the ensemble must be *accurate* and *diverse*. Accuracy means that the learners must perform better than random on their designated task. Diversity means that the classifiers must make different errors on new data points. How do the parameters n and p impact diversity? How does the way the trees are trained ensure accuracy?

13.4.4 Node Size and Tree Depth

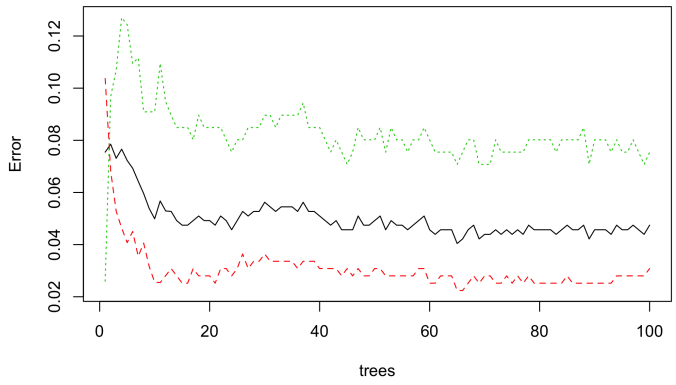
Software packages generally allow the user to control the growth of individual trees within a random forest by specifying node size and tree depth parameters. The **node size** parameter governs the minimum number of training samples present at a node for a split to be considered. The **tree depth** parameter governs the maximum number of connections between the root of the tree and one of its leaves. A split at a particular node will only be considered when there is still some impurity in the outcome at that node (see Chapter 7) and when:

1. The current tree depth is less than the maximum allowed tree depth.
2. The number of samples at a node is at least 2x the minimum node size (since a binary split on a smaller node would result in leaves with less than the minimum required node size).

13.5 The Out-of-Bag Error

The random forest will report a number called the **out-of-bag (OOB) error** as it runs. To calculate OOB error, each tree makes a prediction for each of the training samples *not* used in its construction. This provides an ongoing estimate of the generalization error of the forest.

Here is an OOB error plot for the random classification forest built on the Wisconsin Breast Cancer dataset:



The black line shows the overall OOB error (the percent of OOB points misclassified) after the addition of each new tree. The red line shows the OOB error for points of class 1, which in this case is *B* (benign), and the green line shows the OOB error for points of class 2, which in this case is *M* (malignant).

Question 13.9

What does it mean that the green line is so much higher than the red line? What does this tell you about the relative rates of false positives and false negatives for this random forest?

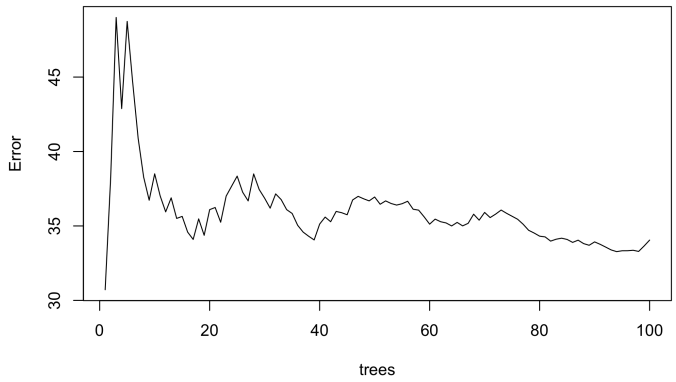
Question 13.10

Here is the final **confusion matrix** for the Wisconsin Breast Cancer random forest.

```
     B    M  class.error
B  346   11   0.03081232
M   16  196   0.07547170
```

It is probably more important to avoid false negatives (*M* tumors that are classified as *B*) than false positives. Speculate on ways in which you could force the forest to produce a lower rate of false negatives, even if it means increasing the number of false positives.

For regression forests, the OOB error is calculated differently. It is defined as the mean square error among the OOB samples. The square root of this error is the average absolute value of the difference between the predicted and actual costs.



13.6 Variable Importance Measures

One of the main disadvantages of random forests is their lack of clarity around which variables are "important". In a regression model, the model output contains hypothesis tests and coefficients for each variable that provide the user with an interpretable importance ranking. Nothing this simple exists for random forests. However, there are some heuristics for ranking variables. These fall into two camps.

13.6.1 Impurity-Based Importance

Trees are built by choosing splits that reduce uncertainty, or impurity, in the outcome. This impurity reduction is a measure of how much splitting on that variable "helps" in purifying the outcome. One way to measure the importance of a variable, therefore, is to average the decrease in node impurity across all splits involving that variable, across all trees in the random forest[3]. This importance measure is called the **Mean Decrease in Impurity (MDI)**. It

[3]Because splits occur at different heights, impurity reduction is typically weighted by how many samples reach a given node.

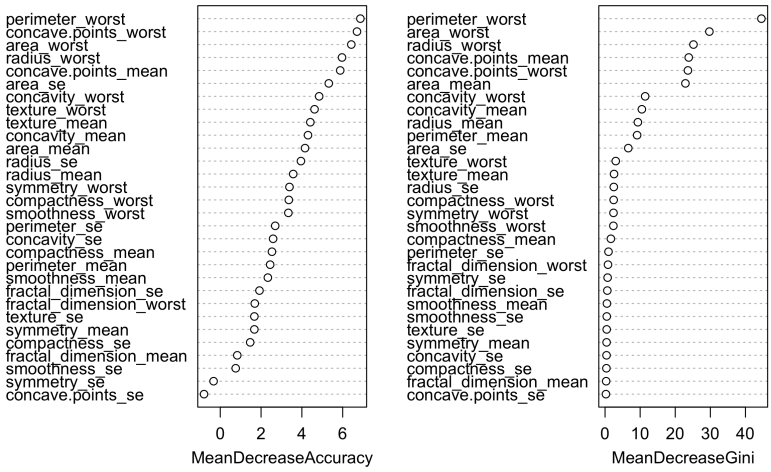
works no matter what your outcome is. Its main advantage is that because the reduction in impurity is what is already used to determine the splits, it requires very little additional computation.

13.6.2 Permutation-Based Importance

An alternative way of measuring the importance of variable j is to see how much it affects the predictive accuracy of trees across the whole forest. We assess this using the OOB samples. First the real OOB error is calculated (by running each sample through the trees for which it is OOB). Then the values of variable j across the OOB samples are randomly permuted, and the OOB error is calculated again. We expect the OOB error to go up in the second case by an amount proportional to how important variable j is. We then average this difference for each variable across all trees. This permutation-based importance measure is called the **Mean Decrease in Accuracy (MDA)**. Again, it works no matter what the outcome is. Its main advantage is that it is more interpretable than MDI.

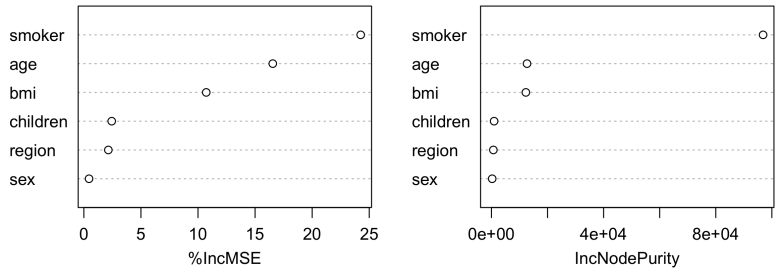
Question 13.11

Here is a variable importance plot for the Wisconsin Breast Cancer classification forest. Which side is MDI and which is MDA? Which variables are most and least important?



Question 13.12

Here is a variable importance plot for the insurance cost dataset regression forest. Which side is MDI and which is MDA? Which variables are most and least important?



13.7 A Note on Software Packages

As we have seen in Chapter 7, there are many different ways to build and optimize decision trees. There are even more ways to build and optimize random forests. This chapter uses the `randomForest` R package by Andy Liaw, a faithful implementation of the original random forest implementation suggested by Breiman (2003), as well as `randomForestSRC`, a faster and more recent package by Ishwaran and Kogalur that provides a unified interface for random forest-based classification, regression, and survival analysis. The `scikit-learn` package in Python provides implementations of random forests for both classification and regression.

Chapter 14

Introduction to Boosting

Random forests (Chapter 13) are one approach to ensemble learning. Today we will examine another approach, **boosting**, that relies on a completely different set of ideas. The first practical boosting algorithm, AdaBoost, was invented in 1995 by Freund and Schapire. However, boosting is a general approach to choosing a set of weak learners that, together, create a strong learner. Since the same idea has spawned many different approaches, boosting is referred to as a **meta-algorithm**.

14.1 AdaBoost

Assume we are trying to solve a supervised learning problem. As usual, our data look like this:

$$\{(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})\}$$

where the $x^{(i)}$ are feature vectors of length p. For now, we will assume that the outcome, Y, is binary (so this is a binary classification problem – see Chapter 2). We'll make one small notational change from earlier chapters. Instead of $Y \in \{0, 1\}$, we'll say $Y \in \{-1, 1\}$. The fact that the positive and negative training examples have opposite sign will help us write the algorithm more concisely.

The basic strategy behind AdaBoost is to build classifiers in series while

maintaining a set of weights for the different training examples. The weights change with each subsequent classifier, increasing if previous classifiers have made incorrect predictions and decreasing if previous classifiers were correct. In this way, the importance of difficult-to-classify training examples increases, leading us to prefer future classifiers that successfully predict these examples.

Here is the algorithm:

1. Initialize the observation weights to $w_i^{(1)} = \frac{1}{N}$ for $i = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Select a classifier, $G_m(x)$, that minimizes the weighted training error according to the current set of weights, $w_i^{(m)}$. Depending on the algorithm, it may be possible to train a single classifier on the weighted training set; in other cases, one may need to select the best-performing classifier from among a predefined set.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)} \cdot \mathcal{I}(y^{(i)} \neq G_m(x^{(i)}))}{\sum_{i=1}^{N} w_i^{(m)}}$$
   (c) Compute voting weight for classifier m:
   $$\alpha_m = \log\left(\frac{1 - \text{err}_m}{\text{err}_m}\right)$$
   (d) Set
   $$w_i^{(m+1)} := w_i^{(m)} \cdot \exp\left[\alpha_m \cdot \mathcal{I}(y^{(i)} \neq G_m(x^{(i)}))\right]$$
   for $i = 1, \ldots, N$.
3. Output
   $$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

The output definition in step 3 is really the key to the whole thing. Your goal is to construct a set of classifiers whose votes will be added together to produce an overall decision. AdaBoost provides one possible set of instructions for how to choose/build the individual classifiers, $G_m(x)$, and how to weight their votes.

14.2 Visualizing the Steps of AdaBoost

We will now train AdaBoost on the happiness dataset from Section 7.3. Here is the dataset, using the new notation for Y. The astute observer will notice that we've added a fourth covariate, X_4 (whether or not the person has a pet). The reason is that subjects 5 and 10 are exactly the same otherwise, so we can never get perfect separation using the original dataset.

Subject ID	friends (X_1)	money (X_2)	free time (X_3)	pet (X_4)	happy (Y)
1	1	1	0	0	-1
2	1	1	1	0	-1
3	0	1	1	0	-1
4	0	0	0	0	-1
5	1	0	0	0	-1
6	0	0	0	0	-1
7	1	2	1	0	1
8	1	0	1	0	1
9	0	0	1	1	1
10	1	0	0	1	1

$$X_1 = \begin{cases} 0 & \text{no friends} \\ 1 & \text{friends} \end{cases} \qquad X_2 = \begin{cases} 0 & \text{poor} \\ 1 & \text{enough money} \\ 2 & \text{rich} \end{cases}$$

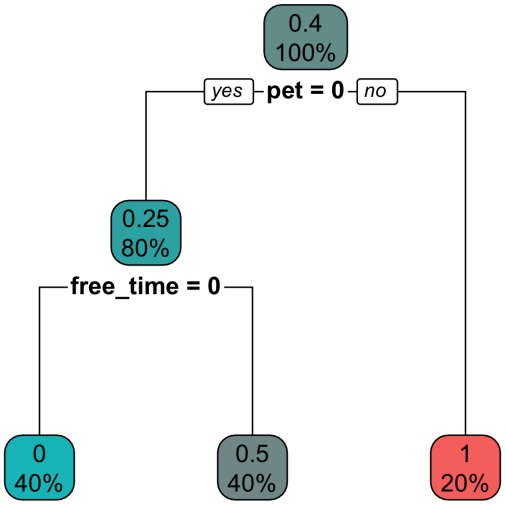
$$X_3 = \begin{cases} 0 & \text{no free time} \\ 1 & \text{some free time} \end{cases} \qquad X_4 = \begin{cases} 0 & \text{no pet} \\ 1 & \text{has a pet} \end{cases}$$

Now, let's go through the process of applying AdaBoost to this dataset, step by step.

(a) Initialize the observation weights for the training data. (Make them uniform.)

$w_1^{(1)} =$ $w_6^{(1)} =$
$w_2^{(1)} =$ $w_7^{(1)} =$
$w_3^{(1)} =$ $w_8^{(1)} =$
$w_4^{(1)} =$ $w_9^{(1)} =$
$w_5^{(1)} =$ $w_{10}^{(1)} =$

(b) We will now grow a decision tree on this dataset, $G_1(x)$, using these weights. We'll use the `rpart` package in R, just as in Section 7.3. Here is the first tree.



Here are the predictions for this tree, along with the current weights.

Datapoint ID	$w_i^{(1)}$	happy (Y)	$G_1(x)$
1	0.1	-1	-1
2	0.1	-1	-1
3	0.1	-1	-1
4	0.1	-1	-1
5	0.1	-1	-1
6	0.1	-1	-1
7	0.1	1	-1
8	0.1	1	-1
9	0.1	1	1
10	0.1	1	1

Compute the misclassification error of this tree, err_1. Compare this tree to the one we constructed by hand in Chapter 7.

$$\text{err}_1 =$$

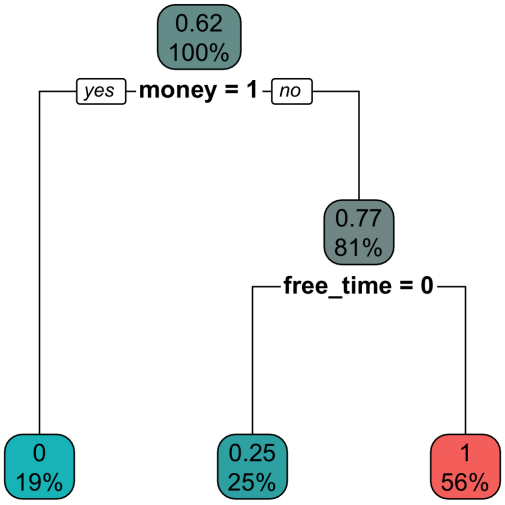
(c) Based on how $G_1(x)$ performs, calculate α_1, its voting weight.

$$\alpha_1 =$$

(d) Re-weight the observation weights for the training examples.

$$w_1^{(2)} = \qquad w_6^{(2)} =$$
$$w_2^{(2)} = \qquad w_7^{(2)} =$$
$$w_3^{(2)} = \qquad w_8^{(2)} =$$
$$w_4^{(2)} = \qquad w_9^{(2)} =$$
$$w_5^{(2)} = \qquad w_{10}^{(2)} =$$

(e) Now we grow another decision tree, $G_2(x)$, using these new weights as inputs to the `rpart` package.



Here are the predictions for this tree, along with the current weights.

Datapoint ID	$w_i^{(2)}$	happy (Y)	$G_2(x)$
1	0.1	-1	-1
2	0.1	-1	-1
3	0.1	-1	-1
4	0.1	-1	-1
5	0.1	-1	-1
6	0.1	-1	-1
7	0.4	1	1
8	0.4	1	1
9	0.1	1	1
10	0.1	1	-1

Compute the misclassification error of this tree, err_2.

$$\text{err}_2 =$$

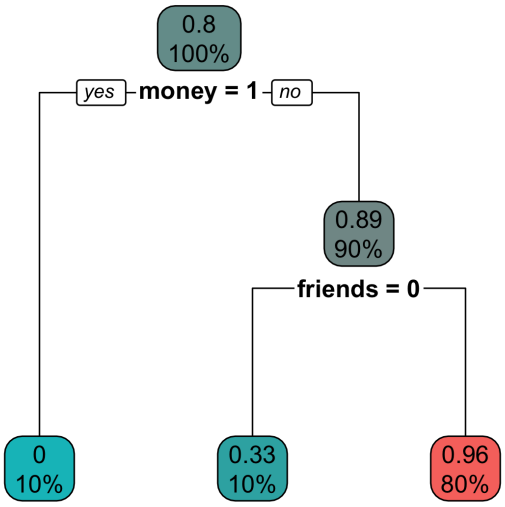
(f) Based on how $G_2(x)$ performs, calculate α_2, its voting weight.

$$\alpha_2 =$$

(g) Re-weight the observation weights for the training examples.

$$\begin{aligned} w_1^{(3)} &= & w_6^{(3)} &= \\ w_2^{(3)} &= & w_7^{(3)} &= \\ w_3^{(3)} &= & w_8^{(3)} &= \\ w_4^{(3)} &= & w_9^{(3)} &= \\ w_5^{(3)} &= & w_{10}^{(3)} &= \end{aligned}$$

(h) Now we grow another decision tree, $G_3(x)$, using these new weights as inputs to the `rpart` package.



Here are the predictions for this tree, along with the current weights.

Datapoint ID	$w_i^{(3)}$	happy (Y)	$G_3(x)$
1	0.1	-1	-1
2	0.1	-1	-1
3	0.1	-1	-1
4	0.1	-1	-1
5	0.1	-1	1
6	0.1	-1	-1
7	0.4	1	1
8	0.4	1	1
9	0.1	1	-1
10	1.5	1	1

(i) Compute the misclassification error of this tree, err_3.

$$\text{err}_3 =$$

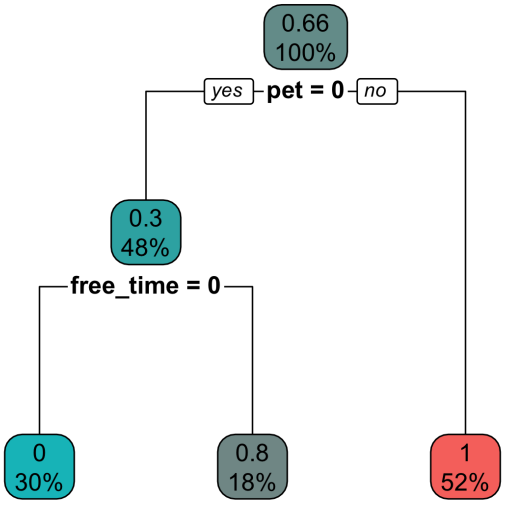
(j) Based on how $G_3(x)$ performs, calculate α_3, its voting weight.

$$\alpha_3 =$$

(k) Re-weight the observation weights for the training examples.

$$w_1^{(4)} = \qquad w_6^{(4)} =$$
$$w_2^{(4)} = \qquad w_7^{(4)} =$$
$$w_3^{(4)} = \qquad w_8^{(4)} =$$
$$w_4^{(4)} = \qquad w_9^{(4)} =$$
$$w_5^{(4)} = \qquad w_{10}^{(4)} =$$

(l) Now we grow another decision tree, $G_4(x)$, using these new weights as inputs to the `rpart` package.



Here are the predictions for this tree, along with the current weights.

Datapoint ID	$w_i^{(4)}$	happy (Y)	$G_4(x)$
1	0.1	-1	-1
2	0.1	-1	1
3	0.1	-1	1
4	0.1	-1	-1
5	1.4	-1	-1
6	0.1	-1	-1
7	0.4	1	1
8	0.4	1	1
9	1.4	1	1
10	1.5	1	1

(m) Compute the misclassification error of this tree, err_4.

$$\text{err}_4 =$$

(n) Based on how $G_4(x)$ performs, calculate α_4, its voting weight.

$$\alpha_4 =$$

(o) Output the weighted average of the four classifiers' votes for each training example:

$$G(x) = \text{sign}\left[\alpha_1 G_1(x) + \alpha_2 G_2(x) + \alpha_3 G_3(x) + \alpha_4 G_4(x)\right]$$

What is the final training error?

Datapoint ID	happy (Y)	$G_1(x)$	$G_2(x)$	$G_3(x)$	$G_4(x)$	$G(x)$
1	-1	-1	-1	-1	-1	
2	-1	-1	-1	-1	1	
3	-1	-1	-1	-1	1	
4	-1	-1	-1	-1	-1	
5	-1	-1	-1	1	-1	
6	-1	-1	-1	-1	-1	
7	1	-1	1	1	1	
8	1	-1	1	1	1	
9	1	1	1	-1	1	
10	1	1	-1	1	1	

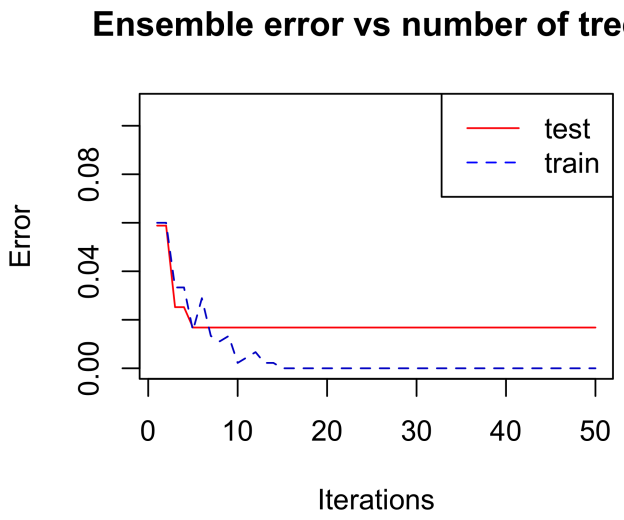
Question 14.1

The most difficult thing to understand in all of this is how the updated weights play into the construction of subsequent trees. A clue comes from the dataset percentages shown in the nodes of each tree. For example, trees 1 and 4 actually have the same structure, but the impurities at each node are different due to the different weights. Discuss how the weights could inform the variables the splitting algorithm chooses to split on (revisit Chapter 7, if necessary, to see the math).

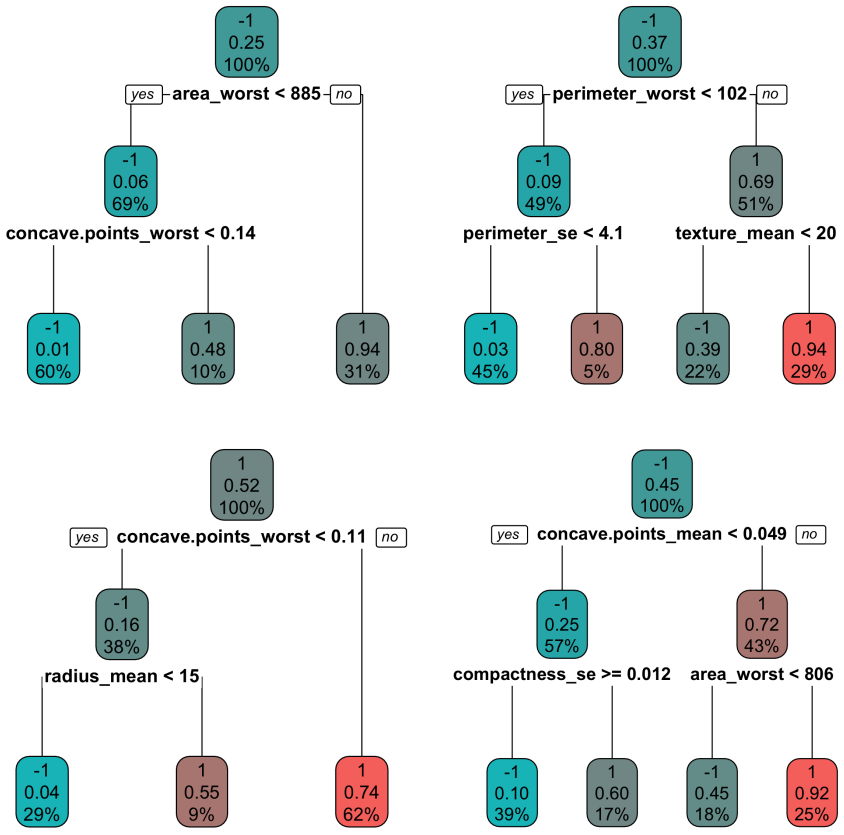
14.3 Revisiting Breast Cancer Classification

Let's once again revisit the classification example for which we built a single decision tree in Chapter 7 and a random forest in Chapter 13. The Wisconsin Breast Cancer Dataset contains information about 30 different imaging features of fine needle aspirate (FNA) samples from breast masses in 569 study participants.

Here is a plot showing the evolution of the training and test error as a function of the number of trees:



where 450 samples are used for training and the remaining 119 for testing. Here are tree numbers 1, 3, 6, and 8:



Question 14.2

Compare and contrast boosting and random forests on the basis of:

- Whether each tree uses all or part of the dataset
- Whether they consider a subset of the predictors at each split or all the predictors
- Whether you can parallelize the construction of different trees (i.e., build them at the same time on different processors)
- Whether the votes of different classifiers are independent
- Ease of use and interpretability

Chapter 15

Model Quality and the Bias-Variance Tradeoff

We have now seen several different algorithms (and meta-algorithms) for supervised learning. We've examined linear models for classification and regression (Chapters 8 and 9), KNN (Chapters 2 and 3), decision trees (Chapters 7), and ensemble methods – usually based on decision trees – including random forests (Chapter 13) and boosting (Chapter 14). In this short chapter, we'll take a step back and talk about some key themes that are relevant to all of these methods.

15.1 Measuring Error: Loss and Objective Functions

Most of us think of error as a fairly intuitive concept, and so far in our discussions of model building and model quality, we've avoided any formal definitions. However, as we begin to extend the concepts we've learned from classification and regression to more challenging problem classes (e.g., survival analysis), a more formal conception of what error means and how to minimize it algorithmically becomes increasingly crucial. Here are some examples:

- In classification, error typically means either the total number of mis-

classified points,

$$\text{error} = \sum_{i=1}^{n} \mathcal{I}\left(y^{(i)} \neq \hat{y}^{(i)}\right),$$

or a weighted average of the number of misclassified points per class. Another way of quantifying classification error is **AUC**, the area under the receiver operating characteristic curve, which is the probability that a randomly chosen positive example (where $y = 1$) will receive a higher score from the model (higher $\hat{y}$) than a randomly chosen negative example (where $y = 0$).

- In regression, error typically means the **mean-squared error (MSE)**,

$$\text{error} = \frac{1}{n} \sum_{i=1}^{n} \left(y^{(i)} - \hat{y}^{(i)}\right)^2,$$

although there are also other ways of quantifying error. For example, you could use the **mean absolute error**

$$\text{error} = \frac{1}{n} \sum_{i=1}^{n} |y^{(i)} - \hat{y}^{(i)}|.$$

Neither is "wrong" or "right", but they have different properties. For example, the MSE gives a higher weight to large errors (outliers). It's also differentiable, which is why it's used so much more often in, e.g., neural networks than the mean absolute error.

- In survival analysis, error is typically quantified using something called **Harrell's concordance index**, which takes into account censored observations[1].

Question 15.1

Harrell's concordance (C) index is probably unfamiliar to you. It is calculated like this:

[1]For a detailed explanation and some experimental results, see Kattan MW, Hess KR and Beck JR (1998). "Experiments to determine whether recursive partitioning (CART) or an artificial neural network overcomes theoretical limitations of Cox proportional hazards regression." *Computers and Biomedical Research*, 31(5), 363-373.

1. Create a list of all possible pairs of patients. There will be $n(n-1)/2$ pairs.
2. Eliminate all pairs for which the patient with the shorter follow-up time does not experience the event of interest (i.e., is censored). The remaining patient pairs are considered "usable" since the patient with the shorter time-to-event is identifiable.
3. Count the number of usable patient pairs for which the patient with the shorter follow-up time had the higher predicted hazard for the event. That is, you want the number of pairs for which the model's predictions about relative times to event are consistent with the observed data.
4. The *C* statistic is the number of consistent pairs divided by the number of usable pairs. The error is defined as $1 - C$.

Here are four patients and the predicted mean time to event from two different survival models (low value means lower time to event). All times are in years. Calculate the *C* statistic for both models.

Patient	Follow-up Time	Observed?	Model 1 Score	Model 2 Score
1	8.3	1	4.6	5.2
2	6.5	0	2.3	7.1
3	2.7	1	0.6	6.7
4	7.4	1	4.7	6.6

First Patient	Second Patient	Usable	Model 1 Consistent	Model 2 Consistent
1	2			
1	3			
1	4			
2	3			
2	4			
3	4			

You should find that the *C* statistic for Model 1 is 0.75, while the *C* statistic for Model 2 is only 0.25. Model 1 is clearly the better model.

The definition of error used to optimize a particular model is called a **loss function**. Loss functions are a general concept from optimization and

decision theory; they represent the cost associated with a decision. If we think of a supervised learning model as an engine for making decisions, the loss is how "bad", on average, those decisions will be. Loss functions are, in turn, part of a broader class of functions called **objective functions**. Most learning algorithms work by either minimizing some measure of "badness" (a loss function) or maximizing some measure of "goodness" (negative of the loss, alternatively called a **reward function**).

Question 15.2

Imagine what would happen if, in Question 15.1, we simply set a follow-up time of 7 years and treated the problem as a classification problem, calculating the AUC for the two models based on that time horizon and ignoring censoring.

Positive Patient	Negative Patient	Model 1 Ranks Pos Higher?	Model 2 Ranks Pos Higher?
3	1	1	0
3	2	1	1
3	4	1	1

We would calculate an AUC of 1.0 for Model 1 and 0.67 for Model 2. Do you think this is a good approach to quantifying error for this problem? Why or why not?

The choice of how to define error, and which loss function to use for quantifying that error, ultimately rests with the model builder. The process of model training is about adjusting the available parameters of the model to minimize the loss.

15.2 Goodness of Fit vs. Generalizability

Once we have an appropriate objective function, we can set about building a model to optimize it. This requires us to think about what constitutes a "good model". It's a bit more complicated than simply minimizing the loss on our training data, because ideally we want a model that is both accurate

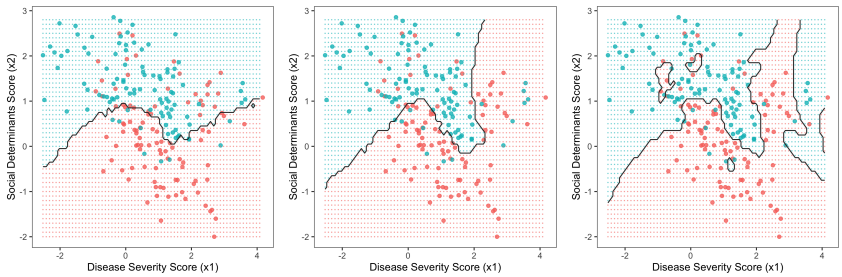
and **parsimonious**, meaning that it is as simple as possible without sacrificing performance. Another way of thinking about this is that we create models to tell stories about the data we see. If they are good stories, they will do two things:

1. Explain the structure of the data that are used to train them (high **goodness of fit**)
2. Make accurate predictions on new data (good **generalizability**)

For a supervised learning model, we quantify (1) using the **training error** (error on the training set) and (2) using the **test error** (error on an independent test set).

Question 15.3

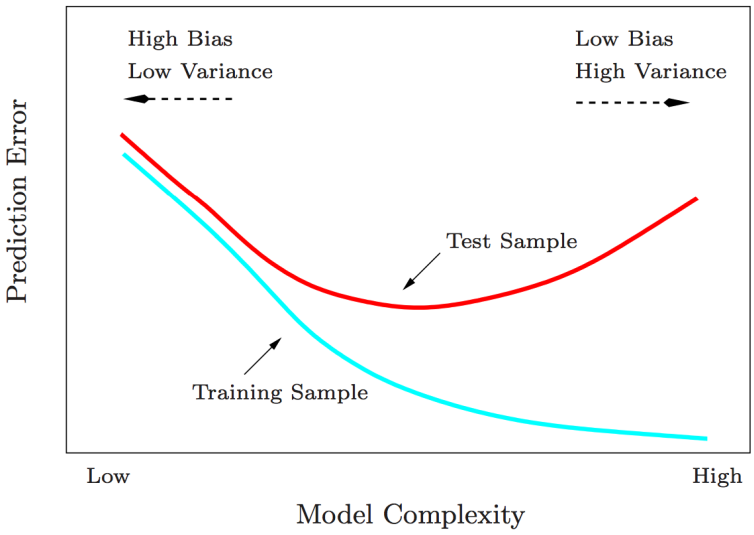
Here we see three decision boundaries for KNN with different values of K (the number of neighbors considered in making a prediction). The data are for the two-class classification problem first discussed in Chapter 2. From left to right, $K = 50, 15$, and 3. What are the tradeoffs in moving from left to right in terms of (a) training error/goodness of fit and (b) test error/generalizability?



In supervised learning, **model complexity** (loosely defined as the effective number of parameters the model must fit) is, therefore, a very important consideration. A model that is not complex enough will fail to capture all of the structure in the training data, and both its training and test error will suffer as a result. We call this situation **underfitting**. Conversely, a model that is too complex may fit the training data very well – maybe perfectly – but will fail to generalize well to new data. We call this situation **overfitting**.

15.3 Bias vs. Variance

It turns out that there is a general principle governing model complexity in supervised learning called the **bias-variance tradeoff**. It is perhaps best illustrated by this figure, which comes from the excellent (and free) book *Elements of Statistical Learning*, by Tibshirani, Hastie, and Friedman (Figure 2.11):



The figure shows us that test error, our measurement of generalization error, comes from two different sources:

$$\text{test error} = \text{bias} + \text{variance}.$$

The term **bias** refers to error that results from underfitting, while **variance** results from overfitting.

Error due to bias can often be reduced by introducing more/different features or by using a different learning algorithm. Error due to variance can often be reduced by increasing the size or diversity of the training data. That's why it's important to know which situation you're in; gathering more training data when your model is too simple is unlikely to help you, and deploying an extremely fancy (and complicated) deep learning model when

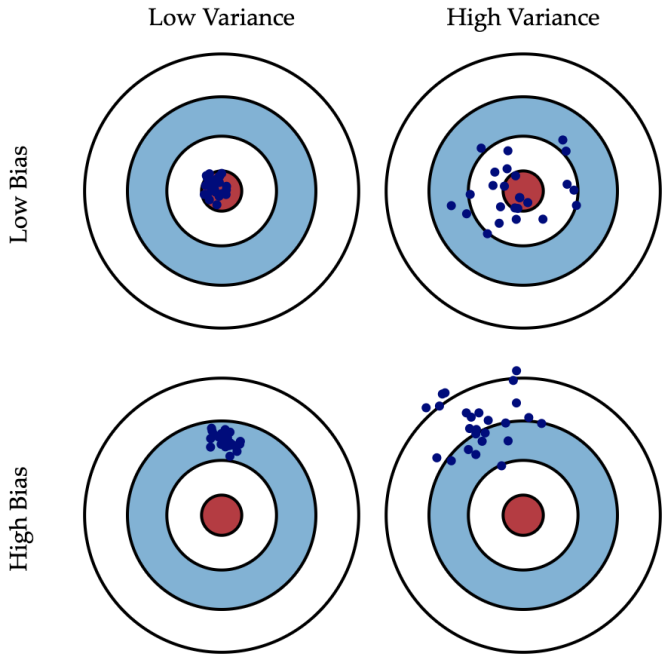
logistic regression has already overfit your training data is also unlikely to help.

Another consideration is that even the "sweet spot" (minimum test error) on the bias-variance tradeoff curve can still be high. This occurs when the data you have available simply aren't sufficient to answer the question – maybe you have the wrong features or there is no relationship between the features and the outcome. Or maybe your features aren't measured correctly. In my experience, these types of issues are not considered often enough in the clinical domain.

Question 15.4

A recent review article had the provocative title "A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models." (Christodoulou E, Ma J, Collins GS, Steyerberg EW, Verbakel JY, Van Calster B. *Journal of Clinical Epidemiology*, 2019, 110:12-22). Assuming this finding is true, what is it telling us about the nature of the error in most clinical risk prediction models? What does it suggest about what we should be doing to improve the performance of these models?

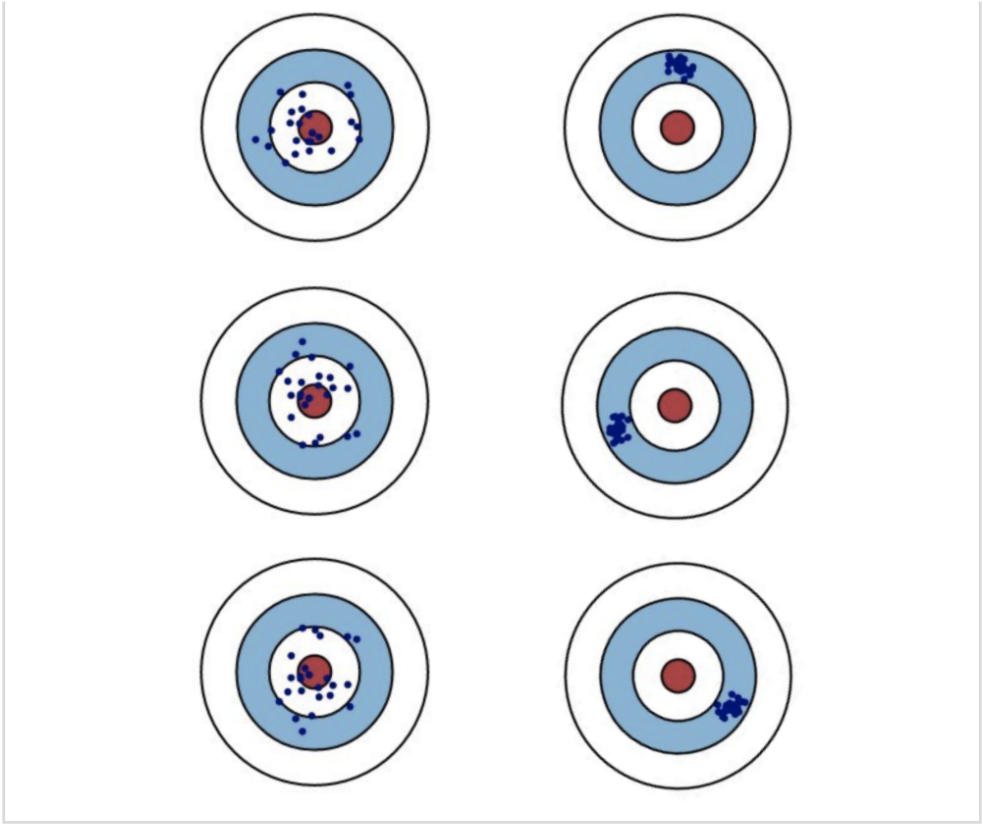
Another way of thinking about bias and variance is in terms of what happens when you make slight changes to the training data (for example, by collecting new data from the same patient population, or drawing repeated bootstrap samples from the same training set). The figure below is from the article *Understanding the Bias-Variance Tradeoff*, by Scott Fortmann-Roe:



Think of each dot as representing a single test example evaluated under the same model trained on slightly different datasets. The center of the target is the prediction the model should make for that test example. In the case of high bias and low variance, all of the models are off, but they are "wrong in the same way". Even if you average their predictions, the answer is still way off the mark. In the case of high variance, the models all make incorrect predictions, but their predictions are off in different directions. As a result, if you average their outputs, you'll get closer to the right answer.

Question 15.5

We saw random forests in Chapter 13 and were introduced to boosting in Chapter 14. It is interesting to compare the two methods because they reduce test error in different ways: one primarily tackles variance, while the other primarily tackles bias. Which one is which, and how does each algorithm succeed in reducing its respective form of error? Hint: The image below may help. One column represents three trees from a random forest; the other represents three trees from a boosting model.



Chapter 16

Feature Selection

Modern clinical datasets tend to suffer from an overabundance of features. In fact, there are often more available features than there are training examples. Not all of these features will contribute equal information about the outcome. Including dozens or hundreds of predictors in a supervised learning model does not guarantee higher accuracy; in fact, it is more likely to lead to models that are unnecessarily complex and overfit (see Chapter 15). Even when features are related to the outcome, they may contribute information that is redundant with other features in the study.

In cases like these, the model designer will either need to choose a subset of features manually or incorporate some form of **feature selection**: a process that automatically or semi-automatically decides which features are most relevant to the model and discards the others. The goal of feature selection is to remove useless and redundant features in a principled way.

16.1 Example: The Pima Indians Dataset

The so-called "Pima Indians diabetes dataset" was collected in the 1980s. It includes information on 768 women from the Pima people, who live near Phoenix, Arizona. The Pima were, as of the late 1980s, under continuous study by the National Institute of Diabetes and Digestive and Kidney Diseases

because of their high incidence of diabetes[1]. There are eight predictors in the dataset and one outcome. The predictors are:

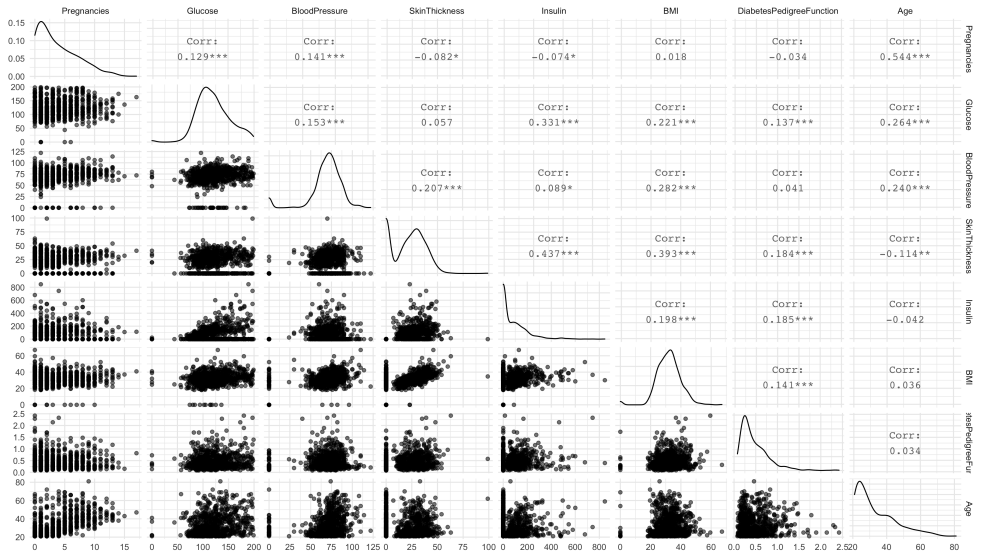
Predictor	Description
Pregnancies	Number of times pregnant
Glucose	Plasma glucose concentration in a two-hour oral glucose tolerance test
BloodPressure	Diastolic blood pressure (mm Hg)
SkinThickness	Triceps skin fold thickness (mm)
Insulin	Two-hour serum insulin (μU/mL)
BMI	Body mass index (weight in kg/(height in m)^2)
DiabetesPedigreeFunction	Diabetes pedigree function (developed by research team; described in paper)
Age	Age in years

The outcome is whether or not the woman went on to develop type II diabetes within 5 years from the time of the survey.

16.2 Correlograms

For datasets with a manageable number of features, A good way to alert oneself to the presence of highly correlated predictors is to create a **correlogram**, or scatterplot matrix, which looks at associations between all pairs of variables. A correlogram for the Pima dataset is below.

[1]The causative factors behind this high diabetes rate are not clear. Some scholars believe that it was driven by a sudden shift in diet during the last century from traditional agricultural crops to processed foods, together with a decline in physical activity L. O. Schulz, P. H. Bennett, E. Ravussin, J. R. Kidd, K. K. Kidd, J. Esparza, and M. E. Valencia. "Effects of traditional and western environments on prevalence of type 2 diabetes in Pima Indians in Mexico and the US". in: *Diabetes Care* 29.8 (2006), pp. 1866–1871.



Question 16.1

This correlogram quantifies correlation using a metric called the **Pearson correlation coefficient**. Which pairs of predictors are the most tightly correlated? Are they positively or negatively correlated? How might you modify your dataset to eliminate redundancies in the information contributed by the different predictors?

16.3 Univariate vs. Multivariate Models

The presence of correlations will affect different types of models in different ways, and some suffer more than others. In the clinical research literature, the standard approach to assessing and accounting for correlations is to start with **univariate** models, in which each predictor's association with the outcome is studied on its own. Those predictors that display some association with the outcome are then incorporated into larger **multivariate** models.

Here are the results of eight univariate logistic regression models that capture the effect of each predictor in the Pima dataset on the outcome of diabetes vs. no diabetes. The coefficients on each predictor are called the **unadjusted** coefficients, and the p-values on the predictor-specific hypothesis

tests are called unadjusted p-values. The exponentiated coefficients are called unadjusted odds ratios[2].

Predictor	Unadjusted Coefficient	Unadjusted Odds Ratio	Unadjusted P-value
Pregnancies	0.137	1.147	<0.001
Glucose	0.038	1.039	<0.001
BloodPressure	0.007	1.007	0.073
SkinThickness	0.010	1.010	0.039
Insulin	0.002	1.002	<0.001
BMI	0.094	1.100	<0.001
DiabetesPedigreeFunction	1.083	2.953	<0.001
Age	0.042	1.043	<0.001

Now, here is a multivariate logistic regression model that includes all eight predictors. The coefficients, exponentiated coefficients, and p-values are often called **adjusted**.

Predictor	Adjusted Coefficient	Adjusted Odds Ratio	Adjusted P-value
Pregnancies	0.123	1.131	<0.001
Glucose	0.035	1.036	<0.001
BloodPressure	-0.013	0.987	0.011
SkinThickness	0.001	1.001	0.929
Insulin	-0.001	0.999	0.186
BMI	0.090	1.094	<0.001
DiabetesPedigreeFunction	0.945	2.573	0.002
Age	0.015	1.015	0.111

Alternatively, one might say that the odds ratios here measure the effect of each predictor, **controlling for** the effects of the other predictors.

[2]See Chapter 9 if you don't understand why you're exponentiating or where the term "odds ratio" comes from. The odds ratio compares the odds of having a positive outcome among two groups separated by a one unit difference of the predictor in question, all else being the same.

Question 16.2

How can the odds ratio for Insulin be so close to 1.0 yet its p-value so low? (Hint: See Section 12.5.)

Question 16.3

Why might the coefficient and p-value for SkinThickness change so much in the shift from unadjusted to adjusted?

16.4 Filter Methods

The approach of creating univariate models and then incorporating the best-performing predictors into multivariate models is part of a broader class of feature selection methods called **filter methods**. Filter methods select subsets of variables as a preprocessing step, independently of the supervised learning model that will eventually be implemented. These methods use **proxy measures** to rank variables; the proxy measure is often chosen to be computationally fast so that large numbers of features can be sifted through quickly.

A predetermined threshold of the proxy measure is usually used to determine which features pass to the multivariate modeling stage. Alternatively, the modeler may decide on a fixed number of features to include. Some examples of filter methods include:

- Any kind of univariate model (e.g. univariate logistic or linear regression)
- Any kind of hypothesis test (e.g. t-test, chi-squared test; see Chapter 6)
- Any kind of correlation coefficient (e.g. Pearson, Spearman)

- Mutual information[3]

$$MI(X_i, Y) = \sum_x \sum_y P(X_i = x, Y = y) \log \frac{P(X_i = x, Y = y)}{P(X_i = x)P(Y = y)}$$

- Variance thresholding (simply remove features with low variance)

Question 16.4

If you wanted to use the univariate logistic regression models above in Section 16.3 as a filter for a downstream model (potentially not even multivariate logistic regression - it could be a decision tree, etc.), how would you rank them and how would you decide on an appropriate cutoff?

Question 16.5

How would you apply a filter-based selection method in a case where you had dozens of different predictors of different types (e.g. some categorical, some binary, some numeric)?

Question 16.6

How might you choose the appropriate threshold for a filter-based method in a data-driven way?

Question 16.7

What is problematic about testing each potential feature, one at a time?

16.5 Wrapper Methods

Filter methods are just one possible approach to feature selection. An alternative to filter methods are **wrapper methods**. These methods use a search algorithm to traverse the space of possible features, evaluating each subset

[3]The mutual information, in another format, is the most common splitting criterion used for decision trees; see Chapter 7. In the case of continuous variables, the sums are replaced by integrals.

by running the chosen model using that subset. They are generally computationally intensive (e.g., imagine trying to find the optimal subset of 10,000 features, or even 50) so **heuristics** generally have to be used to pare down the search space. Some examples of wrapper methods include:

- **Exhaustive search.** Try all possible subsets of features. If there are m features, this means trying 2^m possible subsets.
- **Forward selection.** Start with a baseline (e.g., intercept only) model. Add in each of m possible predictors individually and take the best one based on some performance criterion. Repeat, adding one predictor at each step, until the performance criterion stops getting better or you run out of predictors.
- **Backward elimination.** Start with a complete model (all predictors included). Try removing each predictor and take the one whose removal causes the performance criterion to increase the most. Repeat, removing one predictor at each step, until the performance criterion stops getting better or you are left with no predictors (null model).
- **Forward-backward selection.** A combination of forward selection and backward elimination.
- **Simulated annealing.** Add or remove predictors with some probability depending on how well the model is doing. At each stage, if the new model is better, accept it; it becomes the new baseline. If the new model is worse, accept it with some probability, p, that decreases over time according to a "cooling schedule". This helps prevent the variable selection process from getting stuck in local optima.

Question 16.8

Why is exhaustive search problematic for almost any reasonably sized m?

Question 16.9

Here is the output of forward selection for the Pima example, using R's *MASS* package and the **Akaike Information Criterion (AIC)** as the model performance metric.

```
Start:  AIC=995.48
Outcome ~ 1
                           Df Deviance    AIC
+ Glucose                   1   808.72 812.72
+ BMI                       1   920.71 924.71
+ Age                       1   950.72 954.72
+ Pregnancies               1   956.21 960.21
+ DiabetesPedigreeFunction  1   970.86 974.86
+ Insulin                   1   980.81 984.81
+ SkinThickness             1   989.19 993.19
+ BloodPressure             1   990.13 994.13
<none>                          993.48 995.48

Step:  AIC=812.72
Outcome ~ Glucose

                           Df Deviance    AIC
+ BMI                       1   771.40 777.40
+ Pregnancies               1   784.95 790.95
+ DiabetesPedigreeFunction  1   796.99 802.99
+ Age                       1   797.36 803.36
<none>                          808.72 812.72
+ SkinThickness             1   807.07 813.07
+ Insulin                   1   807.77 813.77
+ BloodPressure             1   808.59 814.59

Step:  AIC=777.4
Outcome ~ Glucose + BMI

                           Df Deviance    AIC
+ Pregnancies               1   744.12 752.12
+ Age                       1   755.68 763.68
+ DiabetesPedigreeFunction  1   762.87 770.87
+ Insulin                   1   767.79 775.79
+ BloodPressure             1   769.07 777.07
<none>                          771.40 777.40
+ SkinThickness             1   770.20 778.20

Step:  AIC=752.12
Outcome ~ Glucose + BMI + Pregnancies

                           Df Deviance    AIC
+ DiabetesPedigreeFunction  1   734.31 744.31
+ BloodPressure             1   738.43 748.43
```

```
+ Age                      1   742.10 752.10
<none>                         744.12 752.12
+ Insulin                  1   742.43 752.43
+ SkinThickness            1   743.60 753.60

Step:  AIC=744.31
Outcome ~ Glucose + BMI + Pregnancies +
          DiabetesPedigreeFunction

               Df Deviance    AIC
+ BloodPressure  1   728.56 740.56
+ Insulin        1   731.51 743.51
<none>               734.31 744.31
+ Age            1   732.51 744.51
+ SkinThickness  1   733.06 745.06

Step:  AIC=740.56
Outcome ~ Glucose + BMI + Pregnancies +
          DiabetesPedigreeFunction +
          BloodPressure

               Df Deviance    AIC
+ Age            1   725.46 739.46
+ Insulin        1   725.97 739.97
<none>               728.56 740.56
+ SkinThickness  1   728.00 742.00

Step:  AIC=739.46
Outcome ~ Glucose + BMI + Pregnancies +
          DiabetesPedigreeFunction +
          BloodPressure + Age

               Df Deviance    AIC
+ Insulin        1   723.45 739.45
<none>               725.46 739.46
+ SkinThickness  1   725.19 741.19

Step:  AIC=739.45
Outcome ~ Glucose + BMI + Pregnancies +
          DiabetesPedigreeFunction +
          BloodPressure + Age + Insulin

               Df Deviance    AIC
<none>               723.45 739.45
```

```
+ SkinThickness  1   723.45 741.45
```

What does the final model look like? Which predictor is missing from the final model? Note: AIC is an estimate of out-of-sample prediction error and depends on the likelihood; thus it does not work for models that do not calculate some form of likelihood.

Question 16.10

Here is the output of backward selection for the Pima example, again using R's *MASS* package and AIC as the model performance metric.

```
Start:  AIC=741.45
Outcome ~ Pregnancies + Glucose + BloodPressure + SkinThickness +
    Insulin + BMI + DiabetesPedigreeFunction + Age

                          Df Deviance    AIC
- SkinThickness            1   723.45 739.45
- Insulin                  1   725.19 741.19
<none>                         723.45 741.45
- Age                      1   725.97 741.97
- BloodPressure            1   729.99 745.99
- DiabetesPedigreeFunction 1   733.78 749.78
- Pregnancies              1   738.68 754.68
- BMI                      1   764.22 780.22
- Glucose                  1   838.37 854.37

Step:  AIC=739.45
Outcome ~ Pregnancies + Glucose + BloodPressure + Insulin + BMI +
    DiabetesPedigreeFunction + Age

                          Df Deviance    AIC
<none>                         723.45 739.45
- Insulin                  1   725.46 739.46
- Age                      1   725.97 739.97
- BloodPressure            1   730.13 744.13
- DiabetesPedigreeFunction 1   733.92 747.92
- Pregnancies              1   738.69 752.69
- BMI                      1   768.77 782.77
- Glucose                  1   840.87 854.87
```

What does the final model look like? How does it compare to the model obtained through forward selection?

16.6 Embedded Methods

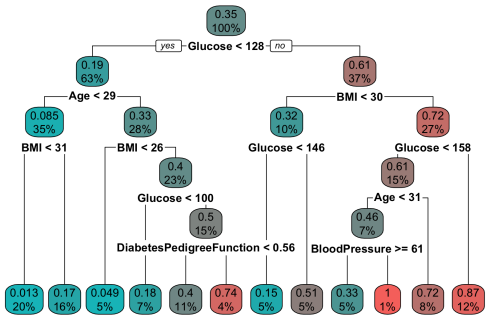
The third and final class of feature selection methods are called **embedded methods**. Embedded methods perform feature selection during the process of model training. They are usually specific to a particular type of model.

16.6.1 Decision Trees

One example of an embedded method is a decision tree (see Chapters 7, 13, and 14), which implicitly performs feature selection by placing the most informative predictors at the top of the tree and ignoring those that are unassociated with the outcome.

Question 16.11

Here is the decision tree produced by CART, using information gain/mutual information as the splitting criterion as usual:



Which features were selected for this tree and which were ignored? How were the features transformed from their original forms in the dataset?

16.6.2 Regularized Models

Another example of an embedded method is **regularization**. The easiest way to understand regularization is through our discussion of maximum likelihood estimation for GLMs in Chapter 12. The goal of maximum likelihood estimation is to find the set of model coefficients, βs, that maximize the joint probability (likelihood) of our observed data given the model. The trouble with this is that more complex models, with more parameters, will generally fit the data better: i.e. produce a higher likelihood.

Regularization addresses this by introducing a penalty term on the likelihood that is proportional to the size of the parameters. In L_1 regularization, a.k.a. **Lasso**, the penalty term is proportional to the absolute values of the coefficients. It looks like this:

$$\lambda \sum_{j=1}^{p} |\beta_j|$$

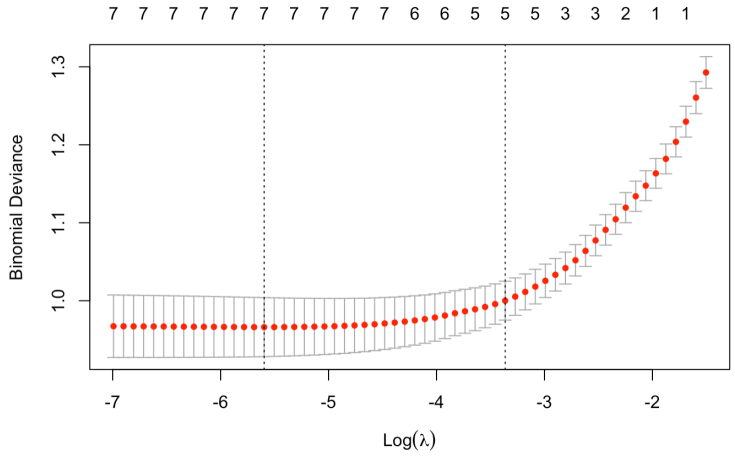
where p is the number of predictors. This creates a tradeoff in the model between the likelihood and the number of parameters. During optimization, the model will set the coefficients on predictors to zero if including those predictors does not sufficiently improve the likelihood. The relative importance of the penalty term and likelihood is adjusted using the parameter λ. We will see regularized regression methods in much greater detail in Chapter 17.

Question 16.12

Here is the raw model output from the multivariate logistic regression model that includes all eight predictors:

```
Call:
glm(formula = Outcome ~ . - LogDiabetesPedigreeFunction, family = "binomial",
    data = d)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5566  -0.7274  -0.4159   0.7267   2.9297

Coefficients:
                           Estimate Std. Error z value Pr(>|z|)
(Intercept)              -8.4046964  0.7166359 -11.728  < 2e-16 ***
Pregnancies               0.1231823  0.0320776   3.840 0.000123 ***
Glucose                   0.0351637  0.0037087   9.481  < 2e-16 ***
BloodPressure            -0.0132955  0.0052336  -2.540 0.011072 *
SkinThickness             0.0006190  0.0068994   0.090 0.928515
Insulin                  -0.0011917  0.0009012  -1.322 0.186065
BMI                       0.0897010  0.0150876   5.945 2.76e-09 ***
DiabetesPedigreeFunction  0.9451797  0.2991475   3.160 0.001580 **
Age                       0.0148690  0.0093348   1.593 0.111192
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 993.48  on 767  degrees of freedom
Residual deviance: 723.45  on 759  degrees of freedom
AIC: 741.45

Number of Fisher Scoring iterations: 5
```

Now let's consider what happens when we use a L_1 regularized logistic regression model, produced using the R package *glmnet*. Here is what happens to the model's error (assessed using 10-fold cross validation; measured using a metric called **binomial deviance**) when we vary λ:




```
Measure: Binomial Deviance

      Lambda Measure      SE Nonzero
```

```
min 0.004468  0.9686 0.02647       7
1se 0.028723  0.9922 0.02118       5
```

We choose λ to be equal to the value that produces the minimum deviance. Here are the coefficients of the final model:

```
9 x 1 sparse Matrix of class "dgCMatrix"
                                      1
(Intercept)              -8.048785391
Pregnancies               0.115632123
Glucose                   0.033559189
BloodPressure            -0.010901115
SkinThickness             .
Insulin                  -0.000837989
BMI                       0.083305233
DiabetesPedigreeFunction  0.847558021
Age                       0.013503422
```

Compare this output to the results of models obtained through forward and backward selection methods, as well as to the full (unregularized) logistic regression model. What are the advantages and disadvantages of the regularization approach vs. wrappers and filters?

Chapter 17

Regularized Regression: Lasso, Ridge, and Elastic Net

In Chapter 16, we discussed embedded feature selection methods. These methods incorporate feature selection into the process of model training. Decision trees and tree ensembles (boosted trees, random forests) naturally perform feature selection in the process of model training, and will simply ignore irrelevant features.

Regression methods, however, have a harder time. When people began to want to apply regression methods to supervised learning problems with large numbers of predictors – particularly when the number of predictors was greater than the number of samples, or when the predictors were highly correlated – they faced serious problems of overfitting and model instability. This led to the development of **regularized regression** (a.k.a. **penalized regression**) methods, which introduce penalty terms into the objective functions being optimized in the model fitting process to try to control the size of the coefficients assigned to various predictors and/or set some of them to zero. The goal of these methods is to allow researchers to stick with the machinery of regression models while reducing the possibility of overfitting.

17.1 Linear Regression

Linear regression (Chapters 3, 8, and 12) uses the mean-squared loss:

$$\text{loss} = \sum_{i=1}^{N} (y^{(i)} - \beta^T x^{(i)})^2$$

where $\beta^T x^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \cdots + \beta_p x_p^{(i)}$ is the model's prediction for the ith training example. To minimize the loss, we adjust the βs to make the model's predictions close to the true $y^{(i)}$ values. This is equivalent to maximizing the likelihood (see Section 12.4).

17.2 Logistic Regression

Logistic regression (Chapters 2, 9, and 12) uses the negative binomial log-likelihood as its loss (compare the expression below to the one in Chapter 12:

$$\text{loss} = -\sum_{i=1}^{n} \left[y^{(i)} \beta^T x^{(i)} - \log \left(1 + \exp(\beta^T x^{(i)}) \right) \right].$$

The expression has a negative sign in front because high log-likelihood is a good thing; we want low values of the log-likelihood to correspond to high values of the loss.

17.3 Lasso, Ridge, and Elastic Net Penalties

In situations where there is a risk of overfitting and model instability (e.g., highly correlated predictors, more predictors than training examples), one can apply a penalty term to the loss function to prevent the model coefficients from taking on unrealistically large or small values.

17.3.1 Lasso Penalty

The **Lasso**, or **L1**, penalty is related to the sum of the absolute values of the coefficients:

$$\text{penalty}_{\text{Lasso}} = \lambda \sum_{j=1}^{p} |\beta_j|$$

This penalty will tend to cause the model to perform feature selection, setting some of the βs to zero. The values of the others may shrink, or they may be unaffected.

17.3.2 Ridge Penalty

The **ridge**, or **L2**, penalty is related to the sum of the squared values of the coefficients:

$$\text{penalty}_{\text{ridge}} = \lambda \sum_{j=1}^{p} \beta_j^2$$

This penalty will tend to shrink the values of all of the model coefficients without setting any to zero.

17.3.3 Elastic Net Penalty

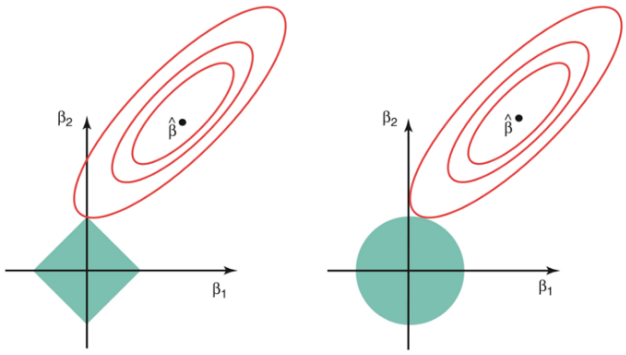
The **elastic net** penalty is just a weighted sum of the Lasso and ridge penalties:

$$\text{penalty}_{\text{EN}} = \lambda \left[\alpha \sum_{j=1}^{p} |\beta_j| + (1 - \alpha) \sum_{j=1}^{p} \beta_j^2 \right]$$

The parameter α governs the relative weights of the two penalties. In practice, λ and α are set using cross-validation. The value of λ is unconstrained, but α must lie between 0 and 1.

Question 17.1

This picture, from *Elements of Statistical Learning* (Figure 6.7) is a geometric picture of what happens to the coefficients under the Lasso and ridge penalties.



Which picture is which? What are the red ellipses? What are the blue shapes? Do you see why Lasso is more likely than ridge to set some of the coefficients to zero?

17.4 Example: Predicting Blood Pressure

Imagine we have collected blood pressure data on 10 patients. In addition, we have information on the patients' sex, age, and obesity status.

	sexf	sexm	age	obesity	blood_pressure
1	0	1	54	1	123
2	1	0	66	0	111
3	1	0	23	0	98
4	0	1	59	1	154
5	0	1	76	1	199
6	1	0	33	0	101
7	0	1	35	1	91
8	1	0	54	0	133
9	1	0	21	0	116
10	0	1	26	0	121

Note: These data are from an example in the paper "Predictive analytics with gradient boosting in clinical medicine", by Zhang et al, published in *Annals of Translational Medicine* in 2019.

Question 17.2

Which two predictors in this model are clearly correlated?

We will now try many different values of λ and α on these data and see what happens to the coefficients. To do this, we use the `glmnet` package in R. Here are the results:

	lambda	alpha	beta_sexf	beta_sexm	beta_age	beta_obesity
1	0.0	0.0	-24.61	-0.00	1.15	-13.78
2	1.0	0.0	-11.19	10.71	1.09	-9.97
3	5.0	0.0	-8.25	8.24	0.91	-2.11
4	10.0	0.0	-6.97	6.99	0.77	1.66
5	0.0	0.2	-24.61	-0.00	1.15	-13.78
6	1.0	0.2	-10.35	9.73	1.07	-7.84
7	5.0	0.2	-6.81	6.72	0.88	0.00
8	10.0	0.2	-6.10	6.07	0.76	0.00
9	0.0	0.5	-24.61	-0.00	1.15	-13.78
10	1.0	0.5	-9.09	8.04	1.04	-4.38
11	5.0	0.5	-5.65	5.50	0.87	0.00
12	10.0	0.5	-3.84	3.77	0.71	0.00
13	0.0	1.0	-24.61	-0.00	1.15	-13.78
14	1.0	1.0	-13.15	0.00	1.00	0.00
15	5.0	1.0	-6.93	0.00	0.84	0.00
16	10.0	1.0	0.00	0.00	0.62	0.00

Question 17.3

Which values for λ and α correspond to: (a) unregularized linear regression, (b) pure Lasso, (c) pure ridge, (d) elastic net with an even combination of Lasso and ridge penalties? What happens to the values of the coefficients in each case?

Chapter 18

The Cox Proportional Hazards Model

We just encountered the Kaplan-Meier estimate of the survival function in Chapter 11. Now we are going to talk about models that essentially treat the survival function as the *outcome* in a supervised learning problem. These models are called **Cox proportional hazards models**.

18.1 Survival and Hazard Functions

Consider a situation where we have some process that generates events, and we're trying to model the time to first event. Assume the probability of the event's occurring at each time, t, is given by the function $f(t)$. The cumulative probability of the event's having occurred by time t is

$$F(t) = \int_0^t f(t)dt$$

and the probability of an individual not having experienced the event by time t is

$$S(t) = 1 - F(t),$$

the **survival function**. The probability of experiencing the event in an infinitesimally small interval starting at t, given that one has not experienced it

by time t, is:

$$\lambda(t) = \frac{f(t)}{S(t)}$$

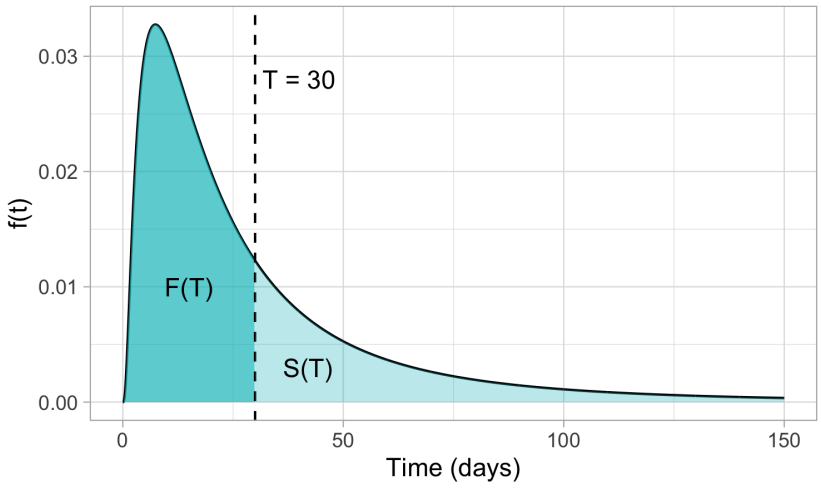
and is called the **hazard**. The **cumulative hazard** function is equal to

$$\Lambda(t) = \int_0^t \lambda(t')dt' = -\log S(t)$$

so $S(t) = \exp(-\Lambda(t))$.

None of these expressions should be immediately obvious to you, because deriving them requires calculus. It's a great exercise to go through the derivations, but for now, let's focus on capturing the intuition.

Here is a graphical representation of some of these quantities. Remember that the probability distribution $f(t)$ must integrate to one.



Question 18.1

What's the interpretation of the hazard, $\lambda(t)$, on this image? What happens to the hazard if $S(t)$ is low vs. high for the same $f(x)$?

Question 18.2

Assume $f(t)$ is exponential: $f(t) = b \cdot \exp(-bt)$, where b is constant. Then $F(t) = 1 - \exp(-bt)$ and $S(t) = \exp(-bt)$. What is the hazard, $\lambda(t)$? What is the cumulative hazard, $\Lambda(t)$?

Question 18.3

The concept of a "cumulative hazard" is pretty weird. How should this quantity be interpreted?

18.2 Estimating Survival and Cumulative Hazard

The Kaplan-Meier estimate of survival (Chapter 11) is the most common estimate of the survival function. One can estimate the cumulative hazard using a couple of different methods.

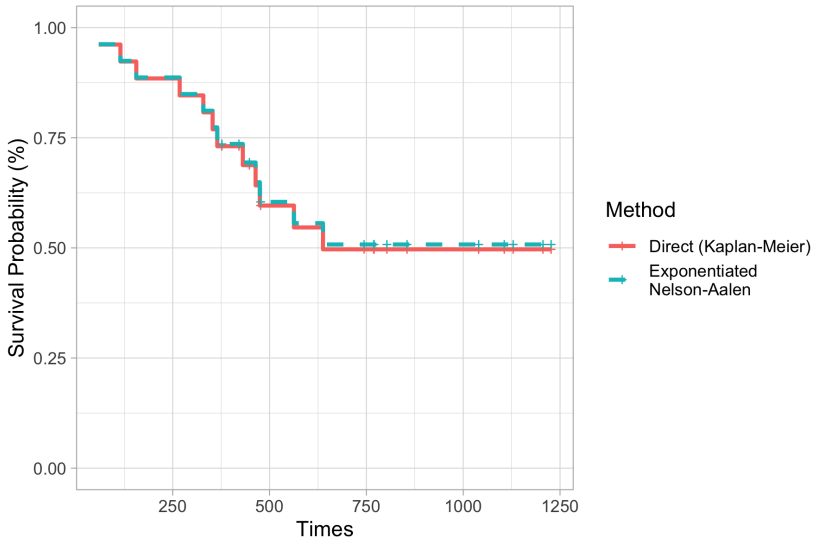
1. Take the negative log of the Kaplan-Meier estimate of survival:

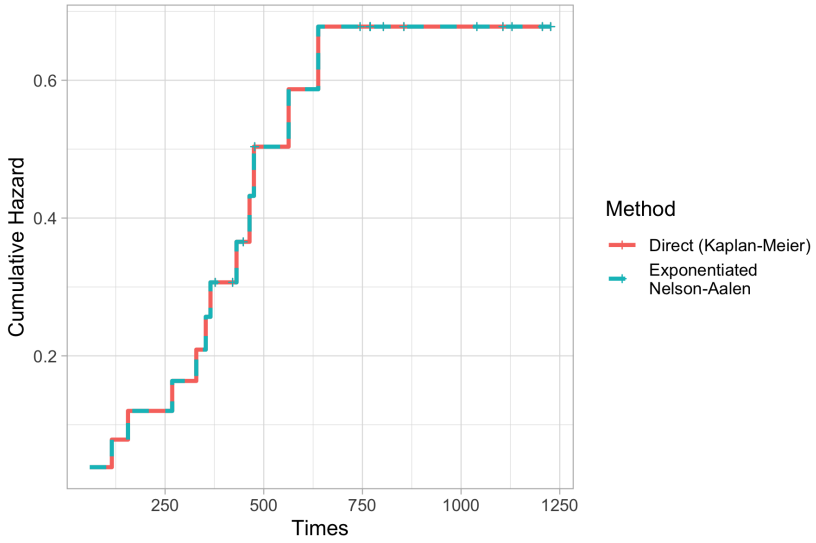
$$\begin{aligned}\hat{\Lambda}_{KM}(t) &= -\log \hat{S}_{\text{KM}}(t) \\ &= -\sum_{i:t_i<t} \log\left(1 - \frac{d_i}{n_i}\right)\end{aligned}$$

2. Use the **Nelson-Aalen estimator**

$$\hat{\Lambda}_{NA}(t) = \sum_{i:t_i<t} \frac{d_i}{n_i}$$

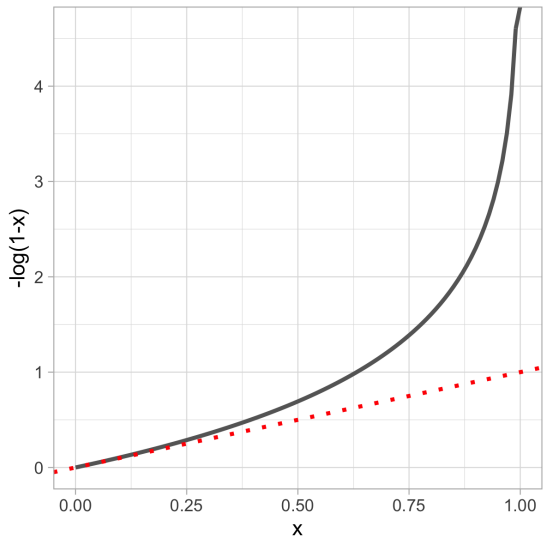
One thing that is confusing about the R `survival` package's output is that is uses the Kaplan-Meier estimate for survival by default, but it uses the Nelson-Aalen estimator for cumulative hazard by default. So if you exponentiate the negative cumulative hazard that comes out of `survfit`, it won't match the survival estimate. The two are close in most cases, though. Here are some pictures for the ovarian cancer survival dataset we discussed in Chapter 11:


1.00
0.75
0.50
0.25
0.00
Survival Probability (%)
250
500
750
1000
1250
Times
Method
Direct (Kaplan-Meier)
Exponentiated
Nelson-Aalen


0.6
0.4
0.2
Cumulative Hazard
250
500
750
1000
1250
Times
Method
Direct (Kaplan-Meier)
Exponentiated
Nelson-Aalen

Question 18.4

Here is a plot showing the function $-\log(1-x)$ vs. x. Look at the expressions for the two estimators for the cumulative hazard, above. Under what conditions will they be similar? Under what conditions will they be different?



Question 18.5

The curvature of the Nelson-Aalen estimator gives you an idea of how the hazard varies with time. A concave shape is an indication of *deceleration* of the hazard; for example, if the event in question is death and time is patient age, this would represent higher infant/childhood mortality than adult mortality. A convex shape is an indication of *acceleration* of the hazard; in the death/age example, this would represent a process that accelerates as one ages (so called "wear-out mortality").

Looking at the graph of the overall cumulative hazard, what do you notice about how the hazard for ovarian cancer changes since the initiation of treatment?

Question 18.6

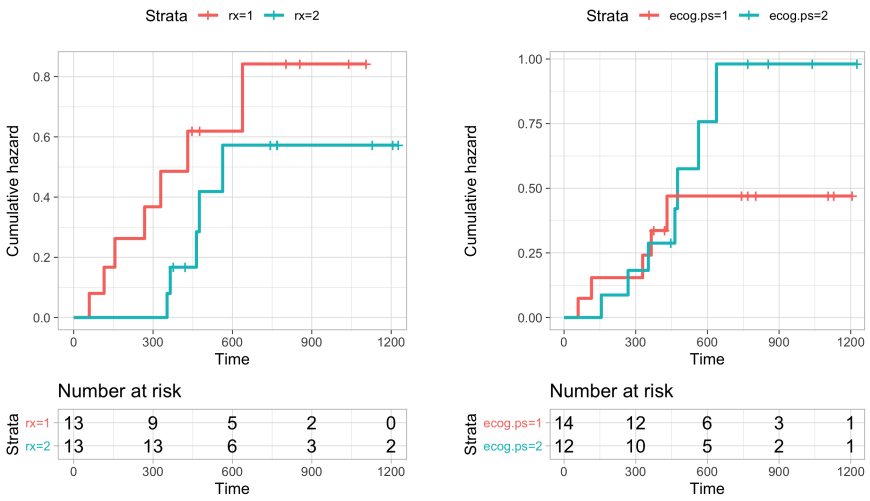
Here are the raw data from treatment group 1 of the `ovarian` dataset. These are the same data we looked at when building the Kaplan-Meier curve in Chapter 11, Question 11.1. Using these data, fill in the remaining cells of the table below. Here we are using the Nelson-Aalen estimator for the cumulative hazard.

	rx	futime	fustat
1	1	59	1
2	1	115	1
3	1	156	1
4	1	268	1
5	1	329	1
6	1	431	1
7	1	448	0
8	1	477	0
9	1	638	1
10	1	803	0
11	1	855	0
12	1	1040	0
13	1	1106	0

j	t_j	n_j	d_j	$\hat{\Lambda}(t_j)$	Calculation
0	0	13	0	0.000	$\frac{0}{13}$
1	59	13	1	0.077	$\hat{\Lambda}(t_0) + \frac{1}{13}$
2	115	12	1	0.160	$\hat{\Lambda}(t_1) + \frac{1}{12}$
3	156				
4	268				
5	329	9	1	0.462	$\hat{\Lambda}(t_4) + \frac{1}{9}$
6	431	8	1	0.587	$\hat{\Lambda}(t_5) + \frac{1}{8}$
7	448	7	0	0.587	$\hat{\Lambda}(t_6) + \frac{0}{7}$
8	477	6	0	0.587	$\hat{\Lambda}(t_7) + \frac{0}{6}$
9	638	5	1	0.787	$\hat{\Lambda}(t_8) + \frac{1}{5}$
10	803	4	0		
11	855	3	0		
12	1040	2	0		
13	1106	1	0		

Question 18.7

Here are plots of the cumulative hazard (Nelson-Aalen estimator) for patients by sex and by ECOG performance score status:



How does treatment group appear to impact the cumulative hazard? What about ECOG score? Do the cumulative hazards appear proportional (i.e., related by a common multiplier) in each case?

18.3 Deriving the Cox Model

We have spent considerable time on the hazard and cumulative hazard because these play important roles in what is perhaps the most famous survival analysis tool: the Cox proportional hazards model ("Cox model", for short), developed by D.R. Cox in 1972. The model has the following form:

$$\lambda(t|x) = \lambda_0(t)\exp(\beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p)$$

and another way of writing it is:

$$\log\left(\frac{\lambda(t|x)}{\lambda_0(t)}\right) = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p.$$

Note that for now we are assuming that the covariates do not depend on time. There is a variant of the Cox model called the **extended Cox model** that allows time-dependent covariates. For now, we will just consider the fixed covariate case.

Question 18.8

Compare the Cox model to a logistic regression model. What is the same? What is different?

Here $\lambda_0(t)$ is called the **baseline hazard**. As usual, we symbolize the linear sum of the βs as $\beta^T x$. Importantly, the baseline hazard can take any shape as a function of time. The $\lambda_0(t)$ part of the equation is, therefore, referred to as the "nonparametric" part, while the $\beta^T x$ part is called the "parametric" part. The overall model is referred to as **semiparametric**.

The ratio of the hazards for two different sets of covariates, x and z, is

$$\frac{\lambda(t|x)}{\lambda(t|z)} = \frac{\lambda_0(t)\exp(\beta^T x)}{\lambda_0(t)\exp(\beta^T z)} = \exp(\beta^T(x-z)).$$

Taking the log of this, we arrive at

$$\log\left(\frac{\lambda(t|x)}{\lambda(t|z)}\right) = \beta^T(x-z).$$

A single coefficient, β_j, is therefore the **hazard ratio** when the corresponding predictor, x_j, increases by one. This ratio is assumed to be constant over time. The hazard ratio is also called the **relative risk**.

Question 18.9

Why doesn't the Cox model have an intercept, β_0?

Question 18.10

Compare the interpretation of the coefficients in a Cox model to their interpretation in a logistic regression model.

18.4 Fitting the Cox Model

Fitting a Cox model means taking a sample of possibly right-censored data and deriving estimates for the parameters, $\beta_1, \ldots, \beta_p$. Cox models are fit using a variant of maximum likelihood estimation called **partial likelihood estimation**.

Let's consider all of the unique times, t_i, that events are observed. For now, we will assume that exactly one event is observed at each of these times (no ties). We will use $R(t_i)$ to refer to the set of subjects who are "at risk" (i.e., not censored) just prior to time t_i. At each failure time, t_i, the contribution to the partial likelihood is:

$$\begin{aligned}\mathcal{L}_i(\beta) &= \frac{P(\text{person } i \text{ experiences event} \mid \text{still around at } t_i)}{\sum_{l \in R(t_i)} P(\text{person } l \text{ experiences event} \mid \text{still around at } t_i)} \\ &= \frac{\lambda(t_i|x_i)}{\sum_{l \in R(t_i)} \lambda(t_i|x_l)} = \frac{\exp(\beta^T x^{(i)})}{\sum_{l \in R(t_i)} \exp(\beta^T x^{(l)})}\end{aligned}$$

The complete partial likelihood over all K observed event times, $t_i = t_1, \ldots, t_K$, is:

$$\mathcal{L}(\beta) = \prod_{i=1}^{K} \frac{\exp(\beta^T x^{(i)})}{\sum_{l \in R(t_i)} \exp(\beta^T x^{(l)})}.$$

This is the thing that the model fitting process is trying to maximize. It is optimized numerically, using the same types of optimization procedures used by logistic regression and other generalized linear models.

Question 18.11

Why is this quantity called a "partial likelihood", instead of just a "likelihood"?

18.5 Interpreting Cox Models

Here is a simple Cox model for survival time vs. age, residual disease, ECOG score, and treatment group in the ovarian cancer dataset.

```{r}
m <- coxph(Surv(futime, fustat) ~ age + resid.ds + ecog.ps + rx, data = d)
summary(m)
```

```
Call:
coxph(formula = Surv(futime, fustat) ~ age + resid.ds + ecog.ps +
    rx, data = d)

  n= 26, number of events= 12

              coef exp(coef) se(coef)      z Pr(>|z|)
age        0.12481   1.13294  0.04689  2.662  0.00777 **
resid.ds2  0.82619   2.28459  0.78961  1.046  0.29541
ecog.ps2   0.33621   1.39964  0.64392  0.522  0.60158
rx2       -0.91450   0.40072  0.65332 -1.400  0.16158
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

          exp(coef) exp(-coef) lower .95 upper .95
age          1.1329     0.8827    1.0335     1.242
resid.ds2    2.2846     0.4377    0.4861    10.738
ecog.ps2     1.3996     0.7145    0.3962     4.945
rx2          0.4007     2.4955    0.1114     1.442

Concordance= 0.807  (se = 0.068 )
Likelihood ratio test= 17.04  on 4 df,   p=0.002
Wald test            = 14.25  on 4 df,   p=0.007
Score (logrank) test = 20.81  on 4 df,   p=3e-04
```

Question 18.12

Interpret the coefficients, exponentiated coefficients, standard errors of the coefficients, Z scores, and p-values in this model. Also try your hand at interpreting the second block of model output, which includes the exponentiated coefficients (again), the exponentiated negative coefficients, and the lower and upper bounds of a 95% confidence interval for the exponentiated coefficients.

18.6 Making Predictions with Cox Models

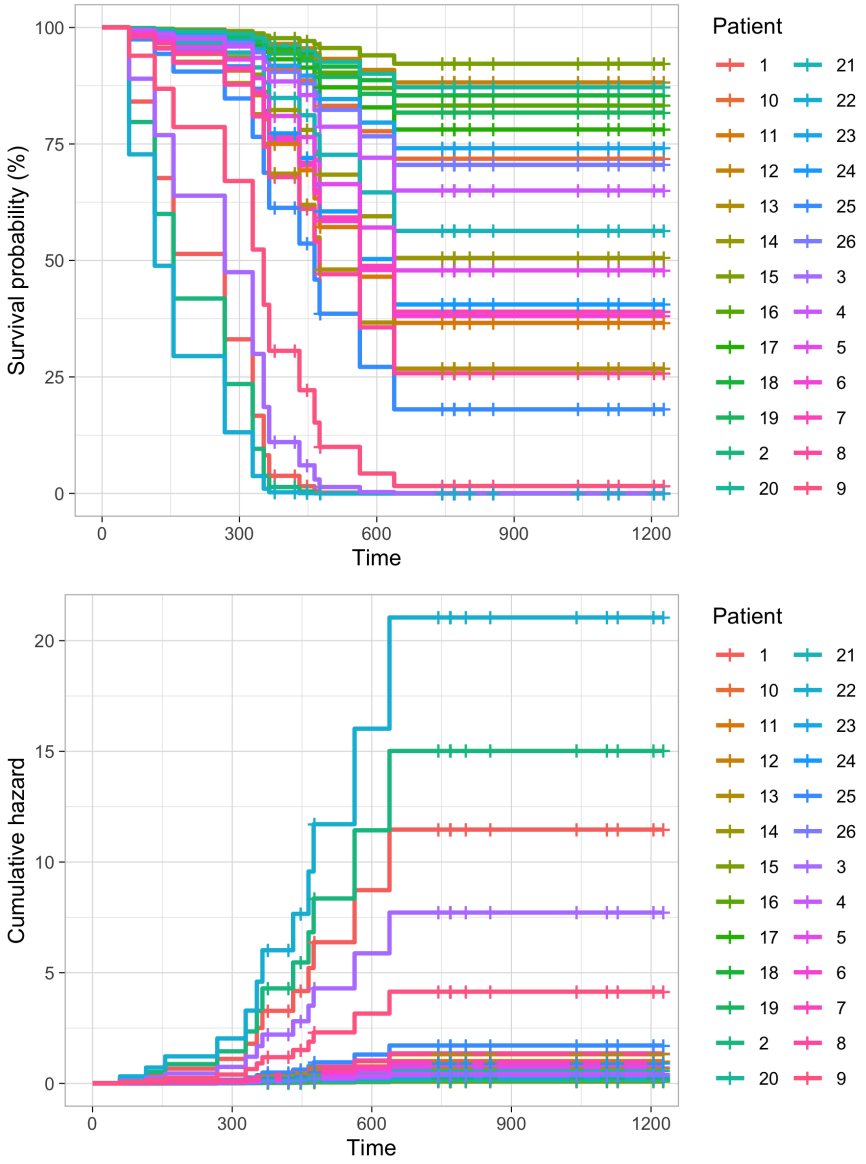
The Cox model fitted using the `coxph` function can be used to make various predictions on the original dataset. This yields a lot of output. All of the available outputs for the `ovarian` dataset are shown below.

	age	lp_age	resid.ds	lp_resid.ds	ecog.ps	lp_ecog.ps	rx	lp_rx	lp	risk	expected	surv
1	72.33	9.03	2	0.83	1	0.00	1	0.00	2.67	14.43	0.16	0.85
2	74.49	9.30	2	0.83	1	0.00	1	0.00	2.94	18.90	0.46	0.63
3	66.47	8.30	2	0.83	2	0.34	1	0.00	2.27	9.71	0.40	0.67
4	53.36	6.66	2	0.83	1	0.00	2	-0.91	-0.61	0.54	0.11	0.89
5	50.34	6.28	2	0.83	1	0.00	1	0.00	-0.08	0.93	0.25	0.78
6	56.43	7.04	1	0.00	2	0.34	1	0.00	0.19	1.21	0.33	0.72
7	56.94	7.11	2	0.83	2	0.34	2	-0.91	0.17	1.18	0.40	0.67
8	59.85	7.47	2	0.83	2	0.34	2	-0.91	0.53	1.71	0.70	0.49
9	64.18	8.01	2	0.83	1	0.00	1	0.00	1.65	5.21	2.15	0.12
10	55.18	6.89	1	0.00	2	0.34	2	-0.91	-0.88	0.42	0.24	0.79
11	56.76	7.08	1	0.00	2	0.34	1	0.00	0.24	1.27	0.94	0.39
12	50.11	6.25	1	0.00	1	0.00	2	-0.91	-1.84	0.16	0.12	0.89
13	59.63	7.44	2	0.83	2	0.34	2	-0.91	0.51	1.66	1.23	0.29
14	57.05	7.12	2	0.83	1	0.00	2	-0.91	-0.15	0.86	0.63	0.53
15	39.27	4.90	1	0.00	1	0.00	1	0.00	-2.28	0.10	0.08	0.93
16	43.12	5.38	1	0.00	2	0.34	1	0.00	-1.47	0.23	0.17	0.84
17	38.89	4.85	2	0.83	2	0.34	1	0.00	-1.17	0.31	0.23	0.79
18	44.60	5.57	1	0.00	1	0.00	1	0.00	-1.62	0.20	0.15	0.86
19	53.91	6.73	1	0.00	1	0.00	2	-0.91	-1.37	0.25	0.19	0.83
20	44.21	5.52	2	0.83	1	0.00	2	-0.91	-1.76	0.17	0.13	0.88
21	59.59	7.44	1	0.00	2	0.34	2	-0.91	-0.33	0.72	0.53	0.59
22	74.50	9.30	2	0.83	2	0.34	1	0.00	3.28	26.49	1.65	0.19
23	43.14	5.38	2	0.83	1	0.00	1	0.00	-0.97	0.38	0.04	0.96
24	63.22	7.89	1	0.00	2	0.34	2	-0.91	0.13	1.14	0.18	0.84
25	64.42	8.04	2	0.83	1	0.00	2	-0.91	0.77	2.16	0.45	0.64
26	58.31	7.28	1	0.00	1	0.00	2	-0.91	-0.82	0.44	0.09	0.91

The term `age` is the raw value for age for each patient, and the term `lp_age` is the linear predictor, $\beta_{\text{age}} x_{\text{age}}^{(i)}$, for patient i. The same is true for the other predictors. The term `lp` is the entire linear predictor, $\beta^T x$, for each $x^{(i)}$. Confusingly, it has been centered, so its value is shifted from the sum of columns 2, 4, 6, and 8 by a fixed amount (Exercise: What is this amount?). The `risk` term is just the overall risk score, exp(`lp`). The `expected` term is the expected number of events given the covariates and follow-up time. The survival probability, `surv`, for each subject is exp(−`expected`).

Cox models make no assumptions about the shape of the baseline hazard, so to make predictions, they simply use the empirical survival (or cumulative hazard) curve for the entire dataset and then adjust it up or down depending on the values of the covariates. The way R does this is super confusing - it estimates the baseline hazard at the means of the covariates after centering, so the baseline hazard is not very interpretable. You can get it out of the model using the `basehaz` function. In any case, here's what you get when you ask

the model to predict the survival and cumulative hazard curves for all of the patients in the training set:



Question 18.13

Which patients have the lowest and highest risk scores? Where do they appear on the patient-level survival and cumulative hazard graphs?

18.7 Testing the Proportional Hazards Assumption

The Cox model makes three important assumptions:

1. *Common baseline hazard.* At any time, t, all individuals experience the same baseline hazard, $\lambda_0(t)$.
2. *Proportional hazards.* The hazard for one individual is proportional to the hazard of any other individual.
3. *Time-invariance.* The constant of proportionality between the hazards of any two individuals does not depend on time.

All of them are potentially problematic. In particular, it's hard to come up with situations in which the hazards for *any* two individuals can reasonably be assumed to be proportional. Thus, it's important to check this assumption.

There are whole book chapters and papers devoted to model diagnostics for the Cox model. I will present a couple of common methods here and leave the others to the course website.

18.7.1 Schoenfeld Residuals

In Section 18.4, we saw how the Cox model was fit using maximization of the partial likelihood. The quantity

$$\frac{\exp(\beta^T x^{(i)})}{\sum_{l \in R(t_i)} \exp(\beta^T x^{(l)})}$$

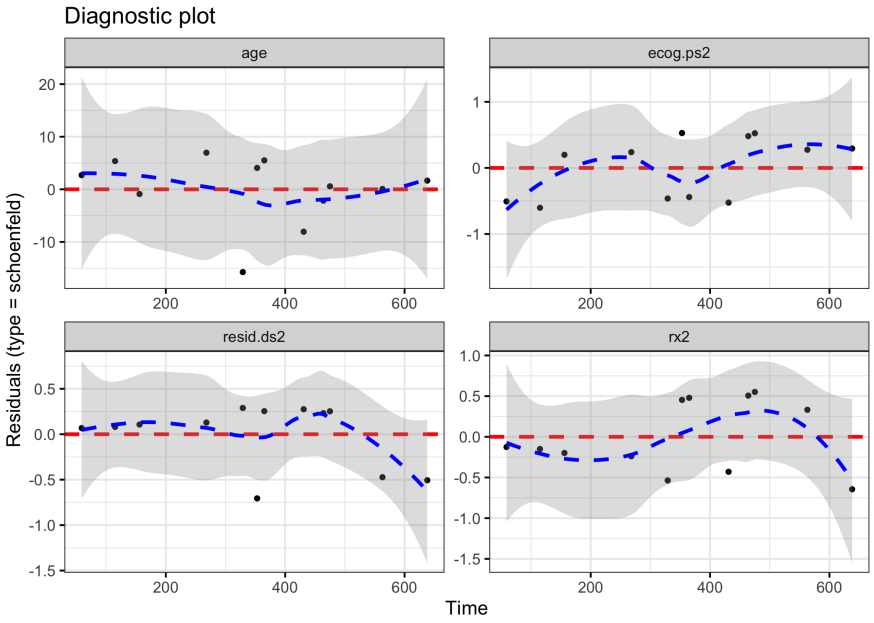
was important because it gave us the probability, according to the model, that the person observed to experience the event at time t_i would experience

it, given all the people in the risk set just prior to time t_i. The **Schoenfeld residual** capitalizes on this idea.

> **Schoenfeld residual:** The covariate value $x_j^{(i)}$ for the person (i) who actually experienced the event at time t_i, minus the expected value of the covariate for the risk set at t_i. Or:

$$\text{residual} = x_j^{(i)} - \sum_{l \in R(t_i)} x_j^{(l)} \frac{\exp(\beta^T x^{(l)})}{\sum_{m \in R(t_i)} \exp(\beta^T x^{(m)})}$$

There is one Schoenfeld residual for each combination of observed event and covariate. Typically, they will be plotted against time to assess if there is a trend. The test for trend comes from a simple linear regression model of the residuals against time, conducted separately for each covariate.



Question 18.14

How would you conduct the test for trend for each covariate using a simple linear regression model?

Question 18.15

Try calculating the Schoenfeld residual for a single covariate and failure time (your choice). All of the information you need is in the table below. The `risk` column is the same as in the previous table. It is exp $(\beta^T x)$.

	futime	fustat	age	resid.ds	ecog.ps	rx	risk
1	59	1	72.33	2	1	1	14.43
2	115	1	74.49	2	1	1	18.90
3	156	1	66.47	2	2	1	9.71
22	268	1	74.50	2	2	1	26.49
23	329	1	43.14	2	1	1	0.38
24	353	1	63.22	1	2	2	1.14
25	365	1	64.42	2	1	2	2.16
26	377	0	58.31	1	1	2	0.44
4	421	0	53.36	2	1	2	0.54
5	431	1	50.34	2	1	1	0.93
6	448	0	56.43	1	2	1	1.21
7	464	1	56.94	2	2	2	1.18
8	475	1	59.85	2	2	2	1.71
9	477	0	64.18	2	1	1	5.21
10	563	1	55.18	1	2	2	0.42
11	638	1	56.76	1	2	1	1.27
12	744	0	50.11	1	1	2	0.16
13	769	0	59.63	2	2	2	1.66
14	770	0	57.05	2	1	2	0.86
15	803	0	39.27	1	1	1	0.10
16	855	0	43.12	1	2	1	0.23
17	1040	0	38.89	2	2	1	0.31
18	1106	0	44.60	1	1	1	0.20
19	1129	0	53.91	1	1	2	0.25
20	1206	0	44.21	2	1	2	0.17
21	1227	0	59.59	1	2	2	0.72

Question 18.16

The R function `cox.zph` performs the test of trend for all predictors as well as a global test of trend using ANOVA (don't worry, we'll get to this later). Here is the output for this model:

```
          chisq df     p
age       0.170  1 0.680
resid.ds  1.155  1 0.282
ecog.ps   2.928  1 0.087
rx        0.595  1 0.440
GLOBAL    4.455  4 0.348
```

For which predictors is there a potentially worrying association between the Schoenfeld residuals and time?

18.7.2 What to do if Violations are Found

Interpretation of the Cox model is relatively insensitive to deviations from proportionality, especially for large sample sizes. However, if nonproportionality is a huge issue for one or more predictors, there are a few strategies to deal with it.

1. *Stratify.* One can stratify the model by different levels of the problematic predictor(s), essentially building separate models for the other covariates at each different level of the problematic predictor(s). This only works for predictors that have discrete levels, however; otherwise, one would need to discretize. A potential downside is that stratification eliminates the model's ability to quantify the effect of the stratification variable(s).

2. *Partition the time axis.* Sometimes proportionality holds for the first part of the time axis but falls apart at the end. In that case, one can analyze the data from the first part of the study separately. The disadvantage, of course, is that one must throw out information from later parts of the study.

3. *Add a nonlinear effect term.* Continuous covariates with nonlinear effects on the outcome may lead to nonproportional hazards. Including transformations of these covariates may help to alleviate the nonproportional hazards.

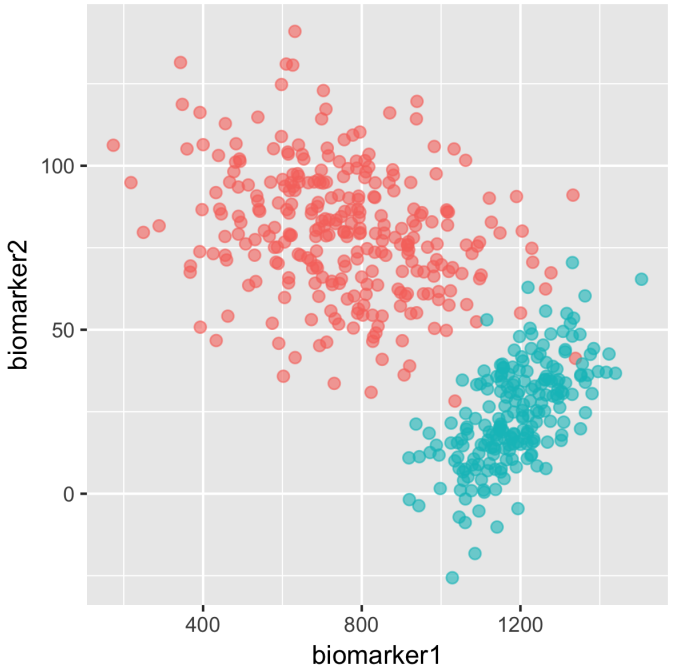
Chapter 19

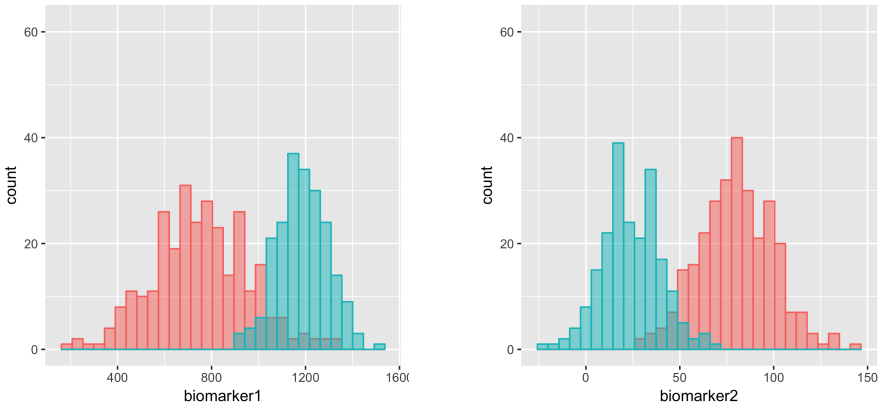
Clustering

Our discussions so far have focused on the problem of supervised learning, in which we create a mapping between a set of inputs and an output. In this chapter, we turn our attention to the problem of **unsupervised learning**, in which our goal is to uncover hidden structure in a dataset. There is no "special" outcome variable in these types of problems. Way back in Chapter 1, examples 7, 8, and 14 were examples of unsupervised learning problems.

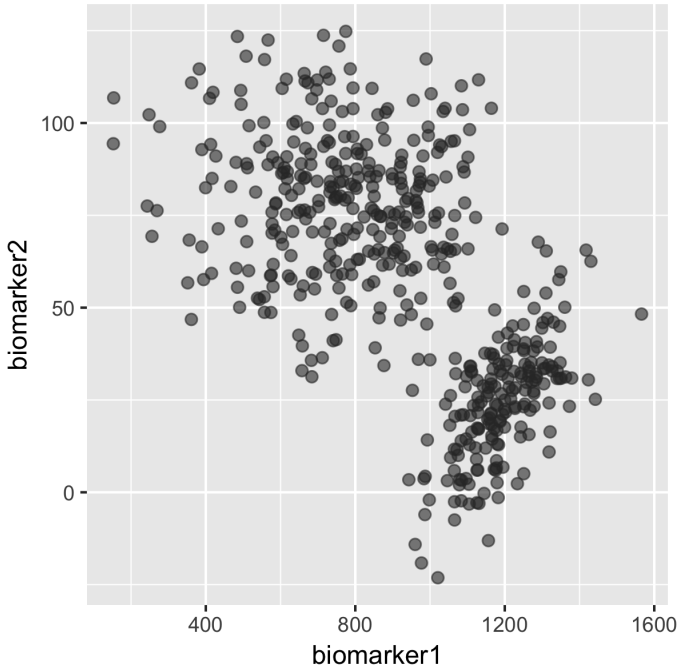
19.1 A Thought Experiment: Flow Cytometry Data

Imagine you have data on 500 cells from two different cell lines. For each cell, you record the fluorescence intensity of two different biomarkers, biomarker 1 (x_1) and biomarker 2 (x_2). The data are shown below.





In real life, you would not have the labels for the two cell lines. You would have no idea which distribution(s) the data were drawn from or what the probabilities associated with the various distributions were. Real flow cytometry data would instead look like this:



The human eye can distinguish structure in a plot like this, but it's harder to train a computer to see it, and it's even harder to prove that the structure the computer finds is real. Over the years, people have tried many different strategies.

Question 19.1

Speculate on some possible approaches for separating data that look like this into groups, or clusters.

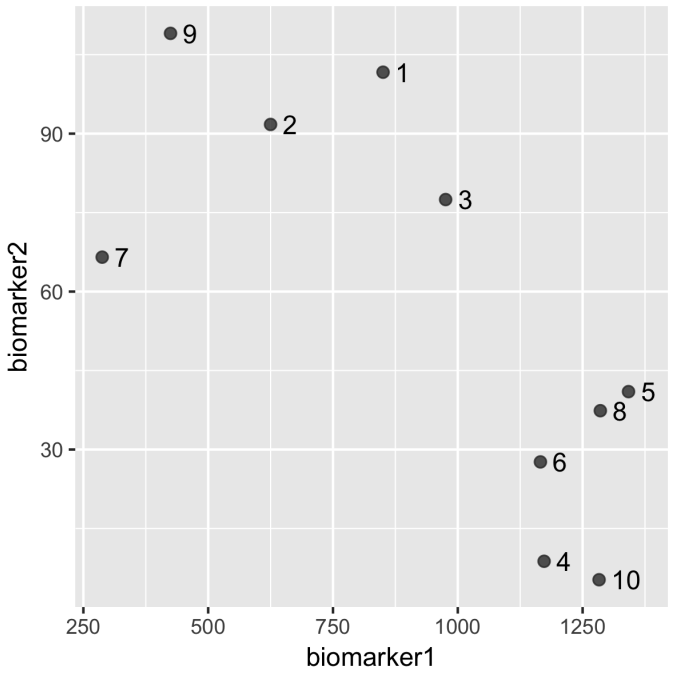
19.1.1 Downsampled Dataset

For the rest of this workshop, we need to be able to calculate things by hand, so we will downsample the flow cytometry data, above, to just 10 datapoints. The downsampled, labeled data are shown here:

i	Biomarker 1 Intensity ($x_1^{(i)}$)	Biomarker 2 Intensity ($x_2^{(i)}$)	Cell Line (z)
1	634.83	110.55	B
2	650.06	74.22	B
3	788.24	81.52	B
4	771.47	84.98	B
5	515.81	91.08	B
6	1101.23	31.05	A
7	649.32	77.05	B
8	652.89	97.16	B
9	1183.02	11.73	A
10	1238.45	33.46	A

Without their labels, the data look like this:

i	Biomarker 1 Intensity ($x_1^{(i)}$)	Biomarker 2 Intensity ($x_2^{(i)}$)	Cell Line (z)
1	634.83	110.55	
2	650.06	74.22	
3	788.24	81.52	
4	771.47	84.98	
5	515.81	91.08	
6	1101.23	31.05	
7	649.32	77.05	
8	652.89	97.16	
9	1183.02	11.73	
10	1238.45	33.46	



19.2 K-Means

Let's first consider a very simple unsupervised machine learning algorithm called **K-means**. The goal of K-means is to cluster [unlabeled] data into groups so that the distances between points within a group are minimized.

Assume you have a dataset $\{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\}$, where each vector $x^{(i)}$ is of length p, and you want to cluster these n vectors into K distinct groups. Here is the K-means algorithm:

1. Assign each of the n datapoints to a random cluster. You can do this one of three ways:

   (1) Choose a random cluster for each point independently.

   (2) Choose K initial points to be the cluster centers.

   (3) Choose K initial points uniformly within the feature space (not necessarily data point locations) to be the cluster centers.

After the initial cluster assignments are made, proceed to the update step, below.

2. **Assignment step.** Assign each point to the cluster whose mean is the closest, using Euclidean distance. Mathematically:

$$c_i^{(t)} := \text{argmin}_j \|x^{(i)} - \mu_j\|$$

where we note that the distance is given by the L_2, or Euclidean, norm.

3. **Update step.** Calculate the means to be the centroids of the points in the clusters.

$$\mu_j^{(t)} := \frac{\sum_{i=1}^{n} x^{(i)} \cdot \mathbb{I}\left\{c^{(i)} = j\right\}}{\sum_{i=1}^{n} \mathbb{I}\left\{c^{(i)} = j\right\}}$$

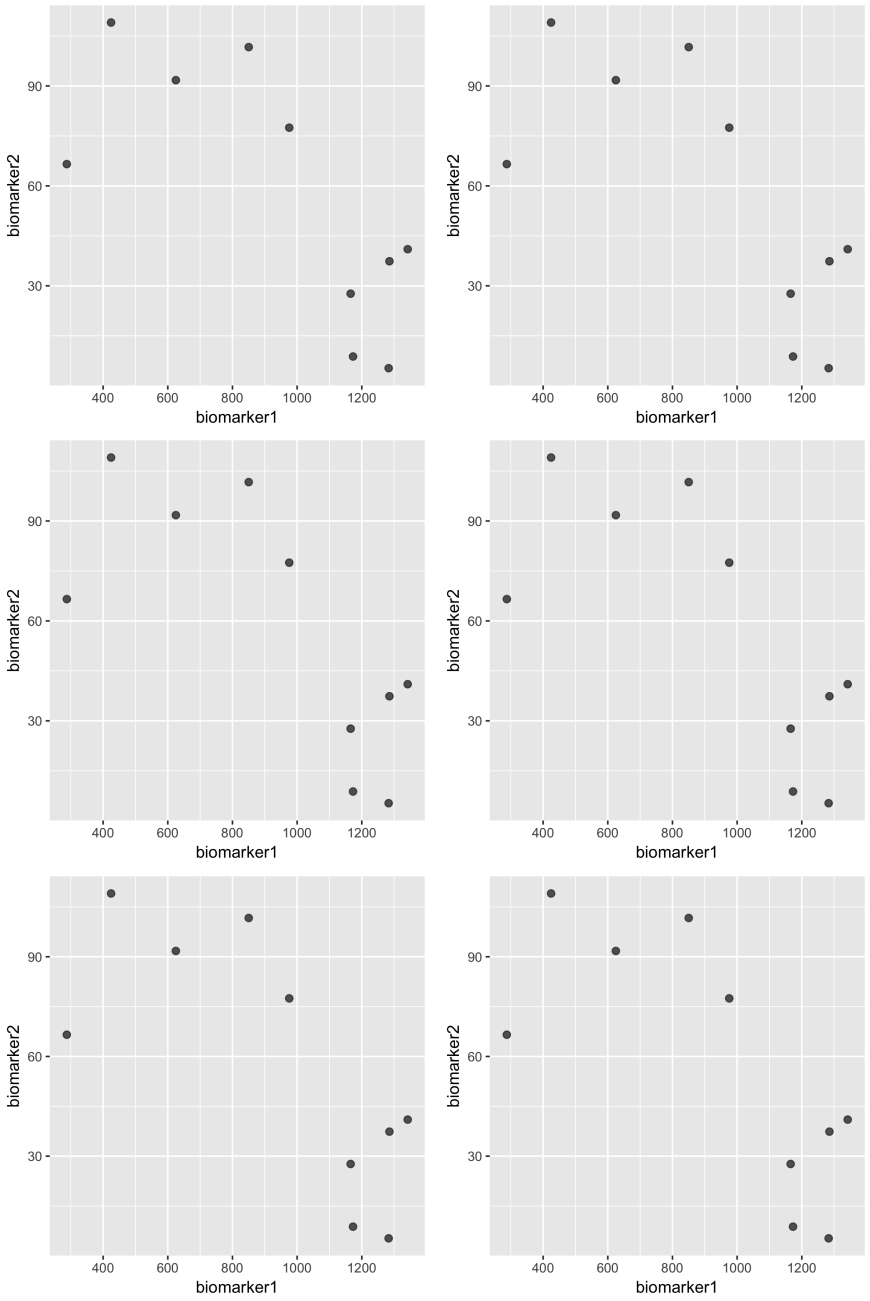
4. Repeat Steps 2 and 3 until no points change clusters (or until convergence).

Question 19.2

Which supervised learning algorithm does this remind you of? What is different between K-means and this algorithm?

Question 19.3

Below is our unlabeled, downsampled flow cytometry dataset. Cluster it into two groups using K-means. You can initialize your clusters however you want. (Note: Assume that the data were standardized in advance so that the spacing between the white lines equals one "unit" for both biomarker 1 and biomarker 2.)



Question 19.4

What are some disadvantages of K-means?

Question 19.5

One application of K-means clustering is image compression, as shown in the figure below (which is from Christopher Bishop's classic book *Pattern Recognition and Machine Learning*).

Draw a picture of the data matrix for the image compression clustering problem. How will the clustering work here? Are we clustering rows or columns? What are the dimensions of the dataset, n and p?

19.3 Mixture Models

Mixture models represent data using mixtures of simple probability distributions. They are a step up in methodological rigor from K-means and are much more flexible, although the basic idea is the same.

Here are the key similarities:

1. The number of clusters, K, is still an input parameter.
2. Mixture models can still converge to local minima depending on the initialization.

Here are the key differences:

1. Mixture models use a **soft clustering** instead of a hard clustering. Each datapoint is assigned a probability distribution over potential clusters.

2. Instead of a distance metric like Euclidean distance, points are distributed over clusters by considering probability densities of the various clusters and how likely it is that the point came from each probability density.

3. No need to assume axes are on an equal scale; the individual probability distributions can account for this.

We'll examine mixtures of Gaussians today because they're easy to visualize, but in reality the mixture components can be any type of well-behaved probability distribution.

19.4 Multivariate Gaussian Distribution

We already saw the univariate Gaussian in Chapter 4. The **multivariate Gaussian** is an extension of the Gaussian to multiple dimensions.

19.4.1 Probability Density and Parameters

The m-dimensional **multivariate Gaussian** probability distribution is given by:

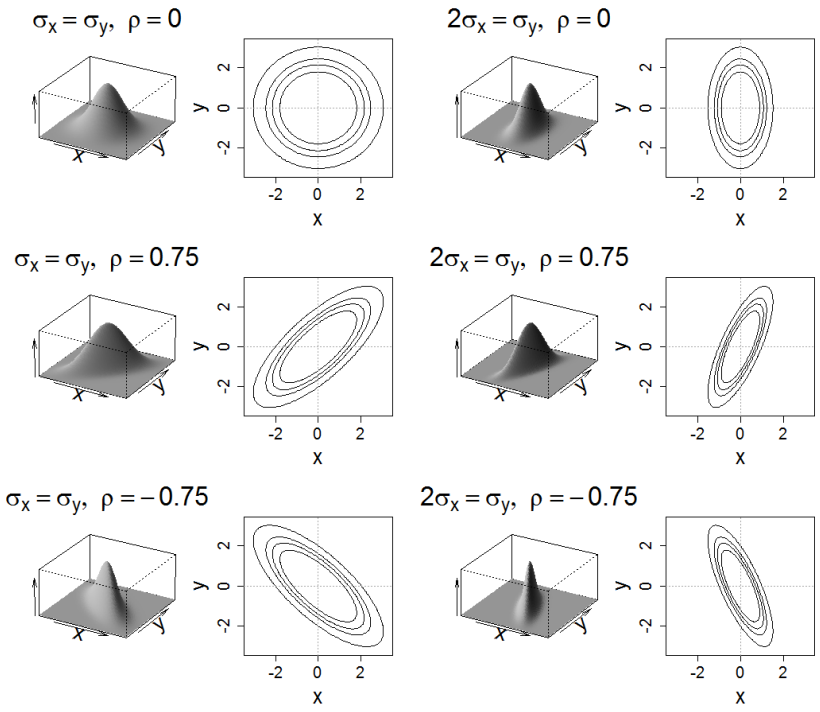
$$p(x|\mu, \mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^m |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(x-\mu)^T \mathbf{\Sigma}^{-1}(x-\mu)\right)$$

where $x \in \mathbb{R}^m$ (a vector of length m) and the mean, μ, is also a vector of length m. The variance of a univariate Gaussian, σ, is replaced by a covariance matrix of dimension $m \times m$:

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 & \dots & \rho_{1m}\sigma_1\sigma_m \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{m1}\sigma_m\sigma_1 & \rho_{m2}\sigma_m\sigma_2 & \dots & \sigma_m^2 \end{bmatrix}$$

where ρ_{ij} is the Pearson correlation of X_i and X_j. Or alternatively, the ijth element of the covariance matrix is $\text{cov}(X_i, X_j) = E\left[(X_i - \mu_i)(X_j - \mu_j)\right]$.

In two dimensions, the cross-sections of the multivariate Gaussian distribution are ellipses. The axes of the ellipses are given by the eigenvectors of the covariance matrix, $\mathbf{\Sigma}$. The first and second eigenvalues of the covariance matrix give the variance of the data along the major and minor axes of the ellipses, respectively. Some examples of bivariate normal distributions are shown below. This figure and the code that generated it can be found here: `http://www.stat.cmu.edu/~kass/KEB/RHTML/R/bivariateNormalPerspectives.r.html`



19.4.2 Maximum Likelihood Estimates

To fit data to a multivariate Gaussian distribution, we once again use our old friend, maximum likelihood estimation (Chapter 5. Here are the maximum likelihood estimates for μ and $\mathbf{\Sigma}$ for the multivariate Gaussian:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)} \qquad \hat{\mathbf{\Sigma}} = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \hat{\mu})(x^{(i)} - \hat{\mu})^T$$

19.5 Gaussian Mixture Models

A Gaussian mixture model fits a set of unlabeled data to a set of K multivariate Gaussians. There are three sets of parameters that the algorithm needs to identify:

1. $\mu_1, \ldots, \mu_K$ (the means of the Gaussians)
2. $\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_K$ (the covariance matrices of the Gaussians)
3. $\phi_1, \ldots, \phi_K$ (the mixing proportions, which must sum to one)

Question 19.6

Mixture models are examples of **generative models**, which tell a story about how the observed data were generated. Below is the actual code that generated the data for the flow cytometry example.

```
```{r Sampling from two cell lines}
cell_line_sample <- function(N) {
  mean.0 <- c(1200, 25)
  mean.1 <- c(750, 80)
  cov.0 <- matrix(c(100^2, 0.6*100*15, 0.6*100*15, 15^2), nrow=2)
  cov.1 <- matrix(c(200^2, -0.4*200*10, -0.4*200*10, 20^2), nrow=2)
  p.group.0 <- 0.4

  group <- runif(N)
  rand.samples = {}
  for(i in 1:N){
      if (group[i] < p.group.0) {
          rand.samples <- rbind(rand.samples, mvrnorm(1, mean.0, cov.0))
      } else {
          rand.samples <- rbind(rand.samples, mvrnorm(1, mean.1, cov.1))
      }
  }
  rand.samples <- as.data.frame(rand.samples)
  names(rand.samples) <- c("biomarker1", "biomarker2")
  rand.samples$group <- group < p.group.0
  rand.samples$name <- seq(1, N)
  return(rand.samples)
}
```
```

It turns out that this code matches the "story" of the Gaussian mixture model perfectly. (With real data, of course, this would not be the case.) What is that story?

Here is the algorithm for fitting a Gaussian mixture model:

1. Initialize the means μ_k, covariances $\mathbf{\Sigma}_k$, and mixing coefficients ϕ_k for all of the Gaussians $k = 1, \ldots, K$.

2. **E step.** Give each point a "voting weight" in each Gaussian equal to the probability (based on current parameter values) that it came from that Gaussian:

$$\begin{aligned} w_j^{(i)} &:= p(z^{(i)} = j | x^{(i)}, \phi, \mu, \mathbf{\Sigma}) \\ &= \frac{\phi_j \cdot \mathcal{N}(x^{(i)} | \mu_j, \mathbf{\Sigma}_j)}{\sum_{k=1}^{K} \phi_k \cdot \mathcal{N}(x^{(i)} | \mu_k, \mathbf{\Sigma}_k)} \end{aligned}$$

   Note that you will have K different voting weights for each point, and there are n points, so you need to do nK total calculations here.

3. **M step.** Re-estimate the parameters for the different Gaussians by letting each point vote in each Gaussian according to its voting weight.

$$\begin{aligned} \phi_j &:= \frac{1}{n} \sum_{i=1}^{n} w_j^{(i)} \\ \mu_j &:= \frac{\sum_{i=1}^{n} w_j^{(i)} x^{(i)}}{\sum_{i=1}^{n} w_j^{(i)}} \\ \mathbf{\Sigma}_j &:= \frac{\sum_{i=1}^{n} w_j^{(i)} (x^{(i)} - \mu_j)(x^{(i)} - \mu_j)^T}{\sum_{i=1}^{n} w_j^{(i)}} \end{aligned}$$

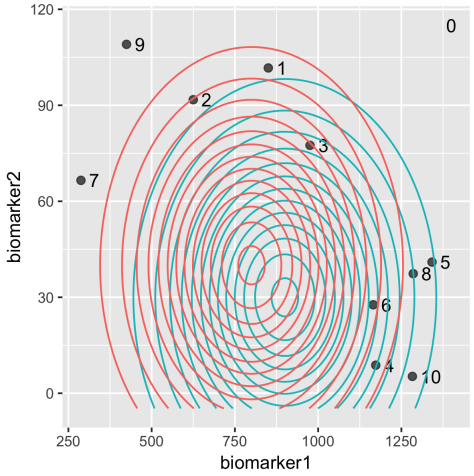
4. Check for convergence of either the parameters or the log-likelihood. If the convergence criterion is not satisfied, return to step 2.

Note that you are *not* guaranteed to get the right answer; the final Gaussians can change depending on how you initialize the model.

19.6 A Gaussian Mixture Model for Flow Cytometry Data

We will now follow the steps from the algorithm above to fit a Gaussian mixture model to the dataset from our flow cytometry example.

1. Initialize the means μ_A and μ_B, the covariances $\mathbf{\Sigma}_A$ and $\mathbf{\Sigma}_B$, and the mixing coefficients ϕ_A and ϕ_B.

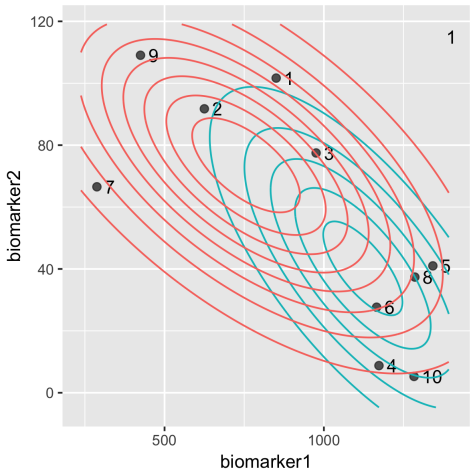


$$\phi_A = 0.5$$
$$\phi_B = 0.5$$
$$\mu_A = \begin{bmatrix} 900 \\ 30 \end{bmatrix}$$
$$\mu_B = \begin{bmatrix} 800 \\ 40 \end{bmatrix}$$
$$\mathbf{\Sigma}_A = \begin{bmatrix} 200^2 & 0 \\ 0 & 30^2 \end{bmatrix}$$
$$\mathbf{\Sigma}_B = \begin{bmatrix} 200^2 & 0 \\ 0 & 30^2 \end{bmatrix}$$

2. Do E step for round 1.

i	$x_1^{(i)}$	$x_2^{(i)}$	$\mathcal{N}(x^{(i)}\|\mu_A, \mathbf{\Sigma}_A)$	$\mathcal{N}(x^{(i)}\|\mu_B, \mathbf{\Sigma}_B)$	$w_A^{(i)}$	$w_B^{(i)}$
1	634.83	110.55	3e-07	1.2e-06	0.201	0.799
2	650.06	74.22	4.1e-06	1e-05	0.282	0.718
3	788.24	81.52	5.2e-06	1e-05	0.338	0.662
4	771.47	84.98	4e-06	8.5e-06	0.320	0.680
5	515.81	91.08	5.3e-07	2.3e-06	0.189	0.811
6	1101.23	31.05	1.6e-05	8.2e-06	0.662	0.338
7	649.32	77.05	3.5e-06	9.3e-06	0.275	0.725
8	652.89	97.16	1e-06	3.3e-06	0.234	0.766
9	1183.02	11.73	8.1e-06	2.7e-06	0.749	0.251
10	1238.45	33.46	6.3e-06	2.3e-06	0.729	0.271
sum					3.979	6.021

3. Do M step for round 1.



$$\phi_A = 0.398$$

$$\phi_B = 0.602$$

$$\mu_A = \begin{bmatrix} 947.6 \\ 53.5 \end{bmatrix}$$

$$\mu_B = \begin{bmatrix} 733.2 \\ 79.7 \end{bmatrix}$$

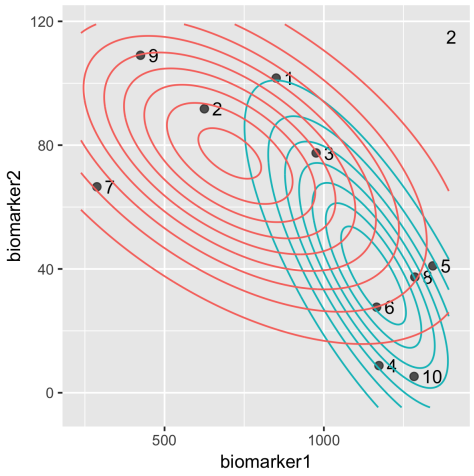
$$\Sigma_A = \begin{bmatrix} 256.6^2 & -0.925 \cdot 256.6 \cdot 32.3 \\ -0.925 \cdot 256.6 \cdot 32.3 & 32.3^2 \end{bmatrix}$$

$$\Sigma_B = \begin{bmatrix} 195.4^2 & -0.855 \cdot 195.4 \cdot 24.7 \\ -0.855 \cdot 195.4 \cdot 24.7 & 24.7^2 \end{bmatrix}$$

4. Do E step for round 2.

i	$x_1^{(i)}$	$x_2^{(i)}$	$\mathcal{N}(x^{(i)}\|\mu_A, \Sigma_A)$	$\mathcal{N}(x^{(i)}\|\mu_B, \Sigma_B)$	$w_A^{(i)}$	$w_B^{(i)}$
1	634.83	110.55	5.9e-06	1.6e-05	0.193	0.807
2	650.06	74.22	1.4e-05	3.1e-05	0.226	0.774
3	788.24	81.52	3.1e-05	5.1e-05	0.287	0.713
4	771.47	84.98	2.7e-05	4.8e-05	0.271	0.729
5	515.81	91.08	7.2e-06	2.2e-05	0.178	0.822
6	1101.23	31.05	3.9e-05	8.5e-06	0.754	0.246
7	649.32	77.05	1.7e-05	3.8e-05	0.227	0.773
8	652.89	97.16	2e-05	4.6e-05	0.219	0.781
9	1183.02	11.73	1.7e-05	1.5e-06	0.884	0.116
10	1238.45	33.46	1.4e-05	1.8e-06	0.837	0.163
sum					4.078	5.922

5. Do M step for round 2. Calculate the new log-likelihood.



$$\phi_A = 0.408$$

$$\phi_B = 0.592$$

$$\mu_A = \begin{bmatrix} 981.2 \\ 49.4 \end{bmatrix}$$

$$\mu_B = \begin{bmatrix} 706.5 \\ 83 \end{bmatrix}$$

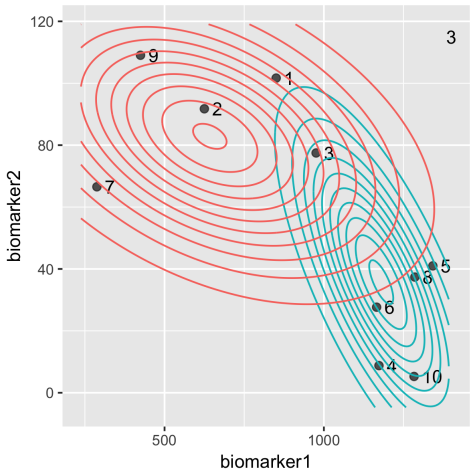
$$\Sigma_A = \begin{bmatrix} 252.6^2 & -0.924 \cdot 252.6 \cdot 32.1 \\ -0.924 \cdot 252.6 \cdot 32.1 & 32.1^2 \end{bmatrix}$$

$$\Sigma_B = \begin{bmatrix} 164.3^2 & -0.793 \cdot 164.3 \cdot 20.8 \\ -0.793 \cdot 164.3 \cdot 20.8 & 20.8^2 \end{bmatrix}$$

6. Do E step for round 3.

i	$x_1^{(i)}$	$x_2^{(i)}$	$\mathcal{N}(x^{(i)}\|\mu_A, \Sigma_A)$	$\mathcal{N}(x^{(i)}\|\mu_B, \Sigma_B)$	$w_A^{(i)}$	$w_B^{(i)}$
1	634.83	110.55	5e-06	1.9e-05	0.153	0.847
2	650.06	74.22	1.1e-05	3.8e-05	0.171	0.829
3	788.24	81.52	2.8e-05	5.9e-05	0.250	0.750
4	771.47	84.98	2.4e-05	5.6e-05	0.230	0.770
5	515.81	91.08	5.4e-06	2.7e-05	0.122	0.878
6	1101.23	31.05	4.3e-05	2.7e-06	0.917	0.083
7	649.32	77.05	1.4e-05	4.7e-05	0.171	0.829
8	652.89	97.16	1.7e-05	5.7e-05	0.167	0.833
9	1183.02	11.73	2e-05	2.1e-07	0.985	0.015
10	1238.45	33.46	1.6e-05	3.8e-07	0.965	0.035
sum					4.132	5.868

7. Do M step for round 3.



$$\phi_A = 0.413$$

$$\phi_B = 0.587$$

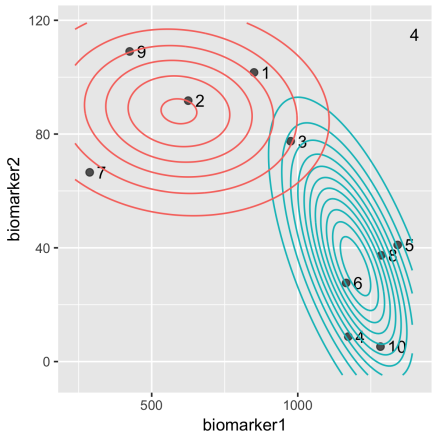
$$\mu_A = \begin{bmatrix} 1025.3 \\ 44.2 \end{bmatrix}$$

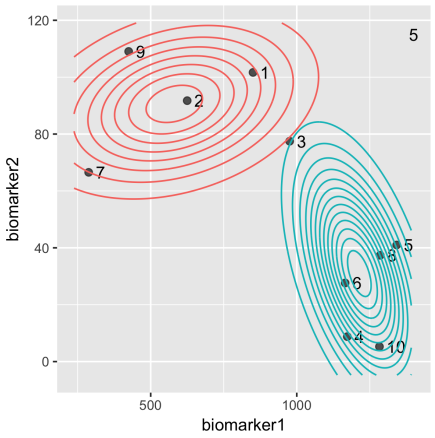
$$\mu_B = \begin{bmatrix} 672.9 \\ 87 \end{bmatrix}$$

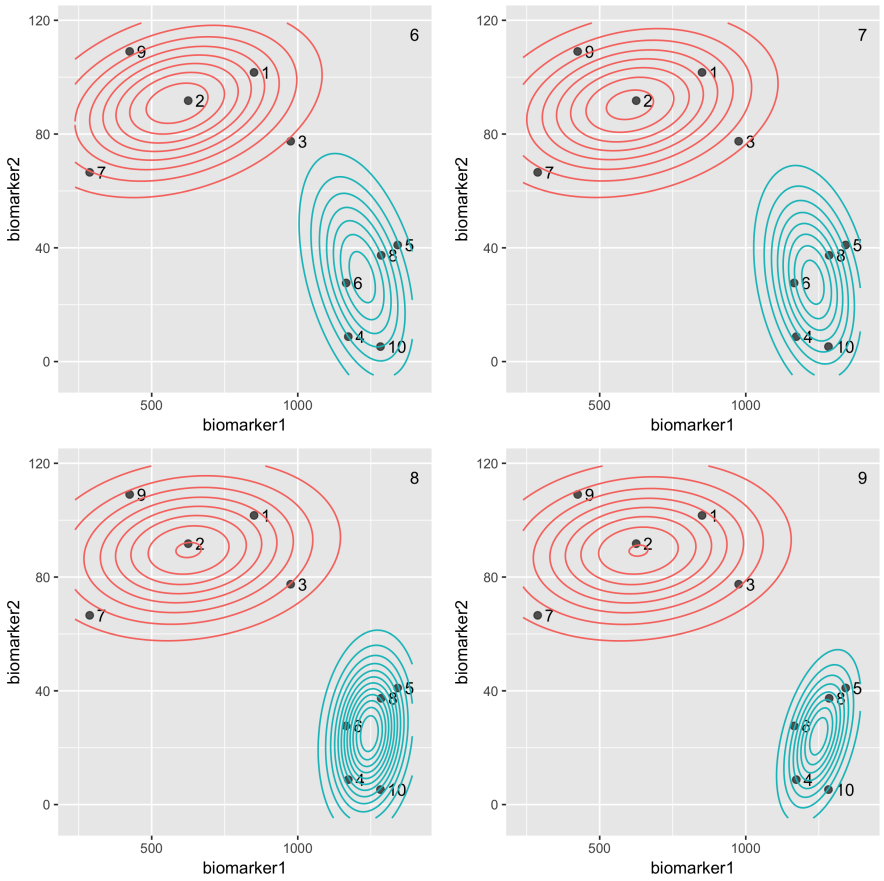
$$\mathbf{\Sigma}_A = \begin{bmatrix} 235.5^2 & -0.916 \cdot 235.5 \cdot 30.3 \\ -0.916 \cdot 235.5 \cdot 30.3 & 30.3^2 \end{bmatrix}$$

$$\mathbf{\Sigma}_B = \begin{bmatrix} 110.6^2 & -0.558 \cdot 110.6 \cdot 14.6 \\ -0.558 \cdot 110.6 \cdot 14.6 & 14.6^2 \end{bmatrix}$$

8. Do EM rounds 4 through 9.







Here is the table of calculations for the E-step after round 9:

i	$x_1^{(i)}$	$x_2^{(i)}$	$\mathcal{N}(x^{(i)}\|\mu_A, \mathbf{\Sigma}_A)$	$\mathcal{N}(x^{(i)}\|\mu_B, \mathbf{\Sigma}_B)$	$w_A^{(i)}$	$w_B^{(i)}$
1	634.83	110.55	1.8e-40	2.6e-05	0.000	1.000
2	650.06	74.22	2.5e-28	7.1e-05	0.000	1.000
3	788.24	81.52	1.6e-21	5.8e-05	0.000	1.000
4	771.47	84.98	2.5e-23	7.7e-05	0.000	1.000
5	515.81	91.08	1.7e-43	3.4e-05	0.000	1.000
6	1101.23	31.05	0.00011	6e-13	1.000	0.000
7	649.32	77.05	5e-29	9.5e-05	0.000	1.000
8	652.89	97.16	2.2e-34	0.00012	0.000	1.000
9	1183.02	11.73	0.00011	6e-18	1.000	0.000
10	1238.45	33.46	0.00011	3.7e-16	1.000	0.000
sum					3.000	7.000

The values of the final parameters are:

$$
\begin{aligned}
\phi_A &= 0.30 \\
\phi_B &= 0.70 \\
\mu_A &= \begin{bmatrix} 1174.2 \\ 25.4 \end{bmatrix} \\
\mu_B &= \begin{bmatrix} 666.1 \\ 88.1 \end{bmatrix} \\
\mathbf{\Sigma}_A &= \begin{bmatrix} 56.4^2 & -0.009 \cdot 56.4 \cdot 9.7 \\ -0.009 \cdot 56.4 \cdot 9.7 & 9.7^2 \end{bmatrix} \\
&= \begin{bmatrix} 3176.8 & -5.0 \\ -5.0 & 94.6 \end{bmatrix} \\
\mathbf{\Sigma}_B &= \begin{bmatrix} 84.8^2 & -0.287 \cdot 84.8 \cdot 11.7 \\ -0.287 \cdot 84.8 \cdot 11.7 & 11.7^2 \end{bmatrix} \\
&= \begin{bmatrix} 7185.8 & -284.8 \\ -284.8 & 137.5 \end{bmatrix}
\end{aligned}
$$

Question 19.7

Compare these parameters to the values from the code that generated the data (Question 19.6). What do you notice?

Question 19.8

Think of 2-3 different unsupervised learning problems from biology or medicine where a mixture model makes sense, conceptually at least, for modeling the data. How would you set up the mixture model in each case?

19.7 The Expectation-Maximization Algorithm

Given a joint distribution $p(x, z|\theta)$ over observed variables X and latent variables Z, governed by parameters θ, the goal of the EM algorithm is to maximize the likelihood function $p(x|\theta)$ with respect to θ. Mixture models are an

example of the EM algorithm. Here is its general form:

1. Choose an initial setting for the parameters θ.
2. **E step.** For each i, set
$$Q_i(z^{(i)}) := p(z^{(i)}|x^{(i)}, \theta)$$
3. **M step.** Set
$$\theta := \arg\max_{\theta} \sum_{i=1}^{n} \sum_{z^{(i)}} Q_i(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}|\theta)}{Q_i(z^{(i)})}$$
4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, return to step 2.

Why does this work? See Andrew Ng's lecture notes on the EM algorithm from CS229 at Stanford – they contain the clearest explanation I've seen and use the same notation as us.

19.8 Extensions

There are many different clustering algorithms. You can find a good summary of all the different ones here: https://en.wikipedia.org/wiki/Cluster_analysis. Some topics you might choose to investigate independently (or that we might look at together in the future) include:

- Hierarchical clustering (e.g. phylogenetic trees)
- Methods for choosing K (the number of clusters)
- Bayesian methods that introduce priors on the parameters in mixture models
- Biclustering

There are also many different types of mixture models that all use the EM algorithm for maximum likelihood optimization. You may want to check out the following (and many more):

- Hidden Markov Models (Baum-Welch algorithm)
- Inside-outside algorithm for induction of probabilistic context-free grammars
- Leicht and Newman's 2007 PNAS paper on finding network clusters using mixture models
- Alignment algorithms for genetic sequence data using HMMs

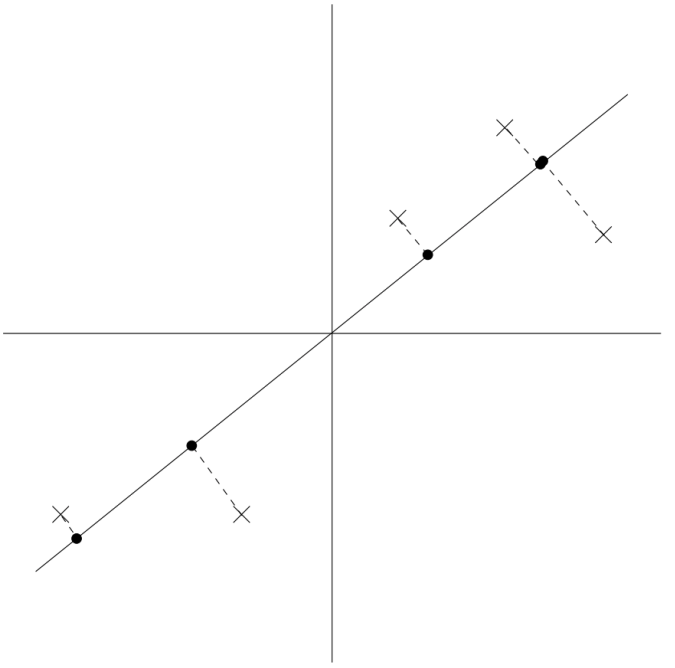
Chapter 20

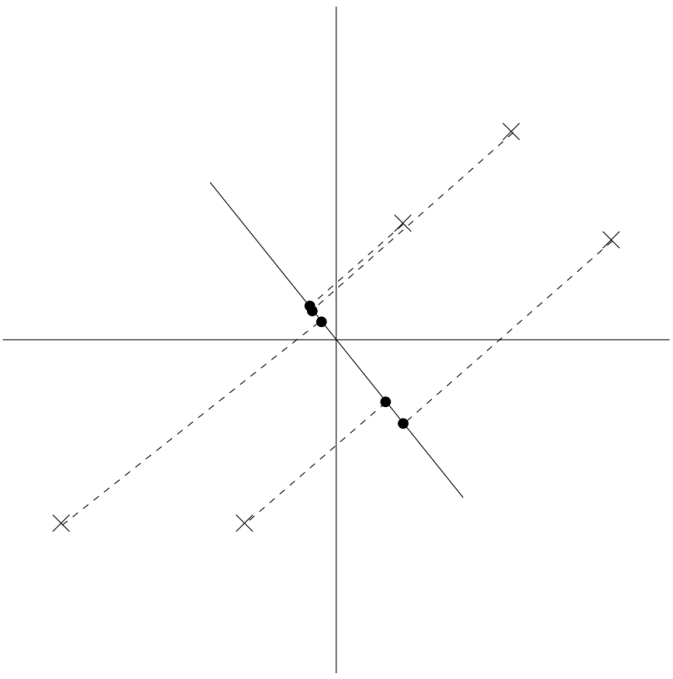
Principal Components Analysis

Clustering (Chapter 19) is one approach to uncovering latent structure in data. Another are matrix decomposition methods, the most famous of which is **principal components analysis (PCA)**. PCA is a powerful statistical tool for analyzing and visualizing datasets. It has been independently discovered several times over the course of history, and can be formulated mathematically a few different ways.

20.1 Principal Components: What are they?

You can think of PCA as a rotation of the feature space onto a set of axes that most efficiently represent the data. The **principal components** are these axes, or **basis vectors**. Here are pictures of the first two principal components for a small dataset. These were borrowed from Andrew Ng's lecture notes for CS229 at Stanford University.





Question 20.1

What do you notice about principal components 1 (top) and 2 (bottom)? Think in terms of (a) the variance (spread) of the data, and (b) the projection errors (dotted lines).

Mathematically, PCA is a linear projection of p-dimensional datapoints, $x^{(1)}, \ldots, x^{(n)}$ onto a k-dimensional space ($k \leq p$) defined by basis vectors $u_1, \ldots, u_k$ such that:

- the projection maximizes the variance from the original dataset that is retained
- the projection minimizes projection error (square loss)
- the basis vectors, $\{u_1, \ldots, u_k\}$, are orthogonal (i.e., perpendicular)

The number of distinct principal components is the smaller of the number of original variables (p) or the number of observations (n) minus one.

20.2 Definitions

20.2.1 Feature Vectors

Say you have n samples in a dataset, and each has dimensionality p. Let $x^{(i)}$ be the vector of p features for the ith person. We write

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_p^{(i)} \end{bmatrix}$$

and we can write our entire dataset as a $n \times p$ matrix like this:

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_p^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_p^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \dots & x_p^{(n)} \end{bmatrix}$$

such that the ith row of the matrix is the vector of features for the ith person. This is exactly the same notation that we have used in previous chapters.

20.2.2 Variance and Covariance

The **sample covariance** of two features x_j and x_k is given by

$$\text{cov}(x_j, x_k) = S_{jk} = \frac{1}{n-1} \sum_{i=1}^{n} \left(x_j^{(i)} - \overline{x_j} \right) \left(x_k^{(i)} - \overline{x_k} \right)$$

where the bar ($\overline{x_j}$) refers to the mean of x_j across the entire dataset. We can also write it in matrix format as

$$S = \frac{1}{n-1} \sum_{i=1}^{n} (x^{(i)} - \overline{x})(x^{(i)} - \overline{x})^T.$$

The covariance can be any real number between positive and negative infinity. One weird thing about it is that it has units - the product of the units of the two features. This means that the covariance depends on the scale that is chosen for each feature, which is somewhat arbitrary. The **sample variance** of a given feature is just the covariance in cases where $j = k$ (the covariance of a feature with itself).

The **correlation** (technically the **Pearson correlation**) is a unitless, normalized version of the covariance and has the form

$$\text{cor}(x_j, x_k) = \frac{\sum_{i=1}^{n} (x_j^{(i)} - \overline{x_j})(x_k^{(i)} - \overline{x_k})}{\sqrt{\sum_{i=1}^{n} (x_j^{(i)} - \overline{x_j})^2} \sqrt{\sum_{i=1}^{n} (x_k^{(i)} - \overline{x_k})^2}}.$$

20.2.3 Orthogonality

Two vectors are **orthogonal** if their **inner product**, or **dot product**, is zero. This is defined as

$$\langle x, y \rangle = x \cdot y = \sum_{i=1}^{n} x_i y_i$$

where the sum is over the vector components. If this mathematical notation makes you uncomfortable, just think of "orthogonality" as "perpendicularity" and you'll be able to understand it visually.

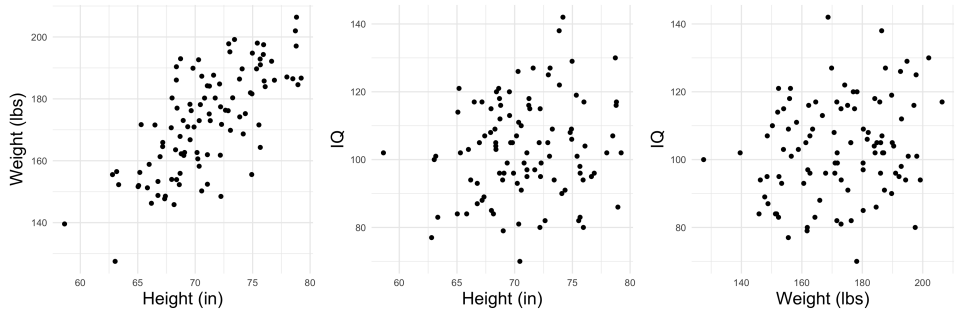
In PCA, the vectors whose inner products we are concerned with will *not* be the feature vectors for individual samples. They will be, instead, the *columns* of X – the values of a single feature for all n samples.

20.3 Example: Running PCA in R

Let's say you have measurements of weight, height, and IQ ($p = 3$) for $n = 100$ men. Here are the first 10 rows of the 100 row dataset:

Patient ID (i)	Height (in)	Weight (lbs)	IQ
1	74.9	155.6	109
2	66.5	171.6	117
3	74.4	175.2	91
4	65.8	151.3	84
5	70.2	162.7	107
6	78.8	206.4	117
7	69.1	162.7	96
8	70.6	150.3	110
9	65.2	156.2	121
10	66.8	153.3	93

Here are some scatterplots showing pairwise comparisons of the three features:



The **correlation matrix** for these data looks like this:

```
       height weight     iq
height  1.000  0.790 -0.103
weight  0.790  1.000 -0.078
iq     -0.103 -0.078  1.000
```

Question 20.2

Looking at the correlation matrix, which features are most tightly correlated? What does this imply about the direction of the first principal component, PC1?

20.3.1 Centering and Scaling

PCA is sensitive to the relative scales of the different variables in the original dataset. For this reason, datasets are usually *scaled* and *centered* before PCA is performed. All this means is that for each column (feature) of the dataset, we subtract its mean and divide by its standard deviation. The transformed column will have mean 0 and standard deviation 1.

If the data are scaled and centered, we can rewrite S (the sample covariance matrix from Section 20.2.2) as $X^T X/(n-1)$.

Question 20.3

What would happen to the principal components if you didn't center and scale the data?

Question 20.4

Why do you think the interpretation of the principal components becomes more difficult if you have features measured on lots of different scales (e.g., some categorical, some numeric/roughly normal, some numeric/highly skewed)?

20.3.2 Input and Output

In R, we can run PCA on our dataset, `d`, using the following command:

```
p <- prcomp(d[,2:4], center = TRUE, scale. = TRUE, rank. = 3)
```

where `rank.` is an optional parameter indicating the number, *k*, of principal components desired. The output looks like this:

```
> p$sdev
[1] 1.3454074 0.9899765 0.4580672

> p$rotation
              PC1         PC2         PC3
height  0.6996236 -0.09446119 -0.70823998
weight  0.6971414 -0.12698944  0.70559726
iq     -0.1565906 -0.98739595 -0.02299201

> p$center
   height    weight        iq
 70.74468 174.43346 101.12000

> p$scale
   height    weight        iq
 3.914989 13.853272 14.183701
```

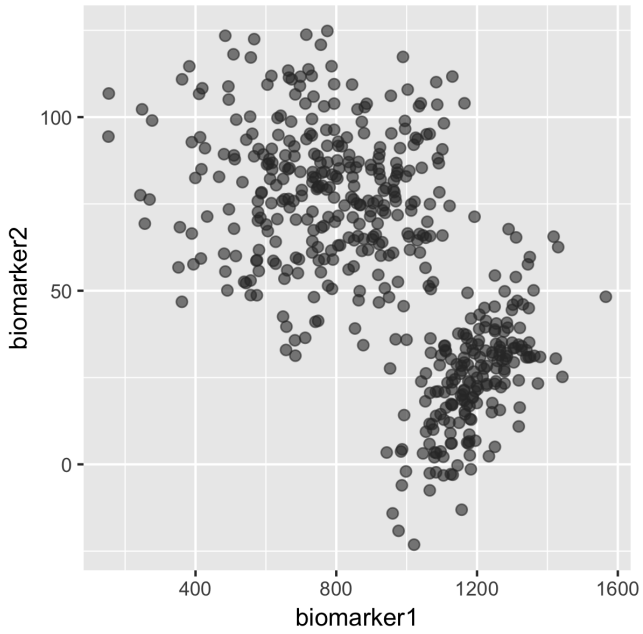
```
> p$x
              PC1         PC2         PC3
 [1,] -0.189268854  1.976180297 -0.416704298
 [2,]  1.794925031  0.710160037  0.575478783
 [3,] -0.404617344  0.283418147  0.151536071
 [4,] -0.471675025 -0.710146894  0.347006397
 [5,] -1.073877096 -0.234684815 -0.638243675
 [6,] -0.654663317  0.043481120 -0.107082277
 [7,] -1.705841441  0.368990046 -0.794047654
 [8,]  2.311353981 -0.501565781  0.033112309
 [9,] -0.006387264 -0.535422493 -1.126177513
[10,] -1.557136911  1.399023167 -0.045281059
[11,]  2.632974868 -1.046037442 -0.239934797
[12,] -0.978337630 -0.193476727  0.007814938
[13,] -0.625980903 -0.456274882 -0.305349616
(continued)
```

Question 20.5

Describe/draw the directions of the three principal component vectors, PC1, PC2, and PC3, in the coordinate system of the original predictors, height, weight, and IQ.

Question 20.6

Here is a picture of the flow cytometry dataset we first encountered in Chapter 19.



What would PC1 and PC2 look like for this dataset? (Why is there no PC3?) How could PCA help you separate the two clusters?

20.4 Applications of PCA

20.4.1 Eigenfaces

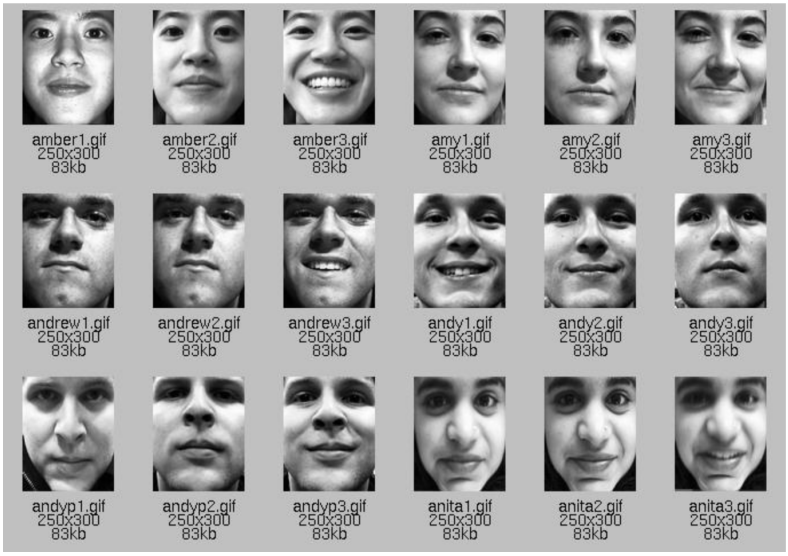
Eigenfaces were an early application of PCA to computer vision, specifically image search and retrieval. Here's what you do to create a set of eigenfaces:

1. Prepare training set of images taken under the same lighting conditions, with mouths and eyes aligned, resampled to a common pixel resolution.
2. Generate a vector for each image by concatenating the pixel intensities

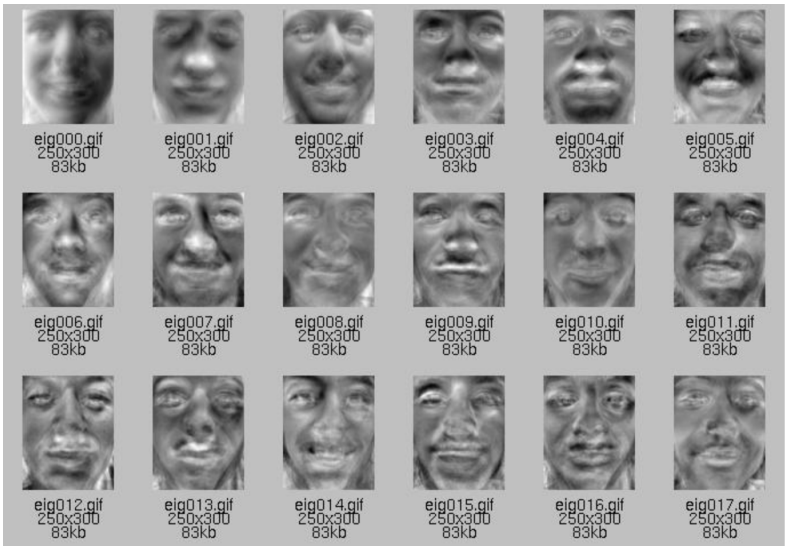
across the rows. So if the image is of dimension $r \times c$, the image vector will have $p = rc$ features.

3. Create the data matrix from the n vectors.
4. Center the data (Why wouldn't we scale the data?).
5. Calculate the eigenvectors and eigenvalues of the sample covariance matrix. Each eigenvector corresponds to one eigenface.

Some real eigenfaces are shown below [1].



[1] Figure details: (top) some of the original faces from the training set (there were 86 images total); (bottom) the eigenfaces corresponding to the 18 largest eigenvalues of the covariance matrix. From https://www.clear.rice.edu/elec301/Projects99/faces/images.html.



Question 20.7

What is X for the eigenfaces problem? What are the principal components? How could you use the principal components to match a new face to an existing database of faces?

20.4.2 Image Compression

PCA can also be used for image compression. In that case, the raw pixels for the image are the data matrix, X (so each "sample" is one row of the image and each "feature" is one column of the image). Generally the data are centered but not scaled here (Why?). A subset of the principal components corresponding to the r largest eigenvalues are used to reconstruct the image. The picture below illustrates the process[2].

[2]Figure: Using PCA for image compression. The image is recomposed by adding together many gridlike images like that shown in (a). From `https://www.projectrhea.org/rhea/index.php/PCA_Theory_Examples`.

Question 20.8

What is X for the image compression problem? What are the principal components? How does using PCA help compress the image?

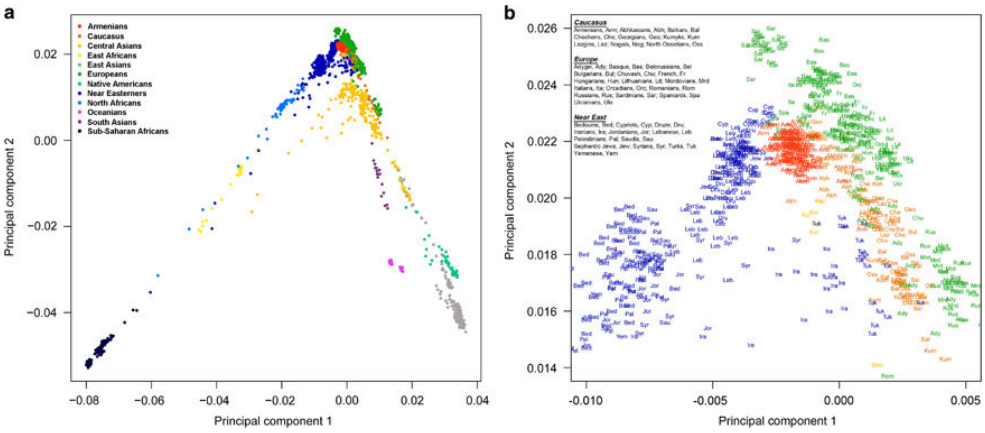
20.4.3 Genetic Ancestry

There are many awesome population genetics papers that use PCA to locate individuals within the "space" of possible genetic mutations. Here's some text from one, a figure from which is shown below[3]. Here's a quote from the original paper:

> Here, we analyse genome-wide variation in 173 Armenians and compare them with 78 other worldwide populations. We find that Armenians form a distinctive cluster linking the Near East,

[3]From European Journal of Human Genetics (2016) 24, 931-936 (2016).

Europe, and the Caucasus. We show that Armenian diversity can be explained by several mixtures of Eurasian populations that occurred between 3000 and 2000 BCE, a period characterized by major population migrations after the domestication of the horse, appearance of chariots, and the rise of advanced civilizations in the Near East. However, genetic signals of population mixture cease after 1200 BCE when Bronze Age civilizations in the Eastern Mediterranean world suddenly and violently collapsed. Armenians have since remained isolated and genetic structure within the population developed 500 years ago when Armenia was divided between the Ottomans and the Safavid Empire in Iran.



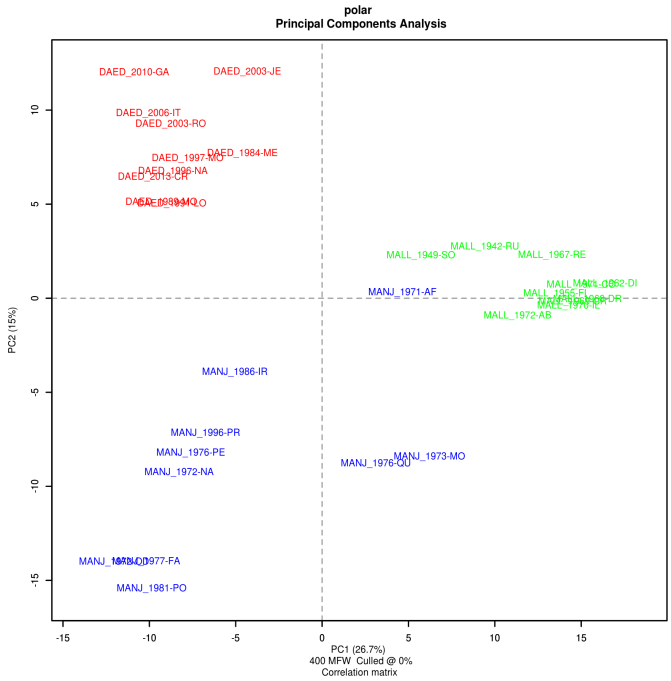
Question 20.9

What is X for the genetic ancestry problem? What are the principal components? How were the principal components used to produce the figure above?

20.4.4 Topic Modeling

PCA also has many uses in natural language processing. It forms the basis for latent semantic indexing (see Deerwester 1990) and can also be used for topic modeling, a form of mixture model (see Section 19.3)[4].

[4]Figure: Topic modeling French crime fiction. From https://dragonfly.hypotheses.org/530.



Question 20.10

What is X for the topic modeling problem? For the latent semantic indexing problem? What do the principal components represent?

Question 20.11

Think of 2-3 different unsupervised learning problems from biology or medicine where PCA makes sense, conceptually at least, for modeling the data. How would you set up the data matrix in each case? What would the principal components correspond to in the data?

20.5 Technical Details (Advanced)

20.5.1 Eigenvalues and Eigenvectors

One interpretation of matrix multiplication, Ax, where A is an $m \times m$ square matrix and x a vector of length m, is that A linearly transforms x by rotating it,

changing its magnitude, or both. An **eigenvector** of A is a vector that, when multiplied by A, grows or shrinks in magnitude but does not rotate. The multiplier of its magnitude is given by the corresponding **eigenvalue**. There are m different eigenvalues and eigenvectors for an $m \times m$ matrix, although the eigenvalues may not all be distinct. The set of all eigenvalues of A is called the **spectrum** of A.

For all eigenvalue-eigenvector pairs, the relationship $Av = \lambda v$, where v is an eigenvector and λ its corresponding eigenvalue, must hold. To find the λs and vs analytically, we can set the determinant $|A - \lambda I|$, called the **characteristic polynomial**, equal to zero and solve for the eigenvalue and eigenvector corresponding to each root. Note that any scalar multiple of an eigenvector is itself an eigenvector; usually we restrict eigenvectors to have magnitude 1 for this reason.

If a matrix is **symmetric** and **positive semidefinite** (all positive or zero eigenvalues), all of its eigenvectors will be orthogonal.

20.5.2 PCA: Eigendecomposition Version

The first way we can find the principal components is by performing an eigendecomposition identical to the one we performed in Section 20.5.1 on the sample covariance matrix. The principal components will be the eigenvectors of this matrix, and the corresponding eigenvalues tell you how much of the dataset's overall variance is accounted for by each eigenvector.

One useful fact about the covariance matrix is that, because it is a symmetric matrix, it can be diagonalized:

$$S = \frac{1}{n-1} X^T X = VLV^T \tag{20.1}$$

where V is a matrix of the p eigenvectors of S (each column is an eigenvector) and L is a diagonal matrix with the eigenvalues of S along the diagonal. This fact will be important later.

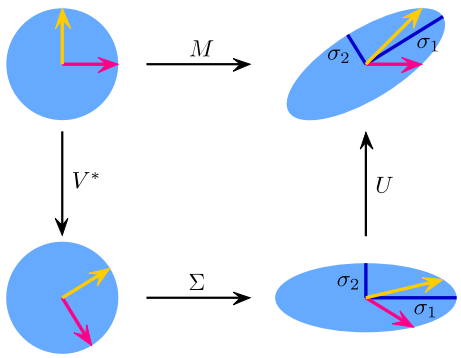
20.5.3 PCA: Singular Value Decomposition (SVD) Version

Although the eigendecomposition of the covariance matrix is generally the way PCA is presented, in practice most software uses another matrix decomposition called the **singular value decomposition (SVD)**, even though it is slower to compute, because it's more numerically stable.

Here's why the two methods are equivalent. Any matrix, M, with real or complex values (we'll focus on real valued matrices here) has an SVD of the form[5]:

$$M = UDV^T$$

where U is an orthogonal matrix, meaning that $U^TU = I$, D is diagonal and all of its nonzero elements are non-negative real numbers called **singular values**, and V is also orthogonal. For a square matrix, the SVD can be viewed as "breaking up" the transformation described by a matrix into three separate transformations: a rotation/reflection, a scaling, and another rotation/reflection[6]:



$$M = U \cdot \Sigma \cdot V^*$$

[5]This is in contrast with the eigendecomposition, which works only on positive semidefinite matrices.

[6]Figure: Graphical representation of the SVD of a square matrix, M. Note that the illustrator here uses Σ instead of D for the middle, diagonal matrix. Attribution: By Georg-Johann (Own work), via Wikimedia Commons.

If we perform the SVD on X, we obtain a decomposition

$$X = UDV^T$$

where D is a diagonal matrix with singular values $d_1, \ldots, d_p$. Knowing this, we can rewrite our covariance matrix as

$$\begin{aligned} S &= \frac{1}{n-1} X^T X \\ &= \frac{1}{n-1} (UDV^T)^T UDV^T \\ &= \frac{1}{n-1} VDU^T UDV^T \\ &= V \frac{D^2}{n-1} V^T. \end{aligned}$$

Comparing this to the form from Equation 20.1, we see that the eigenvectors correspond to the right singular values of the SVD, and the eigenvalues are related to the singular values on the diagonal of D by $\lambda_i = d_i^2/(n-1)$.

There are a lot of interpretations of the SVD. The one I like best is that the SVD provides a "nearest orthogonal matrix" to the original matrix. It's like the data have a coordinate system of orthogonal axes in which they most like to live, and the SVD tells you what that system is.