Modern Clinical Data Science

Course Notes

Bethany Percha

December 3, 2020

Contents

Chapter 1

Probability Distributions

Many of the methods we will examine in these workshops depend on basic concepts from probability theory. The following sections review these concepts and the properties of some of the most common probability distributions.

1.1 Definitions

A **probability distribution** is just a mathematical function that provides the probabilities 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.

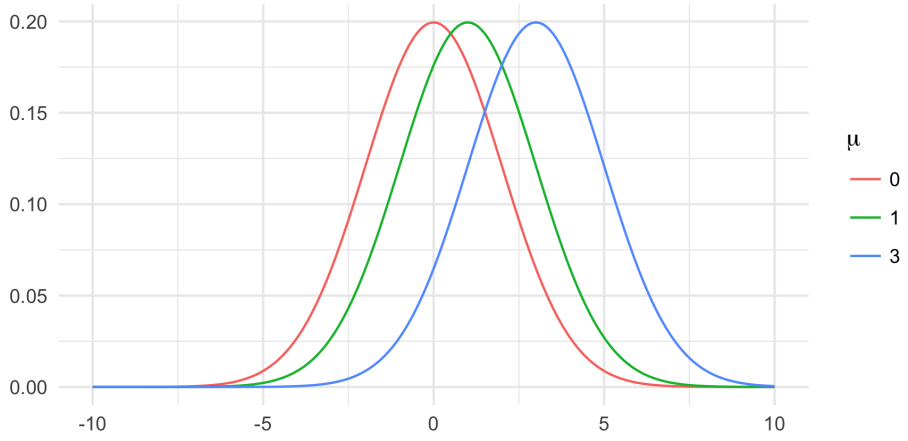
We use the notation $E[x|\theta]$ to refer to the **expected value**, or mean, of a distribution, given its parameter(s), θ. There can be more than one parameter, and it will not always be called θ; this is just an example. We use the notation $\text{var}(x|\theta)$ to refer to the variance, or spread,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

1.2 Normal Distribution

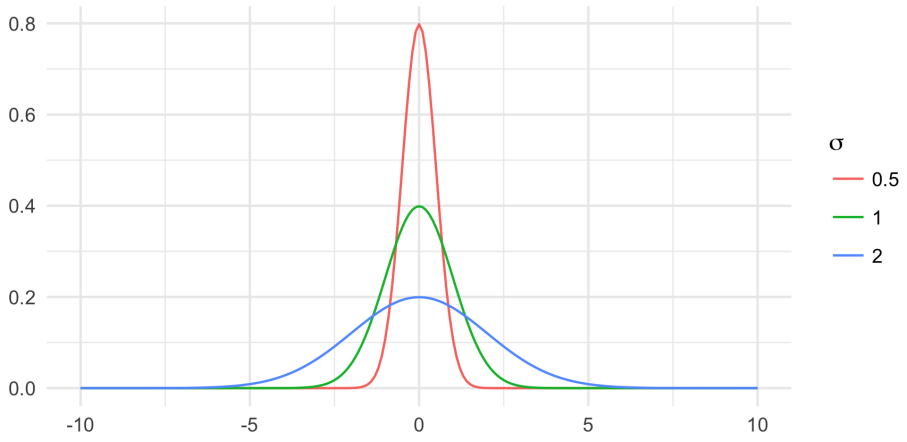
Also called the **Gaussian distribution**, the normal distribution is probably the most well-known continuous probability distribution. It has the following properties:

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

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.



1.3 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.

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 $\sum_{k=1}^{K} \phi_k = 1$.

1.4 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 $x \in \{0, \ldots, n\}$:

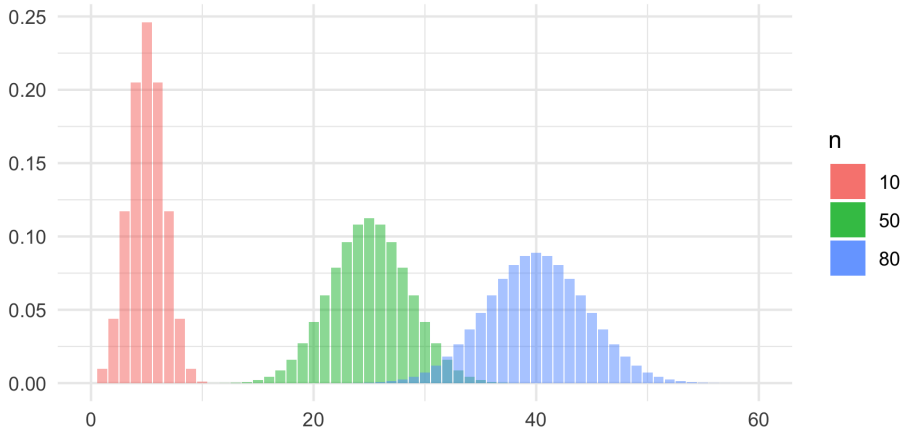
$$p(x|n,\mu) = \binom{n}{x}\mu^x(1-\mu)^{n-x} \qquad E[x|\mu] = n\mu \qquad \text{var}(x|\mu) = n\mu(1-\mu)$$

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

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

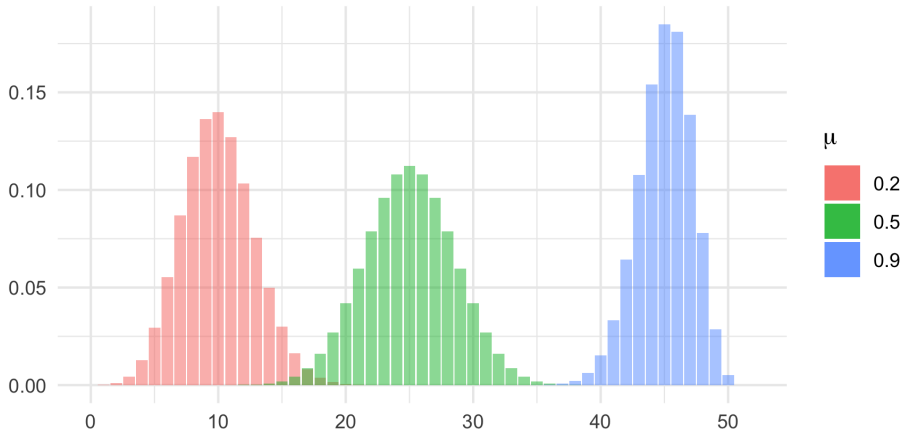
This notation denotes the number of ways it is possible to choose x things out of a group of n things, where the ordering doesn't matter. The exclamation point denotes the **factorial function**: $x! = x(x-1)(x-2)\cdots(2)(1)$.

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:

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

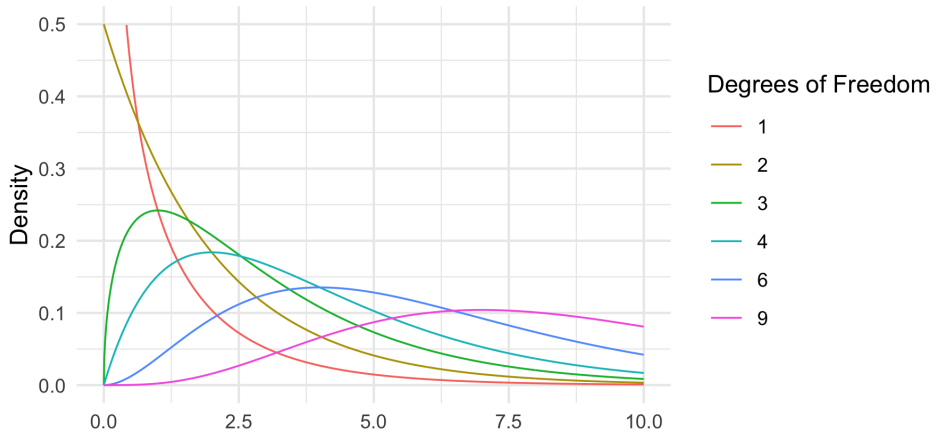


1.5 Chi-Squared Distribution

How this distribution arises:

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,

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

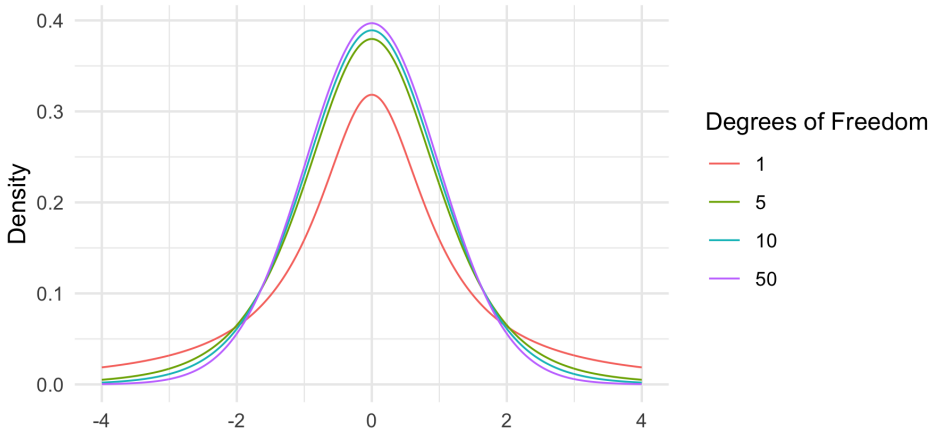
is a generalization of the factorial function to complex numbers. For any positive integer n, $\Gamma(n) = (n-1)!$.

1.6 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:

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

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

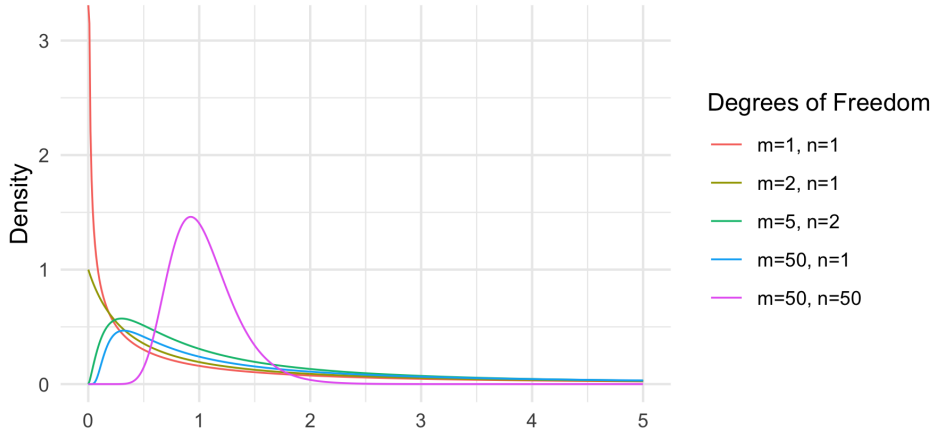
$$\text{var}(x|k) = \begin{cases} \frac{k}{k-2} & k > 2 \\ \infty & 1 < k \leq 2 \\ \text{undefined} & \text{otherwise} \end{cases}$$

1.7 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. The F-distribution looks like this:



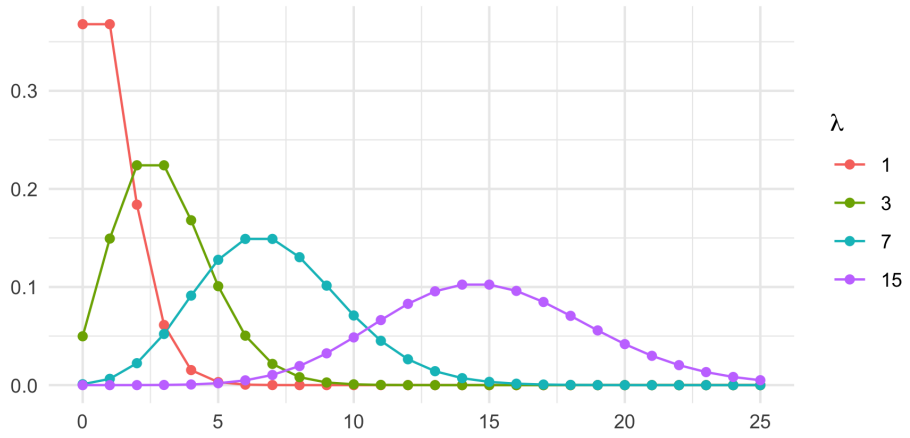
Note that if $T \sim t_k$, then $T^2 \sim F_{1,k}$. The F-distribution plays an important role in a class of statistical analysis techniques called **ANalysis Of VAriance**, or **ANOVA**.

1.8 Poisson Distribution

The **Poisson distribution** is a discrete probability distribution that is often used to model 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.

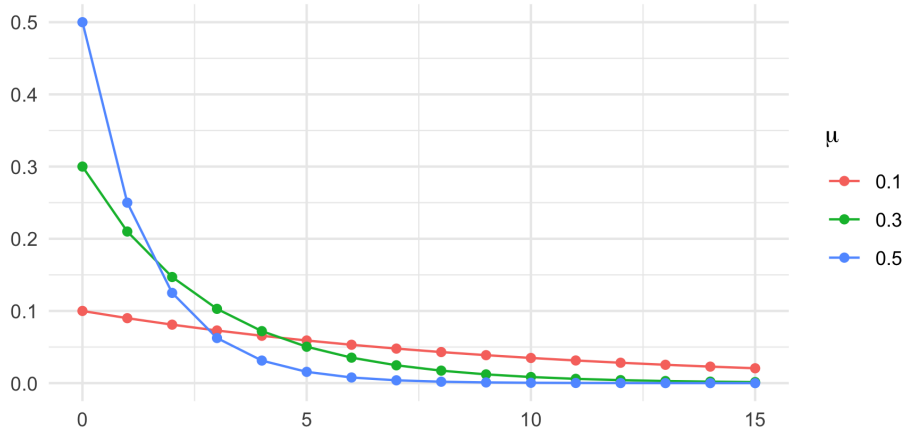


1.9 Geometric

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:



1.10 Exponential

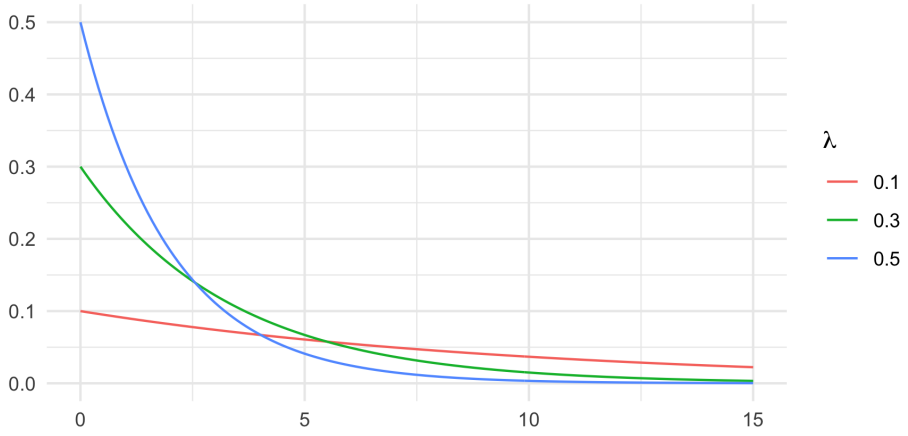
The **exponential distribution** is a continuous probability distribution that models waiting times between events that happen independently and continuously at a constant rate (Poisson process), as well as many other random variables[2]. It has the following properties:

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

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.

[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.



1.11 Summary

This is by no means an exhaustive list of all probability distributions. However, it covers those that play major roles in regression modeling and statistical hypothesis testing. The key things to remember about probability distributions are:

- They describe the relative likelihoods of different possible outcomes of a random variable.
- They are defined over the sample space of the random variable (i.e. all possible outcomes).
- They must sum/integrate to 1.0 over the whole sample space. That is, *something* must happen.

1.12 Discussion Questions

1. 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.

(b) Same situation as (a) except that instead of taking a picture each day, the patient texts you at the moment he/she observes a rash. Let $x^{(i)}$ be the time (in days) at which patient i develops a rash.

(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.

(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.

2. List five examples of random variables that follow a Bernoulli distribution.

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

Chapter 2

Maximum Likelihood Estimation

Maximum likelihood (ML) is the most widely used technique for fitting data to probability distributions. In this technique, we choose parameter value(s) for a probability distribution that maximize the probability, or **likelihood**, that it generated our observed data.

2.1 Likelihood

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

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

[1] Independent sampling just means that the values of different samples do not depend on each other.

Since the data are known but the parameter(s) θ are unknown, we typically view this thing as a function of θ:

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

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 use the logarithm of the likelihood, which we call the **log-likelihood**, because it has some nice properties, including allowing us to work with 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 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)**.

We will now go through a bunch of examples of how to find the MLEs of the probability distributions we saw in Chapter 1.

[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 estimate either way.

2.2 Bernoulli MLE

First, write down the log-likelihood.

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

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

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

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

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

2.3 Binomial MLE

Here we assume m (the number of trials) is a known quantity. First, write down the log-likelihood.

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

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

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

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

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

2.4 Normal MLE

First, 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, take the derivative of the log-likelihood with respect to μ:

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

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, take the derivative of the log-likelihood with respect to σ:

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

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

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

2.5 Poisson MLE

First, write down the log-likelihood.

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

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

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

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

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

[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 really small.

2.6 Geometric MLE

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

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

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

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

2.7 Exponential MLE

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

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

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

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

2.8 Summary

Why is maximum likelihood estimation useful? When we are fitting a regression model or summarizing some data, we often want to represent that data using a **model**, a simplified representation of the properties of the data and/or how they were generated. A probability distribution is one type of model. Maximum likelihood estimation is one way to go from data to model.

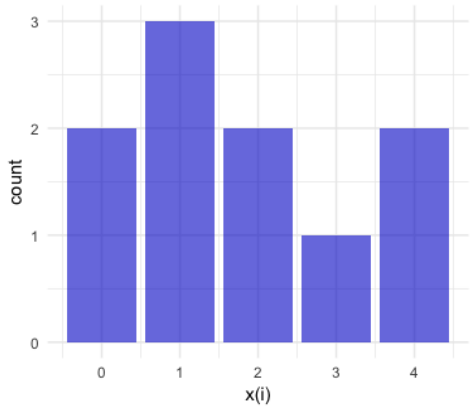
The table below contains a summary of the MLEs of various parameters from different probability distributions.

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	$\frac{n}{\sum_{i=1}^{n} x^{(i)}}$	$\mathbb{R}^+$

2.9 Discussion Questions

In Chapter 1, we examined several examples of experimental conditions and discussed which probability distribution(s) best modeled each one. Here we calculate the maximum likelihood estimates of the parameters of these distributions, based on data.

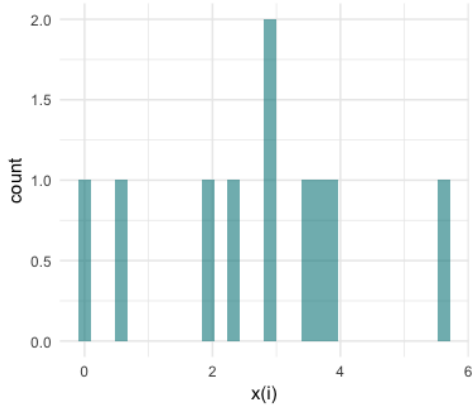
1. You are observing several patients' skin in a clinical study to see how long it takes them to develop a rash. You take a picture each day. You observe the following, where $x^{(i)}$ is the number of days of *no rash* before the rash occurs:



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

What distribution should you use to model these data? Calculate the MLE(s) for the parameter(s) of this distribution.

2. 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:

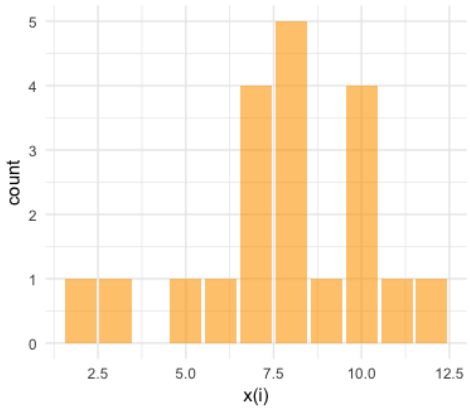


Patient ID (i)	$x^{(i)}$
1	2.25
2	3.43
3	0.68
4	0.04
5	3.78
6	5.65
7	2.88
8	3.88
9	2.83
10	1.87

What distribution should you use to model these data? Calculate the MLE(s) for the parameter(s) of this distribution.

3. Imagine you are Ladislaus Bortkiewicz, and you are modeling the number of persons killed by mule or horse kicks in the Prussian army

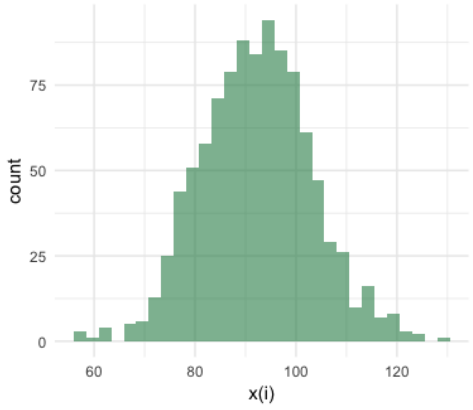
per year. You have data from the late 1800s over the course of 20 years. Let $x^{(i)}$ be the number of deaths in year i.



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

What distribution should you use to model these data? Calculate the MLE(s) for the parameter(s) of this distribution.

4. 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:



What distribution should you use to model these data? Estimate the MLE(s) for the parameter(s) of this distribution.

Chapter 3

Hypothesis Testing DRAFT

Hypothesis testing is the central idea underpinning most of the analysis you'll find in the clinical and biomedical research literature[1]. There are multiple types of hypothesis testing, but the most common type is **null hypothesis testing**, most of the theory of which originated from the statistician R.A. Fisher. In null hypothesis testing, you create a model of how your data should look under default conditions, and then you look to see whether your data deviate appreciably from the model. You quantify your data's deviation from the model by calculating a **test statistic**. One can view this type of hypothesis testing as a form of **anomaly detection**.

The statisticians Jerzy Neyman and Karl Pearson instead favored the use of hypothesis testing as a **model comparison** tool. In their view, you would set up multiple different models and then quantify each model's fit to your data to choose the best one. Fisher hated this approach because it meant accepting one or more models as truth, when in reality it's impossible to account for all potential scenarios.

Most of the hypothesis tests we use today (T-tests, chi-squared tests, etc.) follow Fisher's approach. Likelihood ratio tests and Bayesian methods adhere more to the Neyman-Pearson philosophy.

[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.

3.1 Basic Steps of a Hypothesis Test

1. *Create an initial **research hypothesis***. A hypothesis is an assertion that is capable of being proved false, such as, "If the subject has this genetic mutation, his risk of developing cancer will increase."

2. *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." For some techniques, you also need to state an **alternative hypothesis**, which is the hypothesis that is contrary to the null[2].

3. *List statistical assumptions*. E.g. in **parametric** hypothesis testing methods, we assume the data follow some particular probability distribution under the null. **Nonparametric** methods do not make this assumption.

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

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

6. *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, which is when you reject the null even if it is true (false positive result).

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

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

[2]The alternative hypothesis is the hypothesis that is contrary to the null hypothesis. It is usually taken to be that the observations are the result of a real effect (with some amount of chance variation superposed). As mentioned above, there was a huge controversy between R.A. Fisher and Jerzy Neyman/Karl Pearson over the use of alternative hypotheses. Fisher said you shouldn't use them because rejecting the null doesn't mean accepting that there's a true effect. Neyman and Pearson thought you should use them because it gave statistical tests more power. Most of hypothesis testing ended up following Fisher's approach.

[3]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.

3.2 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 $1 - \beta$.

3.3 The Z-Test

3.4 Student's T-tests

The T-test (actually a family of tests) deals with situations where you have data that are assumed to be normally distributed, and you want to draw a conclusion about the mean of that distribution.

3.4.1 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. You can do this using a test statistic

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

which follows a T-distribution with $n - 1$ degrees of freedom under the null hypothesis that the means are the same. Here $\overline{x}$ refers to the sample mean, and s refers to the sample standard deviation, which is the square root of the sample variance, s^2:

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

3.4.2 One-Sample T-test vs. Z-test

A one-sample T-test looks a lot like a Z-test (example from slides). Here is the difference:

- A Z-test assumes that the population standard deviation, σ, is fixed and known with 100% certainty so that the test statistic follows a normal distribution.
- The T-test estimates the population standard deviation from the data. The sample variance follows a chi-squared distribution with $n - 1$ degrees of freedom, where n is the sample size. In this case, the test statistic follows a Student's T-distribution with $n - 1$ degrees of freedom.

If you have enough samples, 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.

3.4.3 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 test statistic

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

where

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

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

3.4.4 Two Independent Samples, Unequal Variance

Sometimes you have two independent samples but cannot assume the variances are equal. 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

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

3.4.5 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. You want to test whether the second set of values have changed relative to the first set of values.

3.5 Mann-Whitney Test

All of these variants of the T-test make assumptions about the normality of the data. Sometimes you want to compare the means of two groups but you aren't sure whether the normal assumption holds. In this case, you might want to try a **nonparametric** alternative to the two-sample T-test called the **Mann-Whitney Test**, or Wilcoxon Rank Sum Test.

Some interesting points about nonparametric tests:

- There are no distributional assumptions.
- Most nonparametric methods replace data by their ranks, which are invariant to any monotonic transformation of the data.
- Compare this to the T-test: If you use the log-transform of the data instead of the original values and do a T-test on them, the p-value can change. But with ranks that doesn't happen.
- Replacing the data by ranks also makes the test less sensitive to outliers.

- But remember, there is no free lunch: You will always lose power relative to a parametric test if you use the nonparametric alternative in a case where the distributional assumptions of the parametric test are true.

3.6 Pearson's Chi-Squared Test

Imagine you have data on two discrete covariates for a number of different subjects. You want to test whether the value of one covariate depends on the value of the other. **Pearson's chi-squared test** is used to assess the independence of row and column values in contingency tables, provided the cell counts are high enough.

We make some assumptions:

- The data are sampled randomly from a fixed population where each member of the population has an equal probability of selection.
- 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.
- The observations are independent of each other. One observation should not be influenced in any way by the other observations taken before or after it.

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". This test statistic follows a chi-squared distribution with $(r-1)(c-1)$ degrees of freedom.

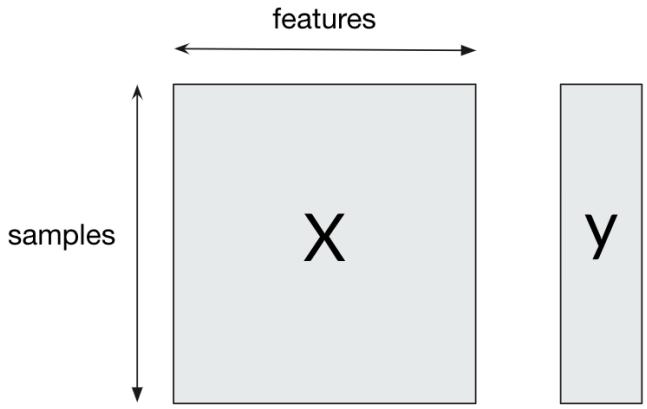
3.7 Fisher's Exact Test

For small sample sizes, the assumptions underlying Pearson's chi-squared test are no longer true. In these cases, it is common to replace the chi-squared test with something called **Fisher's Exact Test**, which calculates an exact p-value directly by considering every possible outcome.

Chapter 4

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. The general setup for supervised learning looks like this:



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.

4.1 Definitions

- **Training data:** The data used, along with an appropriate learning algorithm, to create the mapping between input and output. It is composed of **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.

4.2 Visualizing the Classification Problem

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

In the figure below, the color refers to whether a patient was admitted to the emergency room (ER) within 30 days of discharge (blue = "no", red = "yes"). The location of each point is governed by the patient's disease

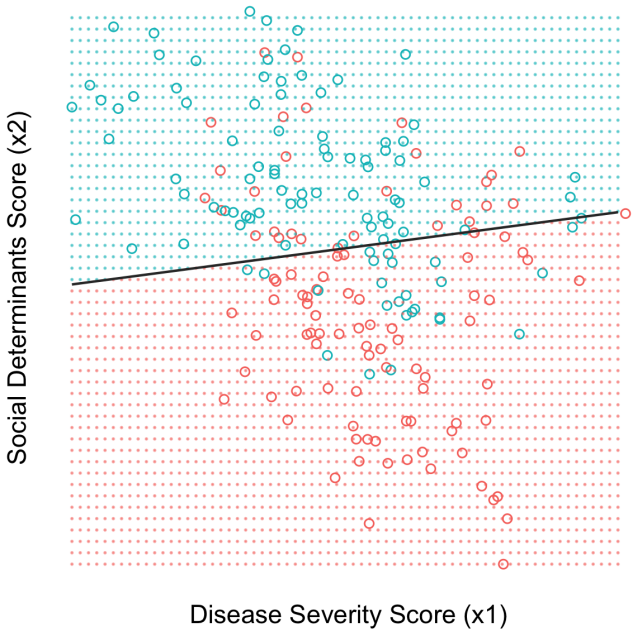
severity score (x_1, horizontal axis) and social determinants score (x_2, vertical axis).

Our goal in classification is to draw a **decision boundary** through this space, on one side of which we will predict the patient to be readmitted, and on the other side of which we will predict the patient *not* to be readmitted. The question in classification is: How do we draw a good boundary? How do we draw a boundary that will lead to accurate predictions on patients our model has never seen before?

4.3 Three Classification Algorithms

4.3.1 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.

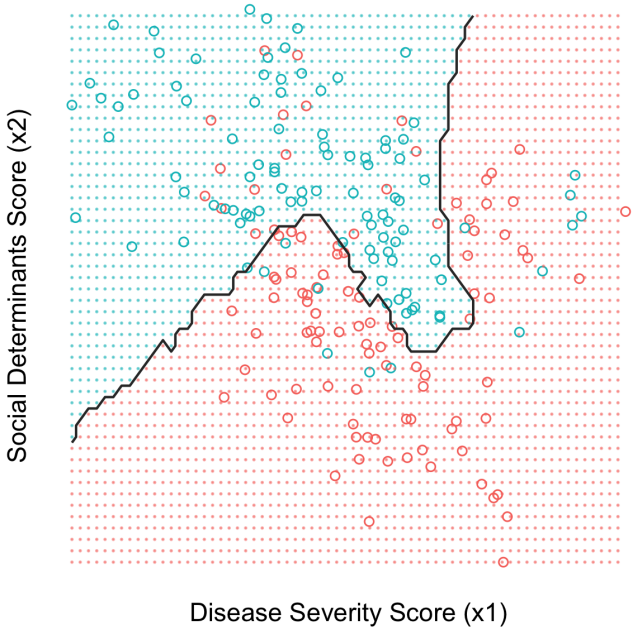


4.3.2 K Nearest Neighbors (KNN)

Another approach is to make no assumptions about the shape of the decision boundary. To make a prediction about a new patient, we simply allow the K nearest neighbors to vote. The parameter K must be set independently and is called a **hyperparameter**.

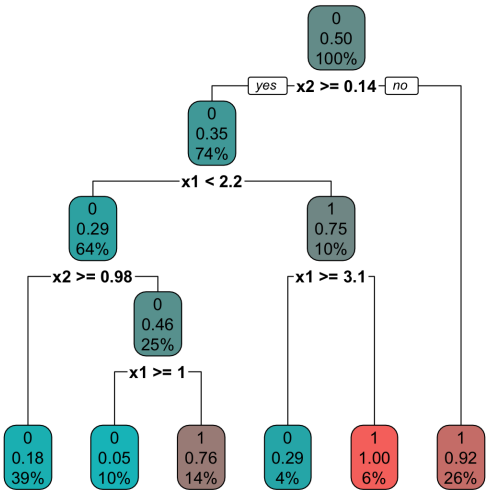
[1] In a higher-dimensional feature space, the decision boundary for logistic regression is a **hyperplane**.

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

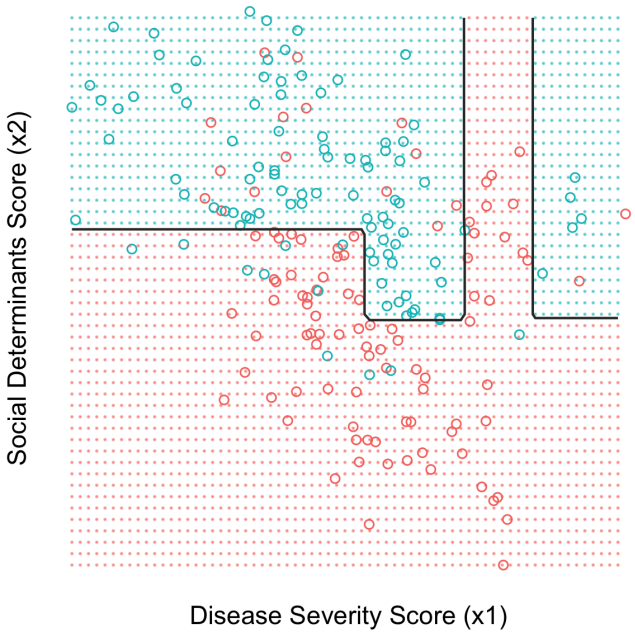


4.3.3 Decision Tree

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:

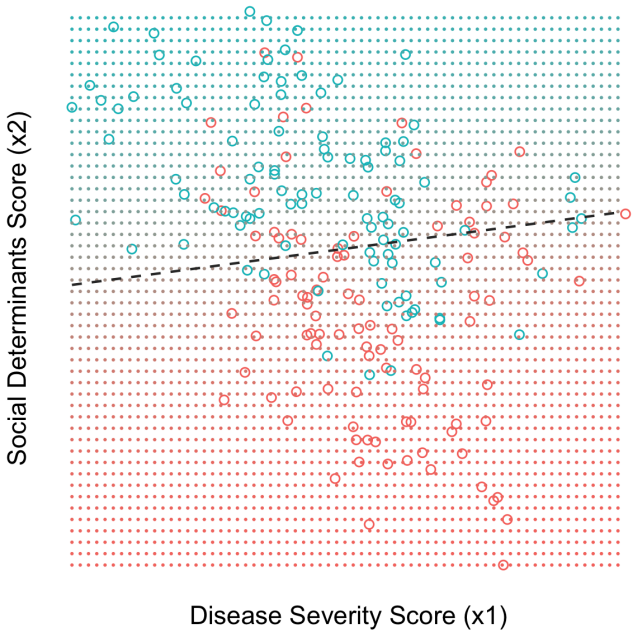


And here is the decision boundary produced by this tree:

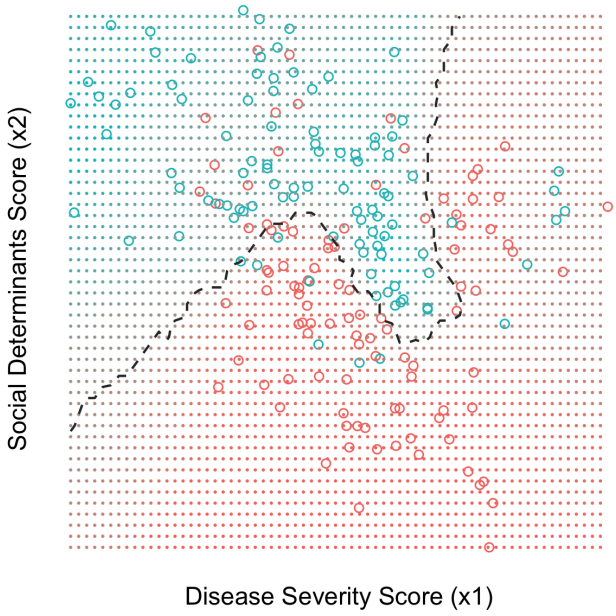


4.4 Classification with Probabilities

We can think of classification as simply drawing a decision boundary, but beneath 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, that a point belongs to the positive or negative 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 or negative:



Here is a plot of probabilities for KNN ($K = 15$):

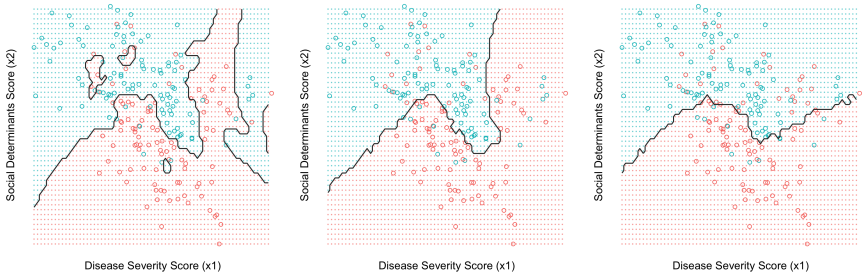


And here is a plot of probabilities for the decision tree:

4.5 Discussion Questions

Logistic regression, KNN, and decision trees are three distinct types of classification algorithms. Fed the same training data, they produce three very different-looking decision boundaries.

1. What are the advantages and disadvantages of the decision boundaries produced by:
   (a) Logistic regression?
   (b) KNN ($K = 15$)?
   (c) Decision tree?
2. What are the advantages and disadvantages of KNN with low K (e.g. $K = 3$) vs. high K (e.g. $K = 50$)? The decision boundaries for the previous example with (left to right) $K = 3, 15$, and 50 are shown below.



3. What makes a good classification algorithm?
4. The logistic regression, KNN, and decision tree algorithms can all be applied to address multi-outcome (i.e. more than two categories) classification problems with only minor modifications. Describe how each could be modified to work in these situations. (Think through this yourself before you Google.)

Chapter 5

Logistic Regression DRAFT

Chapter 6

The Bias-Variance Tradeoff
DRAFT

In classification, **model complexity** (i.e. the effective number of parameters the model must fit) is typically related to the intricacy and complexity of the decision boundary; the more parameters in the model, the more complex the boundary.

6.1 Goodness of Fit vs. Generalizability

Training vs. test error

6.2 Bias vs. Variance

This figure shows the training and test error for KNN as a function of K for a classification example similar to the one discussed in Chapter 4, as well as the training and test error for a linear model (which doesn't vary with K). You can see that the curves have characteristic shapes that vary with K. It turns out these shapes reflect a general principle for all supervised learning called the **bias-variance tradeoff**.

The bias-variance tradeoff: KNN example. The Bayes error rate, or **irreducible error**, is the probability an instance is misclassified by a classifier that knows the true class probabilities given the predictors. From *Elements of Statistical Learning*, Figure 2.4.

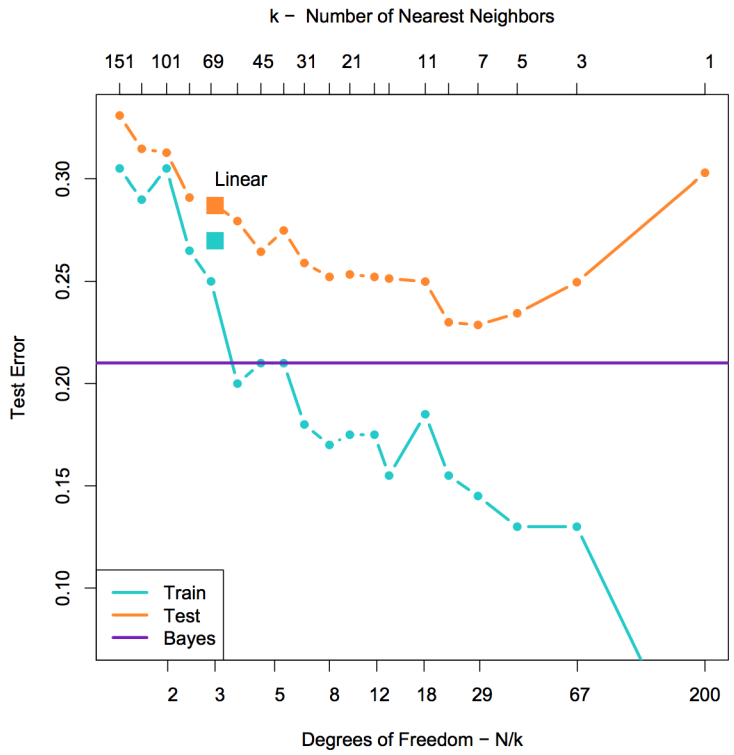
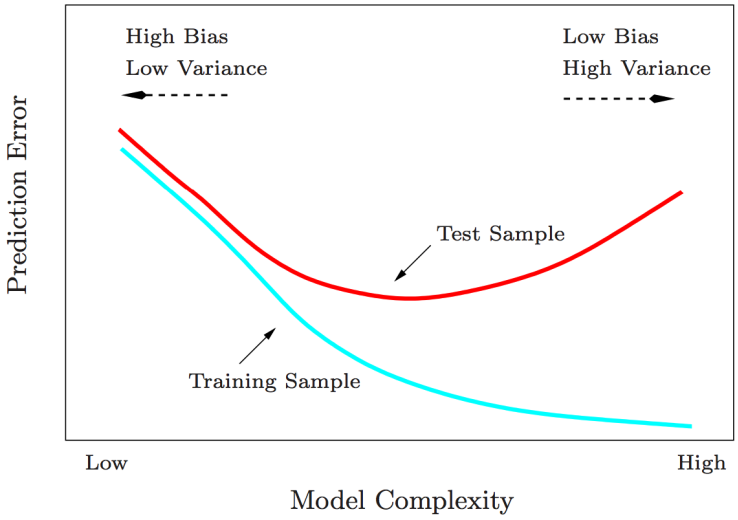
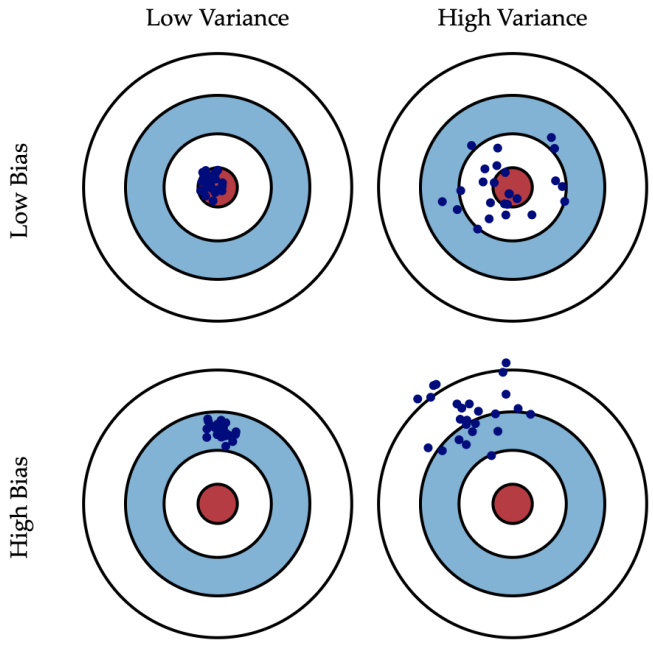


Illustration of training vs. test error as a function of model complexity, as well as the bias-variance tradeoff. From *Elements of Statistical Learning*, Figure 2.11.



A graphical illustration of the difference between bias and variance. 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". If you average their predictions, the answer is still way off the mark. In the case of high variance, the models all make very different predictions on the same training example. However, their predictions are off in random directions from the center, so if you average their outputs, you'll get closer to the right answer.



6.3 Overfitting vs. Underfitting

6.4 Discussion Questions

1. We have discussed bias and variance in the context of classification (a yes/no outcome). How would training and test error, overfitting vs. underfitting, etc. be quantified if the outcome was a number, as in a regression problem (Chapter 7)?

Chapter 7

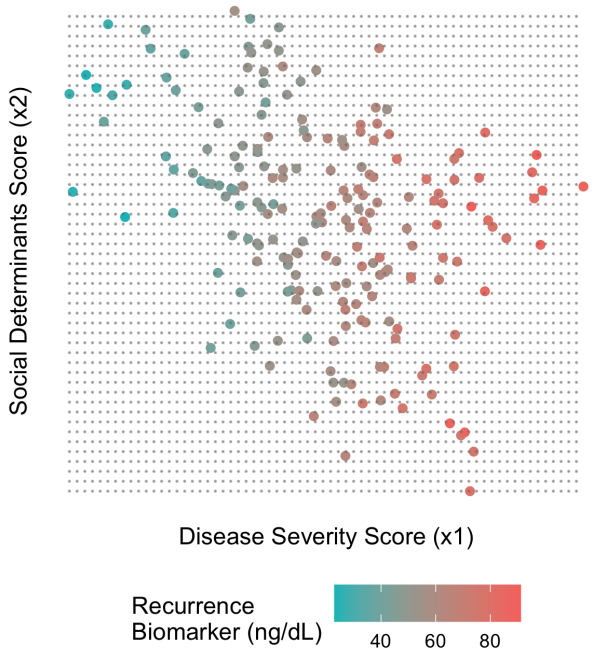
Regression

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).

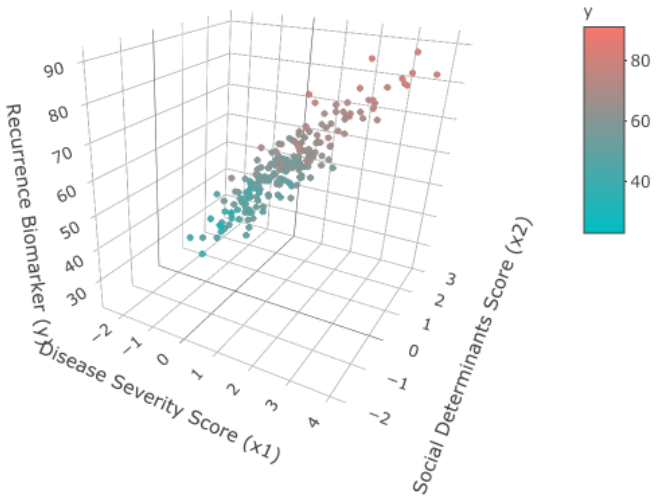
7.1 Visualizing the Regression Problem

Let's consider the same setup from Section 4.2 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 data on the same 200 patients that we examined in Section 4.2.



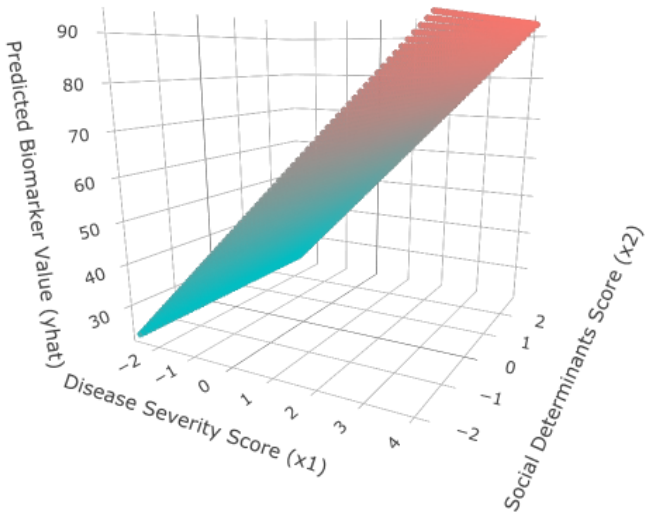
This is a plot of the data in a single plane. You can also view the data in 3D; here the color is redundant, since the height above the $x_1 \times x_2$ plane is represented separately.

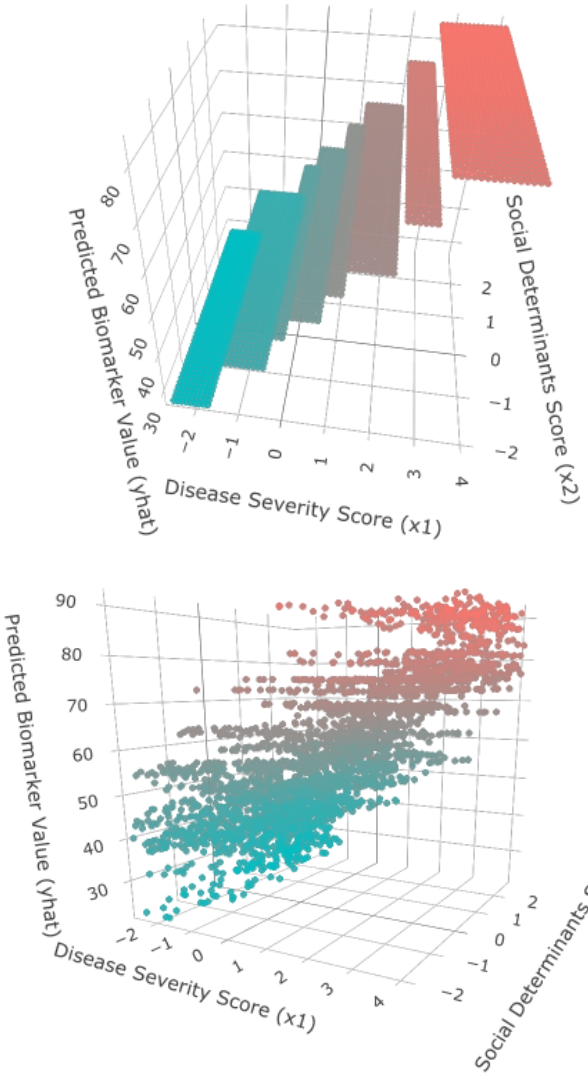


7.2 Discussion Questions

Think about the three algorithms we discussed in Chapter 4. Now think about our new task, which is to predict the *numeric value* of the recurrence biomarker as a function of the two predictors, x_1 and x_2.

1. Just looking at the two predictors, which one appears to more highly influence the value of the recurrence biomarker? Why?
2. How might you adapt KNN to deal with this problem?
3. How might you adapt a decision tree to deal with this problem?
4. How might you adapt logistic regression to deal with this problem? You'll have to "break the algorithm" a bit more this time.
5. These plots are the regression equivalents of the plots we made in Chapter 4. They need to be 3D so you can see the geometry of the predictions more clearly. One contains predictions from KNN with $K = 15$, one contains predictions from a decision tree, and one contains predictions from an algorithm called **linear regression**, which is a relative of logistic regression. Which algorithm goes with which plot?





6. What are the advantages and disadvantages of each regression algorithm?

7. How could classification be viewed as "just another form of regression"?

Chapter 8

Linear Regression DRAFT

Chapter 9

Generalized Linear Models
DRAFT

Generalized linear models (GLMs) are a class of supervised learning models that form a convenient bridge between machine learning and traditional statistics. The basic idea behind a GLM is that your outcome variable (a.k.a. response variable, see Chapter 4), y, follows a probability distribution. A linear function of the predictors (a.k.a. covariates, again see Chapter 4), $x_1, \ldots, x_p$, explains where the mean of that distribution is, but there is still some random spread around that mean due to factors that aren't modeled, plus noise.

Today, we will focus on three classes of GLM: **linear regression**, which models data where the outcome, y, is numeric ($y \in \mathbb{R}$); **logistic regression**, in which the outcome is binary ($y \in \{0, 1\}$), and **loglinear (Poisson) regression**, in which the outcome is a positive integer, or count ($y \in \{0, 1, 2, \ldots\}$).

9.1 Model Assumptions

In GLMs, the predictors can be anything – interval, ordinal, or nominal – regardless of the specific model one chooses. However, there are several

other assumptions that are important to consider before fitting one of these models:

- We assume that the outcome follows a certain type of distribution (e.g. Bernoulli distribution for a logistic regression model, normal for linear, etc.). That 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.
- We assume that the predictors are fixed and known, and thus have no error associated with their measurements (Bayesian versions of these models relax this assumption).
- We assume that the predictors enter the model as a linear combination (this is what makes these "linear models").
- We assume that the n samples in our dataset are collected independently, so that the errors of the n responses are uncorrelated. (Another way of thinking about this is that the outcomes are drawn iid from the relevant distribution.)
- We assume that the predictors are not collinear. If they are, it can lead to model instability even if the model appears to predict the outcome well.

9.2 Modeling the Outcome

9.2.1 Linear Regression

The linear regression model has a long history of development before the advent of GLMs, so it's typically taught in its own course with all of the associated model diagnostics, goodness of fit tests, etc. long before a student ever sees other GLMs. I think a comparative approach is more effective, which is why we're doing it this way.

Outcome Distribution: Normal In linear regression, we assume that the outcome, y, is normal. Recall that the normal distribution is a continuous probability distribution with the following properties:

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

where $y \in \mathbb{R}$.

9.2.2 Logistic Regression

Logistic regression models data where the outcome is binary; i.e. where y is "yes" or "no". Variants of logistic regression, called **multinomial logistic regression** and the **proportional odds model**, can also be used to model data where the outcome contains multiple categories that either have an ordering (ordinal) or do not (nominal). We will see how this works in a second.

Outcome Distribution: Bernoulli In logistic regression, we assume that the outcome, y, is either 0 or 1. We model it using the Bernoulli distribution, which is a discrete probability distribution with the following properties:

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

where $y \in \{0, 1\}$.

9.2.3 Loglinear (Poisson) Regression

In Poisson regression, the outcome is a count. This type of regression is less common than linear and logistic regression, but we include it here mainly so you can see how the ideas from GLM extend to many different classes of outcome distributions within the exponential family.

Outcome Distribution: Poisson In Poisson regression, we model the outcome using the Poisson distribution, which is a discrete probability distribu-

tion with the following properties:

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

where $y \in 0, 1, 2, \ldots$.

9.3 Modeling the Predictors

All of the models we will see today incorporate a **linear combination** of predictors. 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 denote 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 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, written as:

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

9.4 Linking the Predictors to the Outcome

All of the models we see today model the **expected value** of the outcome, $E[y]$ as some function of this linear combination of predictors. The function that relates the two is called the **link function**. Different types of regression use different link functions.

9.4.1 Linear Regression

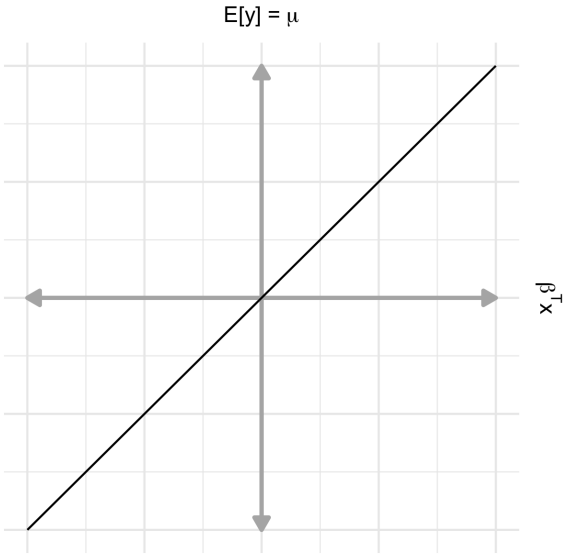
In linear regression, the mean of the outcome distribution, which is normal, can be any real number. We therefore use the **identity link**, setting $E[y]$ directly equal to the linear combination of predictors. Since the outcome is normal, we know that $E[y] = \mu$, the mean of the normal distribution. We therefore write:

$$E[y] = \mu = \beta^T x \tag{9.1}$$

which is usually rearranged and rewritten as:

$$y = \beta^T x + \varepsilon$$

where $\varepsilon \sim N(0, \sigma^2)$. The relationship between $E[y]$ and $\beta^T x$ is shown below.



9.4.2 Logistic Regression

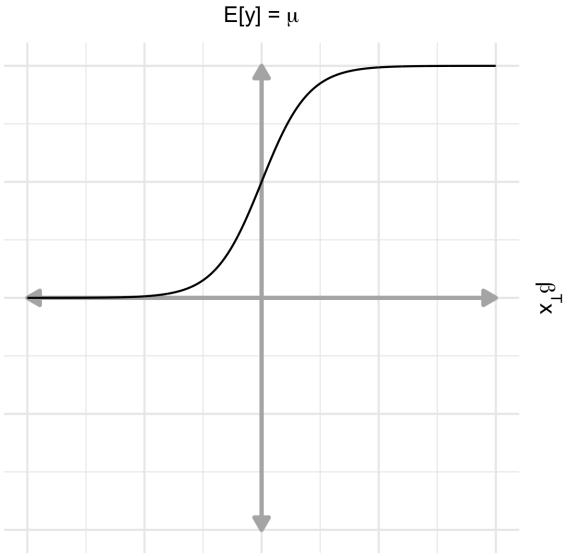
In logistic regression, the mean of the outcome distribution, which is Bernoulli, is a probability. It must therefore be a real number between 0 and 1. No matter how large or small $\beta^T x$ gets, the value of $E[y] = \mu$ cannot be outside this range. We therefore apply the **logistic function**,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

$$E[y] = \mu = \frac{1}{1 + \exp\left(-\beta^T x\right)} \tag{9.2}$$

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

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

which is the standard form of the logistic regression model. The function $\log\left(\mu/(1-\mu)\right)$ is called the logit, and in logistic regression we say we use the **logit link**.



9.4.3 Loglinear (Poisson) Regression

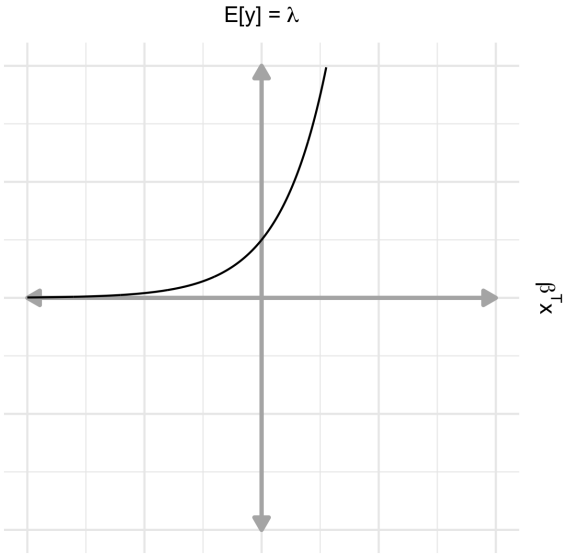
In Poisson regression, the mean of the outcome distribution, which is Poisson, is the expected value of a count. It must therefore be a real number greater than or equal to zero. In particular, no matter how small $\beta^T x$ gets, the value of $E[y] = \lambda$ cannot be negative. We therefore exponentiate $\beta^T x$ to ensure that the result is greater than zero:

$$E[y] = \lambda = \exp(\beta^T x) \tag{9.3}$$

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 we use the **log link**.



9.5 Fitting GLMs

GLMs are typically fit using maximum likelihood estimation (see Chapter 2). 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.

9.5.1 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 9.1 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: $\sum_{i=1}^{n}\left(y^{(i)} - \beta^T x^{(i)}\right)^2$.

9.5.2 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:

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

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; numerical optimization methods are used to perform the optimization.

9.5.3 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.

Chapter 10

Fitting and Interpreting GLMs DRAFT

We've seen the definition of GLMs and a lot of the math behind them in Chapter 9. Now it's time to talk about how to fit them in R and/or Python and how to interpret the model output.

Example: Linear Regression

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).

Fitting in R

To fit the model in R, you would use the `lm` command, which fits a linear regression model using ordinary least squares. Basically this package finds the analytical solution that maximizes the likelihood for linear regression, so it is taking advantage of the fact that such an analytical solution exists.

The commands for fitting the model are:

```
d <- read.csv("lecture-2-data/linear-small-cities-data.csv")
m <- lm(MORT ~ PRECIP + EDUC + NONWHITE + NOX + SO2, data = d)
summary(m)
```

The output of that last `summary` command is:

```
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
```

Fitting in Python

In Python, the `sklearn` package has a linear regression model. You fit it using the following commands:

```
from sklearn import linear_model
import pandas as pd

d = pd.read_csv("lecture-2-data/linear-small-cities-data.csv")
y = d[["MORT"]]
X = d[["PRECIP", "EDUC", "NONWHITE", "NOX", "SO2"]]
lr = linear_model.LinearRegression()
lr.fit(X, y)
lr.coef_
```

The output is somewhat less thrilling than R's:

```
array([[  1.40734044, -14.80139073,   3.19909076,  -0.10796976,
          0.35517618]])
```

but you will notice that the coefficients are identical to those found by R. There are just a lot fewer model diagnostics, etc. If you want something that more closely approximates R's output, you can use the `statsmodels` package in Python. The commands there are:

```
import statsmodels.api as sm

d = pd.read_csv("lecture-2-data/linear-small-cities-data.csv")
y = d[["MORT"]]
X = d[["PRECIP", "EDUC", "NONWHITE", "NOX", "SO2"]]
X = sm.add_constant(X)  # <- manually add the intercept
lr = sm.OLS(y, X)
results = lr.fit()
print(results.summary())
```

and the output looks like this:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   MORT   R-squared:                       0.675
Model:                            OLS   Adj. R-squared:                  0.644
Method:                 Least Squares   F-statistic:                     22.39
Date:                Tue, 17 Oct 2017   Prob (F-statistic):           4.41e-12
Time:                        22:37:41   Log-Likelihood:                -298.78
No. Observations:                  60   AIC:                             609.6
Df Residuals:                      54   BIC:                             622.1
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const        995.6365     91.641     10.865      0.000     811.907    1179.366
PRECIP         1.4073      0.689      2.042      0.046       0.026       2.789
EDUC         -14.8014      7.027     -2.106      0.040     -28.891      -0.712
NONWHITE       3.1991      0.622      5.141      0.000       1.951       4.447
NOX           -0.1080      0.135     -0.800      0.427      -0.379       0.163
SO2            0.3552      0.091      3.905      0.000       0.173       0.538
==============================================================================
Omnibus:                        2.519   Durbin-Watson:                   1.375
Prob(Omnibus):                  0.284   Jarque-Bera (JB):                1.861
Skew:                           0.136   Prob(JB):                        0.394
Kurtosis:                       3.819   Cond. No.                     1.79e+03
==============================================================================
```

Example: Logistic Regression

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.

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

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 would like to predict `LOW` based on all of the other covariates except `BWT` (why not use `BWT`?).

Fitting in R

To fit in R, we use the `glm` package, which fits GLMs using a numerical technique called Fisher scoring. The commands are:

```
d <- read.delim("lecture-2-data/logistic-lowbwt-data.tsv")
d$RACE <- as.factor(d$RACE)  # <- ensure RACE is coded as factor, not number
m <- glm(LOW ~ AGE + LWT + RACE + SMOKE + PTL + HT + UI + FTV,
         family = "binomial", data = d)
summary(m)
```

The output of this model is:

```
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
```

Fitting in Python

You can use `scikit-learn` to fit logistic regression models as well. The commands are not nearly as straightforward:

```
from sklearn import linear_model, preprocessing
import pandas as pd

d = pd.read_table("lecture-2-data/logistic-lowbwt-data.tsv", sep="\t")
```

```
y = d[["LOW"]].values.ravel()
X = d[["AGE", "LWT", "RACE", "SMOKE", "PTL", "HT", "UI", "FTV"]]
X["RACE"] = X["RACE"].astype('category')

race_recoded = pd.get_dummies(X, columns=["RACE"])
X = X[["AGE", "LWT", "SMOKE", "PTL", "HT", "UI", "FTV"]].join(
                    race_recoded.ix[:, 'RACE_2':])

lr = linear_model.LogisticRegression(C=100000.0, fit_intercept=True,
          penalty='l2', solver='liblinear', tol=0.00001)
lr.fit(X, y)
lr.coef_
```

But the end result is the same as what we got earlier in R:

```
array([[-0.02952438, -0.01542876,  0.93984778,  0.54233679,  1.864694  ,
         0.76721899,  0.06538879,  1.27277704,  0.8812321 ]])
```

where we must notice that the columns are in a different order:

```
array(['AGE', 'LWT', 'SMOKE', 'PTL', 'HT', 'UI', 'FTV', 'RACE_2', 'RACE_3'],
      dtype=object)
```

We can also do this in Python using `statsmodels`.

```
import statsmodels.api as sm

d = pd.read_table("lecture-2-data/logistic-lowbwt-data.tsv", sep="\t")
y = d[["LOW"]].values.ravel()
X = d[["AGE", "LWT", "RACE", "SMOKE", "PTL", "HT", "UI", "FTV"]]
X["RACE"] = X["RACE"].astype('category')
race_recoded = pd.get_dummies(X, columns=["RACE"])
X = X[["AGE", "LWT", "SMOKE", "PTL", "HT", "UI", "FTV"]].join(
     race_recoded.ix[:, 'RACE_2':])
X = sm.add_constant(X)  # <- manually add the intercept
lr = sm.Logit(y, X)
results = lr.fit()
print(results.summary())
```

The results look like this:

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  189
Model:                          Logit   Df Residuals:                      179
Method:                           MLE   Df Model:                            9
Date:                Wed, 18 Oct 2017   Pseudo R-squ.:                  0.1423
Time:                        07:51:27   Log-Likelihood:                -100.64
converged:                       True   LL-Null:                       -117.34
                                        LLR p-value:                 0.0001143
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.4806      1.197      0.402      0.688      -1.865       2.827
AGE           -0.0295      0.037     -0.798      0.425      -0.102       0.043
LWT           -0.0154      0.007     -2.229      0.026      -0.029      -0.002
SMOKE          0.9388      0.402      2.335      0.020       0.151       1.727
PTL            0.5433      0.345      1.573      0.116      -0.134       1.220
HT             1.8633      0.698      2.671      0.008       0.496       3.230
UI             0.7676      0.459      1.671      0.095      -0.133       1.668
FTV            0.0653      0.172      0.379      0.705      -0.273       0.403
RACE_2         1.2723      0.527      2.412      0.016       0.239       2.306
RACE_3         0.8805      0.441      1.998      0.046       0.017       1.744
==============================================================================
```

Example: Poisson Regression

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)

Fitting in R

To fit in R, we again use the `glm` package. The commands are:

```
d <- read.table("lecture-2-data/poisson-crab-data.txt", head=TRUE)
m <- glm(satell ~ color + spine + width + weight, family = "poisson", data = d)
summary(m)
```

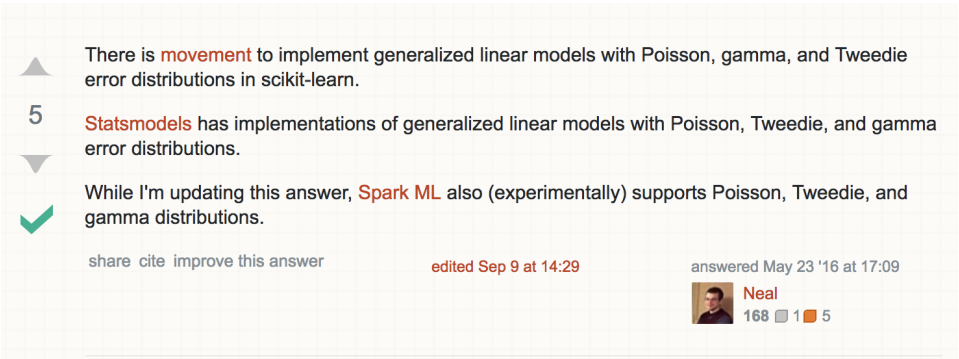
The output of this model is:

```
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
```

Fitting in Python

Right now `scikit-learn` does not allow you to use Poisson regression, but it looks like they're working on it:



For now I would suggest using `statsmodels`. Here's how you'd do Poisson regression using that package:

```
import statsmodels.api as sm

d = pd.read_table("lecture-2-data/poisson-crab-data.txt",
                  sep=r"\s*", engine="python")
y = d[["satell"]].values.ravel()
X = d[["color", "spine", "width", "weight"]]
X = sm.add_constant(X)  # <- manually add the intercept
lr = sm.Poisson(y, X)
results = lr.fit()
print(results.summary())
```

And here's the output:

Poisson Regression Results

Dep. Variable:	y	No. Observations:	173
Model:	Poisson	Df Residuals:	168
Method:	MLE	Df Model:	4
Date:	Wed, 18 Oct 2017	Pseudo R-squ.:	0.08191
Time:	07:50:24	Log-Likelihood:	-453.58
converged:	True	LL-Null:	-494.04
		LLR p-value:	1.102e-16

	coef	std err	z	P>|z|	[0.025	0.975]
const	-0.3435	0.968	-0.355	0.723	-2.242	1.555
color	-0.1849	0.067	-2.780	0.005	-0.315	-0.055
spine	0.0400	0.057	0.704	0.482	-0.071	0.151
width	0.0275	0.048	0.574	0.566	-0.066	0.121
weight	0.0005	0.000	2.865	0.004	0.000	0.001

Chapter 11

Lasso, Ridge, and Elastic Net
DRAFT

Sometimes when building regression models, you run into issues like the following:

- You have more predictors, p, than you have samples, n.
- Your predictors are highly correlated.

Both of these conditions can lead to models that are highly unstable. Maybe they fit your training data well, but if you change your training set even a tiny bit, the coefficients shift wildly. It becomes very hard to trust the coefficient values under these circumstances. One way to combat this is to introduce a **penalty** on the values of the coefficients. There are different types of penalty (see slides) that do different things. Relevant terms include: **ridge regression**, **Lasso**, and **elastic net**.

Chapter 12

Decision Trees DRAFT

Decision trees were developed as an alternative to neural networks in the 1970s. They can be used either for classification or regression. There are several algorithms for fitting decision trees, all of which are heuristic, because the general problem of learning an optimal decision tree for a dataset is NP-complete. All algorithms for tree learning are **greedy** and are not guaranteed to give the optimal solution.

12.0.1 Entropy and Information Gain

Today we will discuss the ID3 algorithm for building decision trees, which relies on the concepts of entropy and information gain. **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 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:

$$H(X) = -\sum_x P(X = x) \log_2 (P(X = x))$$

For a Bernoulli random variable, 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, as usual, is the probability the outcome is 1.

Let Y be the outcome variable of a training set. Let X be some other random variable defined over the training set. It could be one of the original predictors or some arbitrary combination of them. **Information gain** is defined as:

$$\begin{aligned}\text{Gain}(Y, X) &= H(Y) - \sum_x P(X = x)\ H(Y|X = x) \\ &= H(Y) - H(Y|X)\end{aligned}$$

It is a measure of how much our uncertainty in the value of Y is reduced by knowing X.

12.0.2 The ID3 Algorithm

Here is the algorithm:

1. Start with a single node representing the entire dataset.
2. At each current leaf node in the tree:
   (a) Compute the information gain for each feature in turn.
   (b) Split on the one with the highest information gain.
3. Return to Step 2. Stop the recursion when either the class distributions at the leaf nodes are entirely pure (all data points at a leaf have the same outcome class), or there are no more variables left to split on.

12.0.3 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 you use 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.

12.0.4 Numeric Predictors

So far we've also assumed that our predictors are discrete. But decision trees can handle numeric predictors as well. There are many different strategies for deciding on an optimal split for a predictor. Two simple ones:

- Split at the median or mean of the predictor.
- Order the datapoints on the value of the predictor and consider each possible split, looking for the one that gives the greatest information gain/standard deviation reduction. So for example, if you have a predictor called "age" and its values are $10, 11, 16, 18, 20$, and 35, consider all $N - 1 = 5$ possible split points. (This is the approach used by C4.5, a successor to ID3.)

If you have a large dataset, the second option is probably not practical, but you can downsample your dataset first and then look for the optimal cut point(s).

Chapter 13

Random Forests DRAFT

A random forest is just a collection (or **ensemble**) of decision trees whose "votes" are uncorrelated. The trees vote to produce a final prediction.

Two details are important to the construction of random forests:

1. Each tree is built using a subset of training examples sampled with replacement from the original training set. This is called **bagging** (bootstrap aggregating). Typically around 2/3 of training examples are used per tree. Note that bagging is a general-purpose procedure that can be used for other models besides random forests.

2. For each split, the tree considers not all m predictor variables, but only a randomly-chosen subset, usually of size approximately $\sqrt{m}$ (for classification problems) or $m/3$ (for regression problems). This keeps you from building the same tree over and over again and ensures that the votes from different trees are uncorrelated.

Here are two bagged samples of size 6 from the dataset in Table **??**.

ID	friends (X_1)	money (X_2)	free time (X_3)	happy (Y)
5	1	0	0	0
4	0	0	0	0
2	1	1	1	0
10	1	0	0	1
8	1	0	1	1
10	1	0	0	1

ID	friends (X_1)	money (X_2)	free time (X_3)	happy (Y)
5	1	0	0	0
6	0	0	0	0
2	1	1	1	0
5	1	0	0	0
9	0	0	1	1
7	1	2	1	1

Question 3.11: Use a random forest to fit the data from the low birth-weight example used in the logistic regression model, above. Use the following commands exactly as shown to ensure it all runs smoothly and you can view the output:

```
library(randomForest)
d <- read.delim("../data/logistic-lowbwt-data.tsv")
d$RACE <- as.factor(d$RACE)   # <- ensure RACE coded as factor
d$LOW <- as.factor(d$LOW)     # <- ensure LOW coded as factor
r <- randomForest(LOW ~ AGE + LWT + RACE + SMOKE + PTL + HT + UI + FTV, data = d, ntree = 100, do.trace = TRUE)
plot(r)
```

The random forest will report a number called the **out-of-bag (OOB)** error as it runs. To calculate OOB error, the trees are allowed to vote on the points that were *not* used in their construction. This provides an ongoing estimate of the generalization error of the algorithm, so you can see if adding more trees is likely to help.

What is the (approximate) overall OOB error? What is it for the positive outcome class only? The negative outcome class only?

Chapter 14

Boosting DRAFT

Each decision tree within a random forest provides a full model of some subset of the training data. The trees are fully grown and have low bias - most of their generalization error comes from their high variance (they overfit to details of their individual training sets). Averaging the votes from the different trees reduces this variance and increases accuracy.

There is also a different approach, called **boosting**, that uses an ensemble of *biased* learners. As more learners are added, the importance of datapoints that have been previously misclassified is upweighted so that subsequent learners will "focus on" those points. Averaging the votes from the different classifiers reduces the overall bias and increases accuracy in a way distinct from bagging/random forests.

14.0.1 AdaBoost

The first boosting algorithm was called **AdaBoost**, which is what we will look at today. Assume we have a training set

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

We will start by assuming a binary outcome, $Y \in \{1, -1\}$. The $x^{(i)}$ are feature vectors of length p. Assume we have p total classifiers (for example,

one classifier based on each feature in your training data).

1. Initialize the observation weights to $w_i = \frac{1}{N}$ for $i = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Calculate the weighted errors of the available classifiers using the current training weights, w_i. Select the classifier $G_m(x)$ that minimizes the weighted training error.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i \cdot \mathcal{I}(y^{(i)} \neq G_m(x^{(i)}))}{\sum_{i=1}^{N} w_i}$$
   (c) Compute voting weight for classifier m:
   $$\alpha_m = \log\left(\frac{1 - \text{err}_m}{\text{err}_m}\right)$$
   (d) Set
   $$w_i := w_i \cdot \exp\left[\alpha_m \cdot \mathcal{I}(y^{(i)} \neq G_m(x^{(i)}))\right]$$
   for $i = 1, \ldots, N$.
3. Output
   $$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

.

We will now apply AdaBoost to the "happiness" example.

14.0.2 Gradient Boosting

Jerome Friedman and Leo Breiman generalized AdaBoost into a general framework called **gradient boosting**. In this framework, of which AdaBoost is a subset, there are three components:

1. A loss function to be optimized.

2. Weak learners to make predictions.

3. An additive model that adds the contributions of different weak learners to minimize the loss function.

We don't have time to get into the details of the gradient boosting framework today, but its basic advantage is that it formulates the boosting process in such a way that any differentiable loss function can be used. In addition, although classification trees or regression trees are usually the weak learners (and technically, Friedman defined "gradient boosting" as a model that uses trees as learners) the framework is general enough to encompass other types of weak learners.

Chapter 15

Missing Data DRAFT

Index