Exploring Modeling with Data and Differential Equations Using R

John M. Zobitz

Version 2.0.0

Contents

Welcome

This book is written for you: the student learning about modeling and differential equations. Perhaps you first encountered models, differential equations, and better yet, building plausible models from data in your Calculus course.

This book sits "at the intersection" of several different mathematics courses: differential equations, linear algebra, statistics, calculus, data science - as well as the partner disciplines of biology, chemsitry, physics, business, and economics. An important idea is one of *transference* where a differential equation model applied in once context can also be applied (perhaps with different variable names) in a separate context.

I intentionally emphasize models from biology and the environmental sciences, but throughout the text you can find examples from the other disciplines. I hope you see the connections of this content to your own intended major.

This book is divided into 4 parts:

1. Models with differential equations
2. Parameterizing models with data
3. Stability analysis for differential equations.
4. Stochastic differential equations

Unsure what about all these topics mean? Do not worry! The topics are presented with a "modeling first" paradigm that first introduces models, and equally important, how data are used to inform a model. The "conversation" between models and data are important to help build plausibility and confidence in a model. Stability analysis helps to solidify the connection between models and parameters (which may change the underlying dynamical stability). Finally the notion of *randomness* is extended with the introduction of stochastic differential equations.

0.0.1 Changes from the first edition

The first version of this text was in 2019, and based on feedback this version was significantly revised and updated. Here is a brief summary of changes:

- Every section was revised for clarity. Where appropriate, more specific instructions was given in each exercise, with an emphasis on gradually building understanding and difficulty.
- Additional solution techniques for solving differential equations are emphasized.
- New sections on systems of nonlinear equations and visualizing randomness
- Enhanced reliance on `ggplot2` and the `tidyverse` for data wrangling and visualization. Decreased reliance on "black box" functions to do it all.

These changes added just shy of 100 (!) pages of new content. Enjoy.

Computational code

This book makes heavy use of the R programming language, and unabashedly develops programming principles using the `tidyverse` syntax and programming approach. This is intentional to facilitate direct connections to courses in introductory data science or data visualization. Throughout my years learning (and teaching) different programming languages I have found R to be the most versatile and adaptable. The `tidyverse` syntax has also been transformational for me in my own work and as my students - the barrier to compute and write code is lowered.

There is a companion R package available called `demodelr` to run programs and functions in the text. Instructions to install this package are given in Section 2. The minimum version of R Version 4.0.2 (2020-06-22) and RStudio is Version 1.4.1717.

The `demodelr` package uses the following `R` packages:

- `tidyverse` (and the associated packages) (Version 1.3.1)
- `GGally` (Version 2.1.2)
- `formula.tools` (Version 1.7.1)
- `expm` (Version 0.999-6)

There may be others that I am not aware of at the moment.

Questions? Comments? Issues?

Feel free to file an issue report: LINK

Acknowledgments

This book has been developed over the course of several years and has been written across two continents.

- **Augsburg University:** You have been my professional home for over 14 years and given me the space to be intellectually creative in my teaching. I have great colleagues to work with.
- **Augsburg University students:** Thank you for your interest in this topic, providing honest and insightful feedback about the course. This has been a work in progress (albeit bumpy at times).
- **My family:** Shannon, Colin, Grant, and Phoebe for humoring me while this project has been completed.
- **Waterparks, coffee shops, soccer practices:** Many times this was written "in the spaces" between work and home, and especially during downtimes when my kids could play. Turns out my kids love waterparks. Who knew?

Copyright

Part I

Models with Differential Equations

Chapter 1

Models of rates with data

1.1 Rates of change in the world: a model is born

The focus of this textbook is understanding *rates of change* and how you can apply them to model real-world phenomena. Additionally, this textbook focuses on *using* equations with data, building both your competence and confidence to construct a mathematical model from data and a context.

Perhaps you analyzed rates of change in calculus course when answering the following types of questions:

- If $y = xe^{-x}$, what is the derivative function $f'(x)$?
- What is the equation of the tangent line to $y = x^3 - x$ at $a = 1$?
- Where is the graph of $\sin(x)$ increasing at an increasing rate?
- What is the largest area that can be enclosed with 100 feet of fencing, with one side being along a wall?
- If you release a ball from the top of a skyscraper 500 meters above the ground, what is its speed when it impacts the ground?

The first three questions do not appear to be connected in a real-world context - but the last two questions *do* have some context. For the fencing problem, perhaps a person raises chickens and wants to care for their well-being, with a rectangular pen more aesthetically pleasing than a circular pen. In the last example the ball falling off the skyscraper assumes that acceleration of the ball is constant.

The context may reveal underlying assumptions or physical principles, which are the starting point to build a mathematical model. For the chicken coop problem the next step is to use the assumed geometry (rectangle) with the 100 feet of fencing to develop a function for the area as a function of the length of one of the sides of the pen. For the ball problem, the velocity (or the antiderivative of acceleration) can be found, from which the position function can be calculated through antidifferentiation.

Let's say we have observational data and several different (perhaps conflicting) assumptions about the context at hand, and these assumptions describe models that involve rates of change. Which model is the best one to approximate the data? The short answer: it depends. To understand why, let's take a look at a problem in context.

1.2 Modeling in context: the spread of a disease

Consider the data in Figure 1.1, which come from an Ebola outbreak in Sierra Leone in 2014. Ebola is a fatal disease so we can also consider the vertical axis in Figure 1.1 to represent total *infections* due to Ebola.

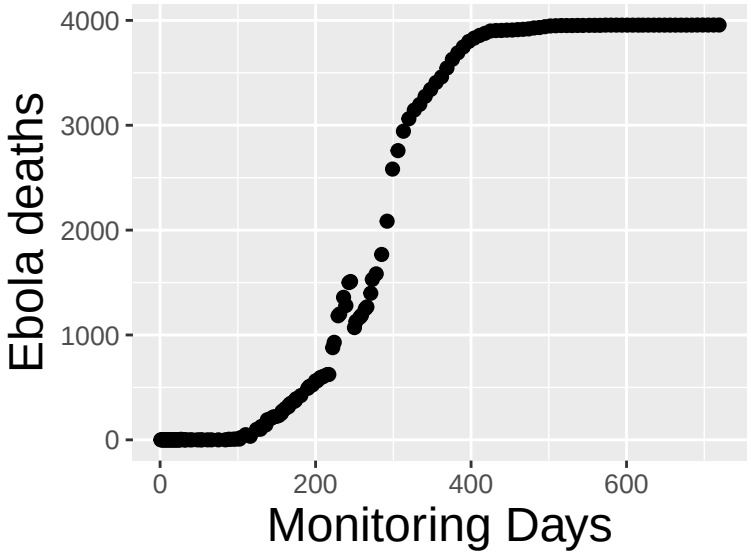


Figure 1.1: An Ebola outbreak in Sierra Leone

Constructing a model from disease dynamics is part of the field of mathematical epidemiology. How we construct a mathematical model of the spread of this outbreak largely depends on the assumptions underlying the dynamics of the disease, such as considering the rate of spread of Ebola. For our purposes here we focus on the population level (person to person) spread of Ebola. Other types of models could focus on the immune response in a single person - perhaps with a goal to design effective types of treatments to reduce the severity of infection.

Here are three initial assumptions one can make regarding the spread of Ebola:

1. The infection rate is proportional to the number of people infected.
2. The infection rate is proportional to the number of people **not** infected.
3. The infection rate is proportional to the number of infected people coming into contact with those not infected.

Let's see what each of these mathematical models would look like if we wrote down a mathematical equation. Since we are discussing *rates* of infection, this means we will need a *rate of change* or derivative. Let's use the letter I to represent the number of people that are infected.

1.2.1 Model 1: Infection rate proportional to number infected.

The first assumption states that the infection rate is proportional to the number of people infected. Translated into an equation this would be the following:

$$\frac{dI}{dt} = kI \tag{1.1}$$

In Equation (1.1) k can be thought of as a proportionality constant, with units of time^{-1} for consistency. Equation (1.1) is an example of a *differential equation*, which is just a mathematical equation with rates of change.

The *solution* to a differential equation is a function $I(t)$.[1] When we "solve'' a differential equation we determine the family of functions consistent with our rate equation. There are a lot of techniques we can use to do that, and we will examine a few later.

Back to this proportionality constant k - another term for it is a *parameter*. We can always try to solve an equation without specifying the parameter - and then if we wanted to plot a solution the parameter would also be specified. In some situations we may not be as concerned with the particular *value* of the parameter but rather its influence on the long-term behavior of the system (this is one aspect of bifurcation theory). Otherwise we can use the collected data shown above with the given model to determine the value for k. This combination of a mathematical model with data is called *data assimilation* or *model-data fusion*.

[1] You may be used to working with *algebraic equations* (e.g. solve $x^2 - 4 = 0$ for x) rather than differential equations. For algebraic equations the solution can be points (for our example, $x = \pm 2$).

Before we think about possible solutions to our differential equation, let's try to reason if the first model would be plausible. The first model assumptions states that the rate of change (the amount of increase) gets larger with the more sick people there are. While this may seem reasonable initially, it could grows quickly unreasonable when the pandemic spreads. In the case of Ebola or any other infectious disease, stringent public health measures would be enacted if the number of people infected become too large[2]. Following public health measures we would expect that the rate of infection would decrease and the number of deaths to slow. So perhaps the second model might be a little more plausible. At some point the number of people who are *not* sick will reach zero, making the rate of infection be zero (or no increase).

1.2.2 Model 2: Infection rate proportional to number NOT infected.

In this description notice how we are talking about people who are sick (which we have denoted as I) and people who are *not* sick. This looks like we might need to introduce another variable for the "not sick" people, which we will call S, or susceptible. So the differential equation we would write down would be:

$$\frac{dI}{dt} = kS \tag{1.2}$$

We are still using the parameter k as with the previous model. Also note we introduced the second variable S is in Equation (1.2). Because we have introduced another variable S we should also include a differential equation for how S changes as well. One way that we can do this is by considering our entire population as consisting of two groups of people: S and I. Infection brings someone over from S to I, which we have in Figure 1.2:

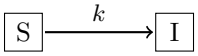


Figure 1.2: Schematic diagram for Model 1

There are three reasons why I like to use diagrams like Figure 1.2: (1) they help organize my thinking about a mathematical model (2) any assumed parameters (k) are listed, and (3) they help me to see that rates can be conserved. If I enter into the box for I, then someone is leaving S. So then the rate of change equation for S is $\frac{dS}{dt} = -kS$. Together with the differentail equation for I I have the following:

$$\begin{aligned} \frac{dS}{dt} &= -kS \\ \frac{dI}{dt} &= kS \end{aligned} \tag{1.3}$$

Equation (1.3) is an example of a *coupled differential equation*. In order to "solve" the system we need to determine functions for S and I. This coupled set of equations looks a little clunky, but there is something interest going on if we add the rates $\frac{dS}{dt}$ and $\frac{dI}{dt}$ together. Algebraically we have:

$$\frac{dS}{dt} + \frac{dI}{dt} = \frac{d(S+I)}{dt} = 0 \tag{1.4}$$

Recall from calculus that if a rate of change equals zero then the function is constant. In this case, the variable $S + I$ is constant, or we can also call $S + I = N$, the number of people in the population. This means that $S = N - I$, so we can re-write Equation (1.3) with a single equation:

$$\frac{dI}{dt} = k(N - I) \tag{1.5}$$

This second model does have some limiting behavior to this model as well. As the number of infected people reaches N (the total population size), the values of $\frac{dI}{dt}$ approaches zero, meaning I doesn't change. There is one caveat to this - if there are no infected people around ($I = 0$) *the disease can still be transmitted*, which might make not good biological sense.

[2]The COVID-19 pandemic that began in 2020 is an example of the heroic efforts of public health officials.

1.2.3 Model 3: Infection rate proportional to infected meeting not infected.

The third model rectifies some of the shortcomings of the second model (which rectified the shortcomings of the first model). The third model states that the rate of infection is due to those who are sick infecting those who are not sick. This would sort of scenario would also make some sense, as it focuses on the *transmission* of the disease between susceptibles and infected people. So if nobody is sick ($I = 0$) then the disease is not spread. Likewise if there are no susceptibles ($S = 0$), the disease is not spread as well.

In this case the diagram outlining the third model looks something like this:

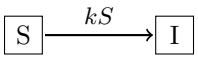


Figure 1.3: Schematic diagram for Model 3

Notice how in Figure 1.3 there is an additional S associated with k to show how the rate of infection depends on S. The differential equations that describe the scenario outlined in Figure 1.3 are the following:

$$\frac{dS}{dt} = -kSI$$
$$\frac{dI}{dt} = kSI$$

Just like before for Model 2 we can combine the two equations to yield a single differential equation:

$$\frac{dI}{dt} = k \cdot I \cdot (N - I) \tag{1.6}$$

Look's pretty similar to Model 2, doesn't it? In this case notice the variable I outside the expression, which this seems to be appropriate - if $I = 0$, then there is no increase in infection. If $I = N$ (the total population size) then there is no increase in the infection.

Let's compare these two different models graphically. For both models let's plot $\frac{dI}{dt}$ versus I, and just so we can plot let's $k = 1$ and $N = 10$ respectively. Plots of these functions are shown in Figure 1.4.

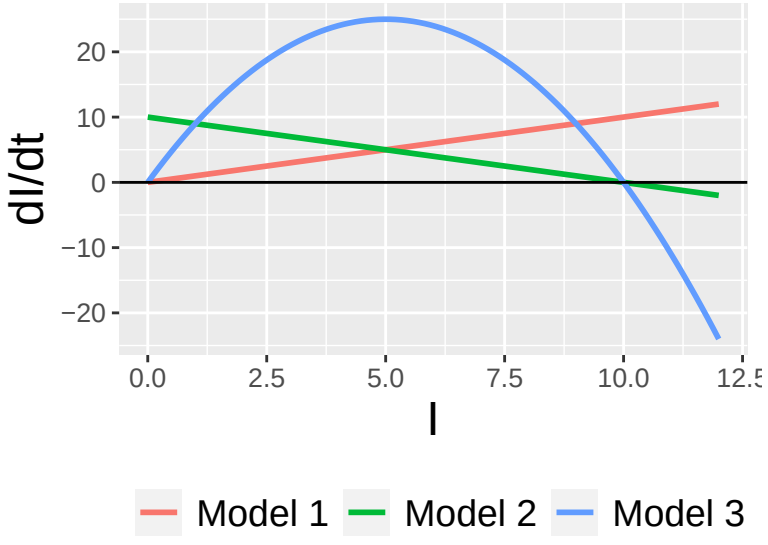


Figure 1.4: Comparing rates of change for three models

Figure 1.4 has a lot to unpack, but we can use some of our understanding of rates of change in calculus to compare the three models. Notice how the sign of $\frac{dI}{dt}$ is always positive for Model 1, indicating that the solution (I) is always increasing. For

Models 2 and 3, $\frac{dI}{dt}$ equals zero when $I = 10$, which also is the value for N After that case, $\frac{dI}{dt}$ turns negative, meaning that I is decreasing.

In summary, examining the graphs of the rates can tell a lot about the *qualitative behavior* of a solution to a differential equation even without the solution.

1.3 Model solutions

Let's return back to possible solutions (in this case formulas for $I(t)$) for our models. Usually a differential equation also has a starting or an initial value (typically at $t = 0$) that actualizes the solution. When we state a differential equation with a starting value we have an **initial value problem**. We will represent that initial value as $I(0) = I_0$, where could be considered another parameter.

With that assumption, we can (and will solve later!) the following solutions for these models:

$$\begin{aligned}
\text{Model 1 (Exponential):}\ & I(t) = I_0 e^{kt} \\
\text{Model 2 (Saturating):}\ & I(t) = N - (N - I_0)e^{-kt} \\
\text{Model 3 (Logistic):}\ & I(t) = \frac{N \cdot I_0}{I_0 + (N - I_0)e^{-kt}}
\end{aligned}$$

Notice how I assigned the names to each model (Exponential, Saturating, and Logistic). That may not mean much at the moment, but Figure 1.5 plots the three functions $I(t)$ together when $I_0 = 5$, $k = 0.03$, and $N = 4000$.

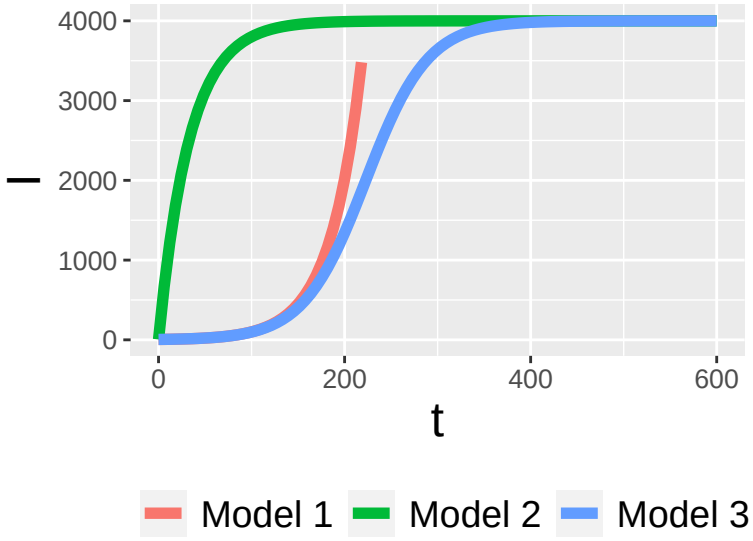


Figure 1.5: Three models compared

Notice how in Figure 1.5 Model 1 increases quickly - it actually grows without bound off the chart! Model 2 and Model 3 have saturating behavior, but it looks like Model 3 might be the one that actually captures the trend of the data.

1.4 Which model is best?

All three of these scenarios describe different modeling scenarios. With the saturating and logistic models (Models 2 and 3) we have some limiting behavior the possibility that the the rate of infecion slows. Of the two models, which one is the *best* one? Here could be some possible criteria we could evaluate:

- Do the model outputs match the data?
- (For timeseries data) are the trends accurately represented?
- Is the model easy to use?
- How will model outputs compare with newly collected measurements?

- Regarding model complexity - how many equations do we have?
- Are the number of model parameters too few or to many?

We will address several of these critera later on in this textbook when we discuss *model selection* (Section 14). Model selection is one key part of the modeling hypothesis - where we investigate the implications of a particular model analyzed. If we don't do this, we don't have an opportunity to test out what is plausible and what is believeable in our models.

1.5 Start here

In summary, it turns out that even with some initial assumptions we can very quickly build up a mathematical model to explain data. Even with these first steps we have a lot more to uncover:

- How would you determine the parameters k and N with the collected data?
- Are there other more complicated models?
- What techniques are used to determine the formulas $I(t)$?
- Are there other numerical techniques to approximate the solution $I(t)$?
- What happens to our solutions when the parameters k and N change?
- What happens to our solutions when the number of infected people change randomly for some reason?

We will study answers to these questions and more. Let's get started!

1.6 Exercises

Exercise 1.1. Solutions to an outbreak model of the flu are the following:

$$\text{Saturating model: } I(t) = 3000 - (2990)e^{-.1t}$$
$$\text{Logistic model: } I(t) = \frac{30000}{10 + (2990)e^{-.15t}},$$

where t is in days. Use these two functions to answer the following questions:

a. Plot the saturating and logistic models when $0 \leq t \leq 100$.

b. For both models, how would you describe the growth of the outbreak as t increases? How many people will be infected overall?

c. Finally, for both models evaluate $\lim_{t\to\infty} I(t)$. How do these results compare to values found on your graph?

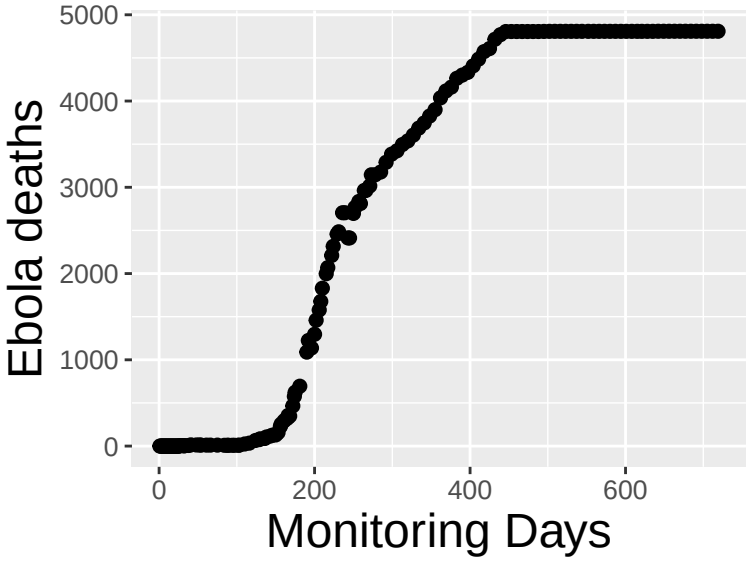


Figure 1.6: An Ebola outbreak in Liberia in 2014

Exercise 1.2. Figure 1.6 shows the Ebola outbreak for the country of Liberia in 2014. If we were to apply the logistic model (Model 3) based on this graphic what would be your estimate for N?

Exercise 1.3. The general solution for the saturating and the logistic models are:

$$\text{Saturating model: } I(t) = N - (N - I_0)e^{-kt}$$
$$\text{Logistic model: } I(t) = \frac{N \cdot I_0}{I_0 + (N - I_0)e^{-kt}},$$

where I_0 is the initial number of people infected and N is the overall population size. Using the functions from Exercise 1.1 for both models, what are N and I_0?

Exercise 1.4. The general solution for the saturating and the logistic models are:

$$\text{Saturating model: } I(t) = N - (N - I_0)e^{-kt}$$
$$\text{Logistic model: } I(t) = \frac{N \cdot I_0}{I_0 + (N - I_0)e^{-kt}},$$

where I_0 is the initial number of people infected and N is the overall population size. For both models carefully evaluate the limits to show $\lim_{t\to\infty} I(t) = N$. How do your limiting values these compare to the steady-state values you found for Models 2 and 3 in Figure 1.5, where $N = 4000$?

Exercise 1.5. A model that describes the growth of sales of a product in response to advertising is the following:

$$\frac{dS}{dt} = .55\sqrt{1-S} - S,$$

where S is the product's share of the market (scaled between 0 and 1). Use this information to answer the following questions:

a. Make a plot of the function $f(S) = .55\sqrt{1-S} - S$. for $0 \le S \le 1$.
b. Interpret your plot to predict when the market share will be increasing and decreasing. At what value is $\frac{dS}{dt} = 0$? (This is called the *steady-state* value.).
c. A second campaign is has the following differential equation:

$$\frac{dS}{dt} = .2\sqrt{1-S} - S$$

d. What is the steady-state value and how does it compare to the previous one?

Exercise 1.6. A more general form of the advertising model is

$$\frac{dS}{dt} = r\sqrt{1-S} - S, \tag{1.7}$$

where S is the product's share of the market (scaled between 0 and 1). The parameter r is related to the effectiveness of the advertising (between 0 and 1).

a. Solve $\frac{dS}{dt} = r\sqrt{1-S} - S$ for the steady state value (where $\frac{dS}{dt} = 0$). Your final answer should be expressed as a function $S(r)$.
b. Make a plot of the steady state value as a function of r, where $0 \le r \le 1$.
c. Based on your plot, what can you conclude about the steady state value as the effectiveness of the advertising increases?

Exercise 1.7. A common saying is "You are what you eat."" This saying is mostly true and can be related in a mathematical model! Here's how: an equation that relates a consumer's nutrient content (denoted as y) to the nutrient content of food (denoted as x) is given by:

$$y = cx^{1/\theta}, \tag{1.8}$$

where $\theta \ge 1$ and c are both constants is a constant. Units on x and y are expressed as a proportion of a given nutrient (such as nitrogen or carbon). Let's start with an example when $c = 1$ and $\theta = 1$. Our function then is $y = x$. In this case the point $(0.05, 0.05)$ would say that if an animal ate food that was 5% nitrogen, their body composition would be 5% as well.

a. Now assume that $c = 1$. How does the nutrient content of the consumer compare to the food when $\theta = 2$? Draw a sample curve and interpret it, contrasting it to when $\theta = 1$.
b. Now assume that $c = 1$. How does the nutrient content of the consumer compare to the food when $\theta = 5$? Draw a sample curve and interpret it, contrasting this curve to the previous two.
c. What do you think will happen when $\theta \to \infty$? Draw some sample curves to help illustrate your findings.

Exercise 1.8. A model for the outbreak of a cold virus assumes that the rate people get infected is proportional to infected people contacting susceptible people, as with Model 3 (the Logistic model). However people who are infected can also recover and become susceptible again with rate α. Construct a diagram similar Model 3 for this scenario and also write down what you think the system of differential equations would be.

Exercise 1.9. A model for the outbreak of the flu assumes that the rate people get infected is proportional to infected people contacting susceptible people, as in Model 3. However people also account for recovering from the flu, denoted with the variable R. Assume that the rate of recovery is proportional to the number of infected people with parameter β. Construct a diagram similar to Figure 1.3 for this scenario and also write down what you think the system of differential equations would be.

Exercise 1.10. Organisms that live in a saline environment biochemically maintain the amount of salt in their blood stream. An equation that represents the level of S in the blood is the following:

$$\frac{dS}{dt} = I + p \cdot (W - S),$$

where the parameter I represents the active uptake of salt, p is the permeability of the skin, and W is the salinity in the water. Use this information to answer the following questions:

a. What is that value of S at *steady state*, or when $\dfrac{dS}{dt} = 0$? Your final answer should be a function $S(I, p, W)$.
b. Witn the steady state solution, Use your what parameters (I, p, or W) cause the steady state value S to increase?

Exercise 1.11. The immigration rate of bird species (species per time) from a mainland to an offshore island is $I_m \cdot (1 - S/P)$, where I_m is the maximum immigration rate, P is the size of the source pool of species on the mainland, and S is the number of species already occupying the island. Additionally the extinction rate is $E \cdot S/P$, where E is the maximum extinction rate. The growth rate of the number of species on the island is the immigration rate minus the extinction rate.

a. Make representative plots of the of the immigration and the extinction rates as a function of S. You may set I_m, P, and E all equal to 1.
b. Determine the number of species for which the net growth rate is zero, or the number of species is in equilibrium. Express your answer as S as a function of I_m, P, and E.
c. Suppose that two islands of the same size are at different distances from the mainland. Birds arrive from the source pool and they have the same extinction rate on each island. However the maximum immigration rate is larger for the island farther away. Which of the two islands will have the larger number of species at equilibrium?

Exercise 1.12. This problem relates to animal size and volume. Assume that an animal assimilates nutrients at a rate R proportional to its surface area. Also assume that it uses nutrients at a rate proportional to its volume. You may assume that the size of the animal is implicitly a function of the nutrient intake and usage. Determine the size of the animal if its intake and use rates were in balance (meaning R is set to zero), assuming the animal is the following shapes:

a. A sphere (assume size is measured with radius r) *Note:* first determine the geometric formulas for surface area and volume.
b. A cube (assume size is measured with length l)

Hint: For both of these problems your goal is to determine a numeric value of r and l.

Chapter 2

Introduction to R

The primary tool we have to analyze models will be `R` and `RStudio`, which are commonly used for scientific and statistical computations. This is an exciting program and powerful program to learn! Admittedly learning a new software may be challenging, however I think it is worth it. With `R` you will have enormous flexibility to efficiently utilize data, design effective visualizations, and process statistical models.

2.1 R and RStudio

First let's talk terminology. The program `RStudio` is called an *Integrated Development Environment* for the statistical software language `R`.

To get both `R` and `RStudio` requires two separate downloads and files, which can be found here:

- `R`: https://cran.r-project.org/mirrors.html (you need to select a location to download from; choose any one that is geographically close to you.)
- `RStudio`: https://www.rstudio.com/products/rstudio/download/.

2.1.1 Why do we have two programs?

Think of `R` as your basic program - this is the engine that does the computation. `RStudio` is a programe where you can see everything you are working on in one place. Figure 2.1 shows an example of an RStudio pane that I have:

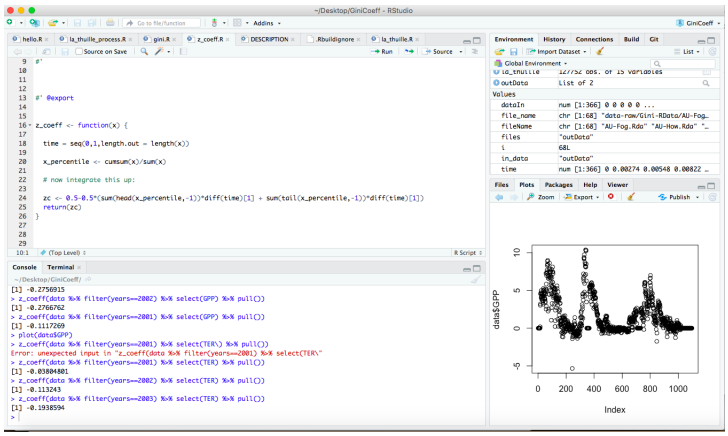


Figure 2.1: A sample RStudio pane from one of my projects.

There are 4 key panels that I work with, clockwise from the top:

- The **source** window is in the upper left - notice how those have different tabs associated with them. You can have multiple source files that you can toggle between. For the moment think of these as commands that you will want to send to R.

- The **environment** and **history** pane - these tables allow you to see what variables are stored locally in your environment, or the history of commands.
- The **files** and **plots** command (a simple plot I was working on is shown currently), but you can toggle between the tabs. The files tab shows the files in the current `Rstudio` project directory.
- Finally, the **console** pane is the place where `R` works and runs commands. You can type in there directly, otherwise we will also just "send' ' commands from the source down to the console.

Now we are ready to work with `R` and `RStudio`!

2.2 First steps: getting acquainted with R

Let's get started! Open up RStudio. Task one will be to create a project. A project is a central place to organize your materials for this course. You may do this already, but R can be picky about its working directory - and navigating to it. I found creating a project file is an easy way to avoid some of that fussiness. Let me describe steps in how to do this.

1. In `RStudio` select "File" then "New Project"
2. Next select the first option "New Directory" in the window - this will create a new folder on your computer.
3. At the next window choose New Directory or Existing Directory - it depends on where you want to place this project.
4. Name the project as you like.
5. Click the "Create Project" button.

2.2.1 "Working" with R

Our next step: where do we get R to do something? For example if we wanted to compute of `4+9` (yeah, it is 13, but this is an illustrative example), we could type this command in the `R` console (lower left) window. Let's try this now.

1. In the console type `4+9`
2. Then hit enter (or return)
3. Is the result 13?

Success! Now let me show you another way that works well if you have multiple lines of code to evaluate or save. Working with a script (`.R` file) is better. This will utilize the upper left hand corner of your `RStudio` window. (You may not have anything there when you start working on a project, so let's create one.)

1. In `RStudio` select "File" then "New File"
2. Next select the first option "New Script"
3. A new window called "UntitledX" should appear, where X is a number. You are set to go!

I like to use this window as a file to type stuff in and then evaluate it, which we will do next.

Pro tip: There are shortcuts to creating a new file: Ctrl+Shift+N (Windows and Linux) or Command+Shift+N (Mac)

2.2.2 Sending commands down to console.

Now we want to type and evaluate a command for R to do something. Click anywhere in the source file that you created and type the following:

```
4 + 9
```

```
## [1] 13
```

You have several options:

1. Copying and pasting the command to the window. Shortcuts are Ctrl+C / Command+C for copying and Ctrl+V / Command+C for Windows / Mac.
2. Run the line. This means that your cursor is at the line in your source file, then clicking the 'Run' button in the upper right hand side of the source window. Shortcuts are Ctrl+Enter / Command+Enter.

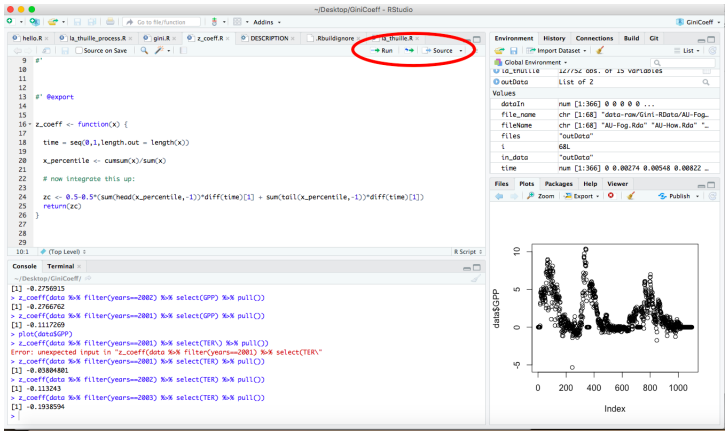


Figure 2.2: Sending a command to the console.

3. You can also source the whole file, which means runs all the lines from top to bottom. You do this by clicking the source button, or with shortcuts Ctrl+Shift+Enter / Cmd+Shift+Enter (Windows / Mac). You can imagine this makes things easier when you have SEVERAL lines of commands to evaluate.

Why do I like working with a script file? Well if I run some commands that have an error then it is much easier to just fix a quick mistake and then re-run the code.

It is also helpful to annotate your code with comments (`#`), which appear as green text in `RStudio`.

2.2.3 Saving your work

The neat part about a source file is that it allows you to save the file (Ctrl+S / Cmd+S). The first time you do this you may need to give this a name. The location where this file will be saved is in the same directory as your `.Rproj` project file. Now you have a file that you can come back to! In general I try to use descriptive names of files to I can refer back to them later.

2.3 Increasing functionality with packages

One awesome versatility with R is the ability to add packages - these packages extend the functionality of R with contributed, specialized code. These are similar to apps on your phone. You can get packages from a few different places:

- CRAN, which stands for **C**omprehensive **R** **A**rchive **N**etwork. This is the clearing house for many contributed packages - and allows for easy cross platform functonality.

One key package is `tidyverse`, which is actually a collection of packages. If you take an introductory data science course you will most likely be learning more about this package, but to install this at the command line you type the following:

```
install.packages("tidyverse")
```

Typing this line will connect to the CRAN download mirrors and install this set of packages locally to your computer. It make take some time, but be patient.

Another package you should install is `devtools`:

```
install.packages("devtools")
```

Sometimes when you are installing packages you may be prompted to install additional packages. In this case just say yes.

- Github. This is another place where people can share code and packages (including myself!). The code here has not been vetted through CRAN for compatibility, but if you trust the person sharing the code, it should work.

For this textbook I have written a collection of functions and data that we will use. This package name is called `demodelr` (**D**ifferential **E**quations and **Model**s in **R**). To install this package you will run the following line.

```
devtools::install_github("jmzobitz/demodelr", build = TRUE, build_opts = c("--no-resave-data", "--no-manual"),
```

The start of this command `devtools::` calls the function `install_github` from the `devtools` library. This is super handy when you just want to call one function from a library. What this command will do is pull in the package structure from my github page and install it locally.

Here is the good news: *you only need to install a package once before using it!* To load the package up into your workspace you use the command `library`:

```
library(tidyverse)
library(demodelr)
```

You need to load up your these libraries *each time you restart your R session.* This is part of the benefit of a script file - at the start I always declare the libraries that I will need at the start of the script file, as shown in the following figure:

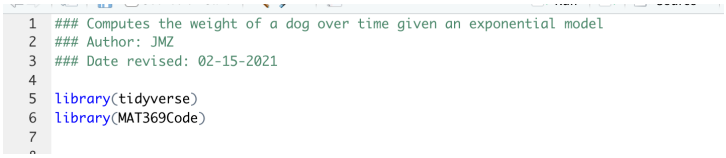


Figure 2.3: A sample R script.

Also notice here that the first few lines of the script file I used comments (prefaced with `#`) to denote the basic purpose of the file, who wrote it, and the date it was last revised. This type of information is good programming practice at the start.

2.4 Working with R: variables, data frames, and datasets

2.4.1 Creating variables

The next thing we will want to do is to define variables that are stored locally. This is pretty easy to do:

```
my_result <- 4 + 9
```

The symbol `<-` is assignment (you can use equals (=), but it is good coding practice to use the arrow for assignment). Notice how I named the variable called `my_result`. Generally I prefer using *descriptive* names for variables for the context at hand (In other words, `x` would be an odd choice - too ambiguous.) I also used snake case to string together multiple words. In practice you can use snake case, or alphabetic cases (`myResult`) or even `my.result` (although that may not be preferred practice in the long run). However, you can't use `my-result` because it looks like subtraction between variables `my` and `result`.

Once we have defined a variable, we can compute with it. For example `10*my_result` should yield 130. Cool, no?

As an example, let's define a sequence, spaced from 0 to 5 with spacing of 0.05. Store this in a variable called `my_sequence`. To do this we use the `seq` command and requires the starting value, ending value, and step size:

```
my_sequence <- seq(from = 0, to = 5, by = 0.05)
```

The format for the function `seq` is `seq(from=start,to=end,by=step_size)`. The `seq` command is a pretty flexible - there are alternative ways you can generate a sequence by specifying the starting and the end values along with the number of points. If you want to know more about `seq` you can always use `?` followed by the command - that will bring up the help values:

```
?seq
```

Once you get more comfortable with syntax in `R`, you will see that `seq(0,5,0.5)` gives the same result as `seq(from=0,to=5,by=0.05)`, but it is helpful to write your code *so that you can understand what it does.*

2.4.2 Data frames

A key structure in `R` is that of a data frame, which allows different types of data to be collected together. A data frame is like a spreadsheet where each column is a value and each row a value (much like you would find in a spreadsheet), as given in Table 2.1.

Table 2.1: A data frame

	mpg	disp
Mazda RX4	21.0	160
Mazda RX4 Wag	21.0	160
Datsun 710	22.8	108
Hornet 4 Drive	21.4	258
Hornet Sportabout	18.7	360

Table 2.2: Model solutions

time	model_1	model_2	model_3
0.000000	5.000000	5.0000	5.000000
6.060606	5.996981	669.1571	5.995486
12.121212	7.192755	1222.9000	7.188814
18.181818	8.626962	1684.5848	8.619147
24.242424	10.347145	2069.5158	10.333332

Table 2.1 shows the miles per gallon in one column (the variable `mpg` and the engine size (the variable `disp`) for different types of cars. The row names (`Mazda RX4`) just tell you the type of the car. Sometimes row names are not shown.

Another data frame may list solutions to a differential equation, like we did with our three infection models in Section 1 (Table 2.2).

Data frames are an example of *tidy* data, where each row is an observation, each column a variable (which can be quantitative or categorical). There are several different ways to define a data frame in `R`. I am going to rely on the approach utilized by the `tidyverse`, which calls data frames `tibbles`. So for example, here is I am going to define a data frame that computes the quadratic function $y = 3x^2 - 2x$ for $-5 \leq x \leq 2$.

```
x <- seq(from = -5, to = 2, by = 0.05)
y <- 3 * x^2 - 2 * x

my_data <- tibble(
  x = x,
  y = y
) # Notice I sam specifically defining x and y
```

Notice that the data frame `my_data` uses the column (variable) names of `x` and `y`. You could have also used `tibble(x,y)`, but it is helpful to name the columns in the way that you would like them to be named.

2.4.3 Reading in datasets

`R` has a lot of built in datasets! In fact to see all the datasets, type `data()` at the console. This will popup a new window in `RStudio` with the names. Take some time exploring them. So cool!

If you want to see the datasets for a specific package (such as `demodelr`) you type `data(package = "demodelr")` at the console.

Perhaps what is most important is being able to read in datasets provided to you. Data come in several different types of formats, but one of the more versatile ones are csv (**c**omma **s**eparated **v**alues). What you need to do is the following:

- Where you have your .Rproj file located, create a folder called `data` or `datasets`
- Save the file locally on your computer. Take note where you have it saved on your computer, and drag the file to your `data` folder.
- To read in the file you will use the command `read_csv`, which has the following structure:

```
in_data <- read_csv(FILENAME)
```

The data gets assigned to the variable `in_data` (You can call this variable what you want.) For example I have the following csv file of ebola data, which I read in via the following:

Table 2.3: Weight of a dog over time

days	mass
31	6.25
62	10.00
93	20.00
99	23.00
107	26.00
113	27.60
121	29.80
127	31.60
148	37.20
161	41.20
180	48.70
214	54.00
221	54.00
307	63.00
452	66.00
482	72.00
923	72.20
955	76.00
1308	75.00

```
ebola <- read_csv("data/ebola.csv")
```

Notice the quotes around the FILENAME. **Pro tip:** If you have the data files in the data folder, in RStudio you can type "data" and it may start to autocomplete - this is hand (you can also use tab.)

2.5 Visualization with R

Now we are ready to begin visualizing data frames. Two types of plots that we will need to make will be a scatter plot and a line plot. We are going to consider both of these separately, with examples that you should be able to customize.

2.5.1 Making a scatterplot

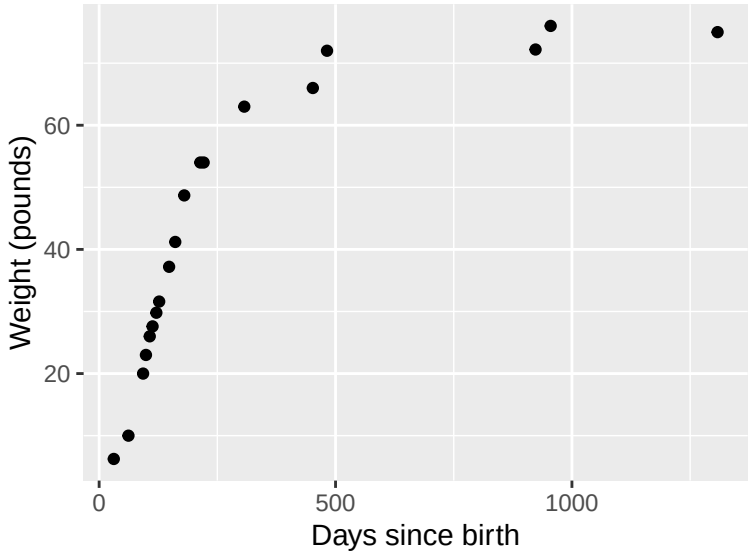
One dataset we have is the mass of a dog over time, adapted from here. We have two variables here: D = the age of the dog in days and W = the weight of the dog in pounds. I have the data loaded into the `demodelr` package, which you can investigate by typing the following at the command line (I display it below as well in Table 2.3).

```
glimpse(wilson)
```

(Notice that I have assumed you have the `demodelr` library loaded.) You can also explore the documentation for this dataset by typing `?wilson` at the console.

Notice that this data frame has two variables: `days` and `mass` To make a scatter plot of these data we are going to use the command `ggplot`:

```
ggplot(data = wilson) +
  geom_point(aes(x = days, y = mass)) +
  labs(
    x = "Days since birth",
    y = "Weight (pounds)"
  )
```



Wow! This looks complicated. Let's break this down step by step:

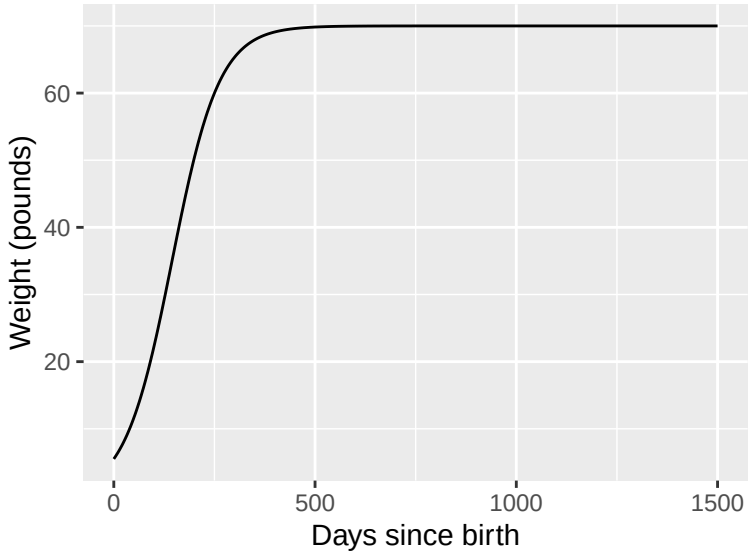
- `ggplot(data = wilson)` + sets up the graphics structure and identifies the name of the data frame we are including.
- `geom_point(aes(x = days, y = mass))` defines the type of plot we are going to be making.
- `geom_point()` defines the type of plot geometry (or *geom*) we are using here - in this case, a point plot.
- `aes(x = days, y = mass)` determines the *aesthetics* of the plot. On the x axis is the days variable, on the y axis is the mass variable.
- The statement beginning with `labs(x=...)` defines the labels on the x and y axes.

I know this seems like a lot to write for a plot, but this structure is actually used for some more advanced data visualization. Trust me - learning how to make informative plots can be a useful skill!

2.5.2 Making a line plot

Using the same `wilson` data, later on we will discover that the function $W = f(D) = \frac{70}{1 + e^{2.46 - 0.017D}}$. represents these data. In order to make a plot of this function we can use need to first build a data frame:

```
days <- seq(from = 0, to = 1500, by = 1) # Choose spacing that is "smooth enough"
mass <- 70 / (1 + exp(2.46 - 0.017 * days))

wilson_model <- tibble(
  days = days,
  mass = mass
)

ggplot(data = wilson_model) +
  geom_line(aes(x = days, y = mass)) +
  labs(
    x = "Days since birth",
    y = "Weight (pounds)"
  )
```



Notice that once we have the data frame set up, the structure is very similar to the scatter plot - but this time we are calling using `geom_line()` than `geom_point`.

2.5.3 Changing options

Want a different color? Thicker line? That is fairly easy to do. For example if we wanted to make either our points or line a different color, we can just choose the following:

```
ggplot(data = wilson) +
  geom_point(aes(x = days, y = mass), color = "red", size = 2)
labs(
  x = "Days since birth",
  y = "Weight (pounds)"
)
```

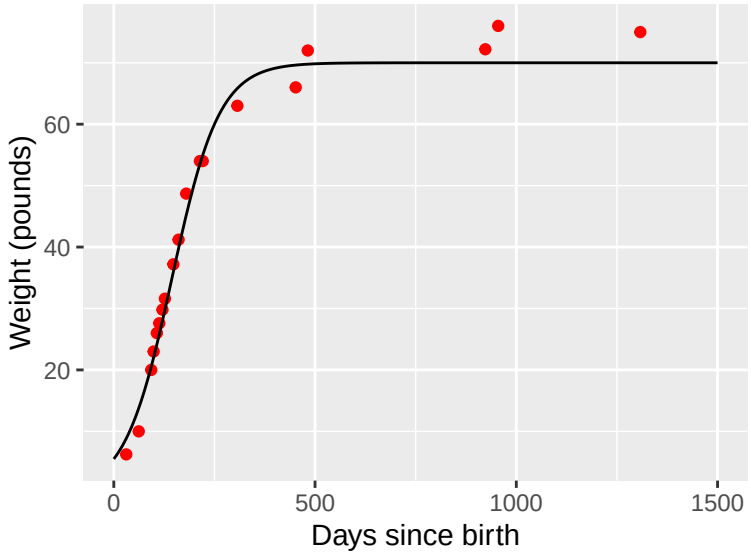
Notice how the command `color='red'` was applied *outside* of the aes - which means it gets mapped to each of the points in the data frame. `size=2` refers to the size (in millimeters) of the points. I've linked more options about the colors and sizes you can use here:

- **Named colors in R:** LINK Scroll down to "Picking one color in R" - you can see the list of options!
- **More colors:** LINK. More information about working with colors.
- **Using hexadecimal colors:** LINK (You specify these by the code so `"#FF3300"` is a red color.)
- **Changing sizes of lines and points:** LINK

2.5.4 Combining scatter and line plots.

This is actually easy to do, especially since we are combining both the plot geoms together. Try running the following code (I am still using the data frame `wilson_model` as defined above:

```
ggplot(data = wilson) +
  geom_point(aes(x = days, y = mass), color = "red") +
  geom_line(data = wilson_model, aes(x = days, y = mass)) +
  labs(
    x = "Days since birth",
    y = "Weight (pounds)"
  )
```



Notice in the above code a subtle difference when I added in the dataset `wilson_model` with `geom_line`: you need to name the `data` bringing in a new data frame to a plot geom.

While it may be useful to have a legend to the plot, for this course we will make plots where this the context will be more apparent. Additional reading on legends can be found here.

2.6 Defining functions

We will study lots of other built-in functions for this course, but you may also be wondering how you define your own function (let's say $y = x^3$). We need the following construct:

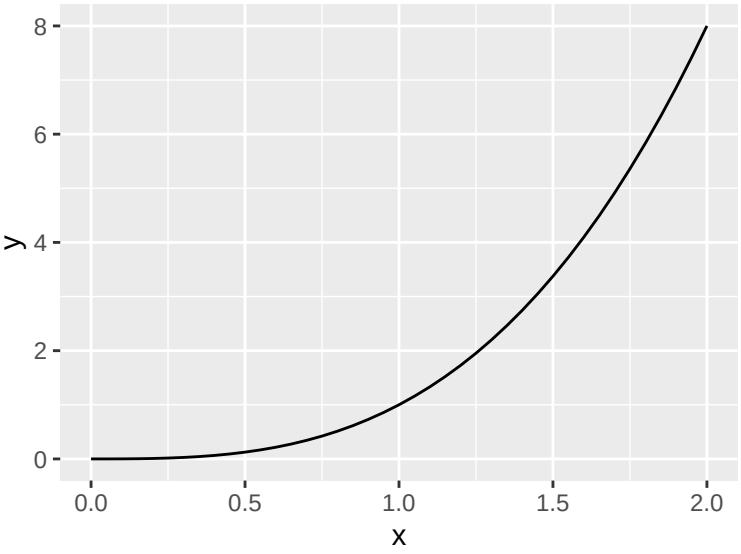
```
function_name <- function(inputs) {

  # Code

  return(outputs)
}
```

Here `function_name` serves as what you call the function, inputs are what you need in order to run the function, and outputs are what gets returned. So if we are doing $y = x^3$ then we will call that function `cubic`:

```
cubic <- function(x) {
  y <- x^3
  return(y)
}
```

So now if we want to evaluate $y(2) = 2^3$ we type `cubic(2)`. Neat! Now let's make a plot of the graph $y = x^3$ using the function defined as `cubic`. Here is the `R` code that will accomplish this:

```
x <- seq(from = 0, to = 2, by = 0.05)
y <- cubic(x)

my_data <- tibble(x = x, y = y)

ggplot(data = my_data) +
  geom_line(aes(x = x, y = y)) +
  labs(
    x = "x",
    y = "y"
  )
```



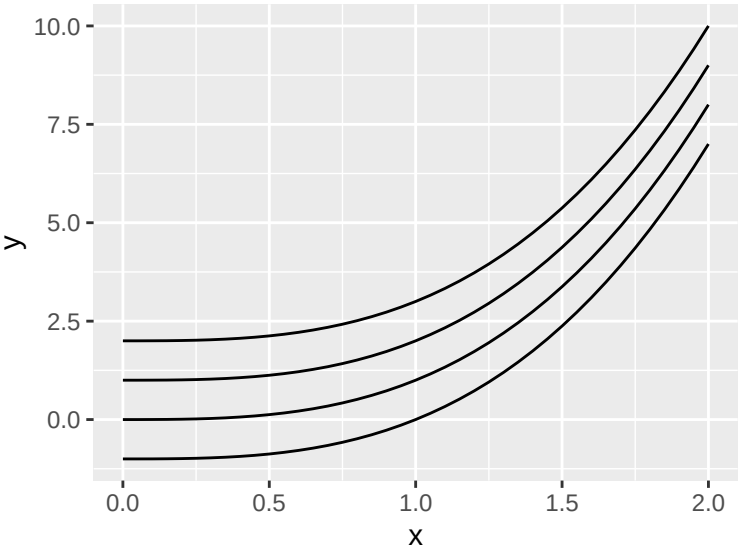
2.6.1 Functions with inputs

Sometimes you may want to define a function with different input parameters, so for example the function $y = x^3 + c$. To define that, we can modify the function to have input variables:

```
cubic_revised <- function(x, c) {
  y <- x^3 + c
  return(y)
}
```

So if we want to plot what happens for different values of `c` we have the following:

```
x <- seq(from = 0, to = 2, by = 0.05)


my_data_revised <- tibble(
  x = x,
  c_zero = cubic_revised(x, 0),
  c_pos1 = cubic_revised(x, 1),
  c_pos2 = cubic_revised(x, 2),
  c_neg1 = cubic_revised(x, -1)
)

ggplot(data = my_data_revised) +
  geom_line(aes(x = x, y = c_zero)) +
  geom_line(aes(x = x, y = c_pos1)) +
  geom_line(aes(x = x, y = c_pos2)) +
  geom_line(aes(x = x, y = c_neg1)) +
  labs(
    x = "x",
    y = "y"
  )
```



Notice how I defined multiple columns of the data frame `my_data_revised` in the `tibble` command, and then used mutiple `geom_line` commands to plot the data. Since we had combined the different values of `c` in a single data frame we didn't need to define the `data` with each instance of `geom_line`.

2.7 Concluding thoughts

This is not meant to be a self-contained section in `R` but rather one to get you - those miles have been trod by others, and here are few of my favorites that I turn to:

- **R Graphics**. This is a go to resource for making graphics. (I also use google a lot too.)
- **The Pirates Guide to R**. This book promises to build your R knowledge from the ground up.
- **R for Reproducible Scientific Analysis**. This set of guided tutorials can help you build your programming skills in R.
- **R for Data Science** this is a useful book to take your R knowledge to the next level.

The best piece of advice: DON'T PANIC! Patience and persistence are your friend. Reach out for help, and recognize that like with any new endeavor, practice makes progress.

2.8 Exercises

Exercise 2.1. Create a folder on your computer and a project file where you will store all your R work for this textbook.

Exercise 2.2. Install the packages `devtools`, `tidyverse` to your R installation. Once that is done, then install the package `demodelr` from my github page.

Exercise 2.3. What are the variables listed in the dataset `phosphorous` in the `demodelr` library? (Hint: try the command `?phosphorous`.)

Exercise 2.4. Make a scatterplot (`geom_point()`) of the dataset `phosphorous` in the `demodelr` library. Be sure to label the axes.

Exercise 2.5. Change the line plot of Wilson's weight over time so the line is blue and the size is 4.

Exercise 2.6. Change the color of the scatterplot of Wilson's weight over time to a either a hexadecimal color or a named color of your choice.

Exercise 2.7. For this exercise you will do some plotting:

a. Define a sequence (call this sequence x) that ranges between -12 to 12 with spacing of .05.
b. Also define the variable y such that $y = \sin(x)$.
c. Make a scatter plot to graph $y = \sin(x)$. Set the points to be red.
d. Make a line plot to graph $y = \sin(x)$. Label the x-axis with your favorite book title. Label the y-axis with your favorite food to eat.

Exercise 2.8. An equation that relates a consumer's nutrient content (denoted as y) to the nutrient content of food (denoted as x) is given by: $y = cx^{1/\theta}$, where $\theta \geq 1$ and c are both constants is a constant. Let's just assume that $c = 1$ and the $0 \leq x \leq 1$.

Write a function called `nutrient` that will make a sequence of `y` values for an input `x` and `theta` ($theta$). Then use that code to make a make a line plot (`geom_line()`) for five different values of $\theta > 1$, appropriately labeling all axes.

Exercise 2.9. Researchers measured the phosphorous content of *Daphnia* and its primary food source algae. This is the dataset `phosphorous` in the `demodelr` library.

Researchers believe that *Daphnia* has strict homeostatic regulation of the phosphorous in algae, and as such want to determine the value of θ in the equation $y = y = cx^{1/\theta}$. They have already determined that the value of $c = 1.737$.

a. If you haven't already, make a scatterplot (`geom_point()`) of the dataset `phosphorous` in the package library. Be sure to label the axes correctly.
b. Use your function `nutrient` from the previous exercise to make an initial guess for `theta` (θ) that would be consistent with the data. You can evaluate your guess by plotting (with `geom_line()`) against the data.
c. Use guess and check to refine the value of θ that seems to work best.
d. Report your value of θ.

Exercise 2.10. For this exercise you will investigate some built-in functions. Remember you can learn more about a function by typing `?FUNCTION`, where `FUNCTION` is the name.

a. Explain (using your own words) what the function `runif(1,100,1000)` does.
b. Explain (using your own words) what the function `ceiling()` does, showing an example of its use.

Exercise 2.11. For this exercise you write a sample function file.

a. Create a new source file and save it as `myFunction.R`.
b. Type this code the file you created: `myInteger <- ceiling(runif(1, 100, 1000))` (This will declare a variable `myInteger` that you will work with in the following steps.)
c. Determine a function in R that will **compute a cumulative sum** from 1 to the value of `myInteger`. Modify your file so that it also computes the cumulative sum and then source your file.
d. Copy and paste your function into your homework document for evaluation.

Exercise 2.12. The Ebola outbreak in Africa in 2014 severely affected the country of Sierra Leone. A model for the number of deaths D due to ebola is given by the following equation:

$$D(t) = \frac{K \cdot N_0}{N_0 + (K - N_0)\exp(-rt)},$$

where $K = 3980$, $N_0 = 5$ and $r = 0.0234$. The variable t is in days. Use `geom_line()` to visualize this curve from $0 \leq t \leq 700$.

Exercise 2.13. Consider the following piecewise function:

$$y = \begin{cases} x^2 & \text{for } 0 \leq x < 1, \\ 2 - x & \text{for } 1 \leq x \leq 2 \end{cases} \tag{2.1}$$

a. Define a function in `R` that computes y for $0 \leq x \leq 2$.
b. Use `geom_line()` to generate a graph of $y(x)$ over the interval $0 \leq x \leq 2$.

Exercise 2.14. An insect's development rate r depends on temperature T (degrees Celsius) according to the following equation:

$$r = \begin{cases} 0.1 & \text{for } 17 \leq T < 27, \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

a. Define a function in `R` that computes r for $0 \leq T \leq 30$.
b. Use `geom_line()` to generate a graph of $r(T)$ over the interval $0 \leq T \leq 30$.

Chapter 3

Modeling With Rates of Change

So far we have looked at some examples for how we can apply rates of change to develop a mathematical model, and also learned a little bit about the ways we can apply computational software such as `R`. In this section we are going to look some additional examples of how we can translate equations with rates of change to understand phenomena. The focus here will be on writing differential equations from a contextual description.

Oftentimes when we construct differential equations from a contextual description we bring our own understanding and knowledge to this situation. How *you* may write down the differential equation may be different from someone else - *do not worry!* This is the fun part of modeling: models can be considered testable hypotheses that can be refined when confronted with data. In this section I work through a few well-known examples from mathematical biology and you will apply that knowledge to develop models from context.

3.1 Lynx and Hares

Our first example is a *system of differential equations*. The context is between the snowshoe hare and the Canadian lynx. Figure 3.1 shows a picture of them below from link

Figure 3.1: The lynx and hare - aren't they beautiful?

Figure 3.2 timeseries of their population is shown with this figure from Stenseth et al. (1997).

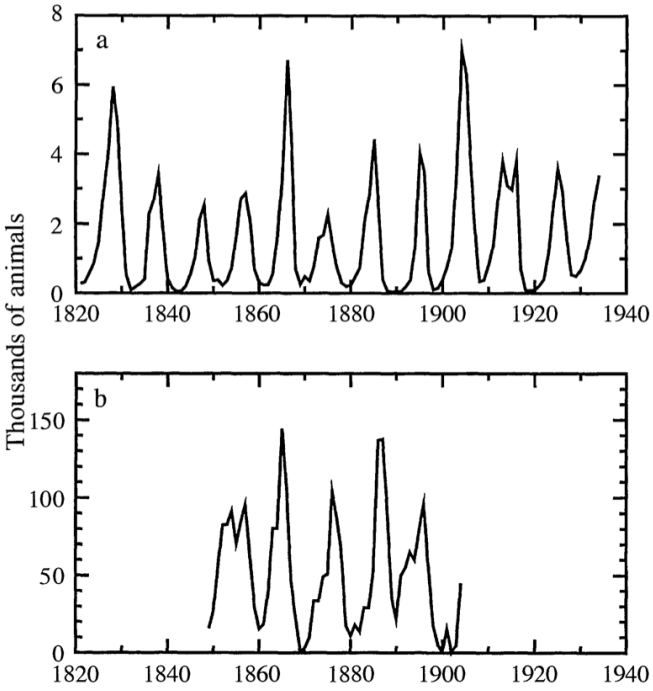


FIG. 1. Cycles in abundance of Canadian lynx and snowshoe hare as revealed in the fur harvest records of the Hudson's Bay Company. (a) Lynx fur harvest from the MacKenzie River District (after Elton and Nicholson 1942*b*). (b) Snowshoe hare harvest from the regions near Hudson's Bay (after MacLulich 1957).

Figure 3.2: A timeseries of the combined lynx and hare system. Notice how the populations are coupled with each other.

Notice how in Figure 3.2 both populations seem to fluctuate periodically. One plausible reason is that the lynx prey on the snowshoe hares, which causes the population to initially decline. Once the snowshoe hare population declines, then there is less food for the lynx to survive, so their population declines. The decline in the lynx population causes the hare population to increase, and so on it goes ...

In summary it is safe to say that the two populations are *coupled* to one another. But in order to understand how they are coupled together, first let's consider the two populations *separately*.

The hares grow much more quickly than then lynx - in fact some hares have been known to reproduce several times a year. A reasonable assumption for large hare populations is that rate of change of the hares is proportional to the hare population. Based on this assumption Equation (3.1) describes the rate of change of the hare population, with H is the population of the hares:

$$\frac{dH}{dt} = rH \tag{3.1}$$

In this case we know that the growth rate r is positive, so then the rate of change (H') will be positive as well, and H will be increasing. Typical values given for r in Stenseth et al. (1997) are between 1.8 - 2.0 year^{-1}. You may be thinking that the units on r seem odd - (year^{-1}). Another way to think about r is to take its inverse: $r^{-1} \approx 0.5$ - 0.55 years. Then r^{-1} represents the amount of time that passes before the hare population increases (pretty short!)

Let's consider the lynx now. A approach is to assume their population declines exponentially, or changes at the rate proportional to the current population. Let's consider L to be the lynx population, with the following differential equation (Equation (3.2)):

$$\frac{dL}{dt} = -dL \tag{3.2}$$

We assume the death rate d in Equation (3.2) is positive, leading to a negative rate of change for the Lynx population (and a decreasing value for L). Typical values of d are 0.9 - 2.4 year^{-1}. Similar to r, another way to understand d is to take its inverse (about 0.4 - 1.1 years), which represents the amount of time that passes before the lynx population decreases by one.

The next part to consider is how the lynx and hare interact. Since the hares are prey for the lynx, when the lynx hunt, the hare population decreases. We can represent the process of hunting with the following adjustment to our hare equation:

$$\frac{dH}{dt} = rH - bHL \tag{3.3}$$

So the parameter b represents the hunting rate. Notice how we have the term HL for this interaction. This term injects a sense of realism: if the lynx are not present ($L = 0$), then the hare population can't decrease due to hunting. We say that the *interaction* between the hares and the lynx with multiplication. Typical values for b are 480 - 870 hares $\cdot$ lynx^{-1} year^{-1}. It is okay if that unit seems a little odd to you - it should be! Here is one way to think about it. The quantity $\frac{dH}{dt}$ represents the *rate of change* of the hares, so it should have units of hares per year. Since the term bHL has both lynx and hare, the units for b need to account for this.

How does hunting affect the lynx population? One possibility is that it increases the lynx population:

$$\frac{dL}{dt} = bHL - dL \tag{3.4}$$

Notice the symmetry between the rate of change for the hares and the lynx equations. In many cases this makes sense - if you subtract a rate from one population, then that rate should be added to the receiving population. You could also argue that there is some efficiency loss in converting the hares to lynx - not all of the hare is converted into lynx biomass. In this situation we sometimes like to adjust the hunting term for the lynx equation with another parameter e, representing the efficiency that hares are converted into lynx:

$$\frac{dL}{dt} = e \cdot bHL - dL \tag{3.5}$$

(sometimes people just make a new parameter $c = e \cdot b$, but for now we will just leave it as is). Equation (3.6) shows the coupled system of differential equations:

$$\begin{aligned} \frac{dH}{dt} &= rH - bHL \\ \frac{dL}{dt} &= ebHL - dL \end{aligned} \tag{3.6}$$

The schematic diagram representing these interactions is the following is shown in Figure 3.3:

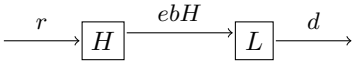


Figure 3.3: Schematic diagram Lynx-Hare system.

Equation (3.6) is a classical model in mathematical biology and differential equations - it is called the *predator prey* model, also known as the Lotka-Volterra equations. There is a lot of interesting mathematics from this system of equations that we will study later in this textbook. In later sections we will graphically and numerically analyze these equations and their solutions.

3.2 The Law of Mass Action

Notice in the previous section that the interaction between the lynx and the hare was of the form bHL - meaning you needed both positive values of H and L for the interaction to continue. This law states that the rate of a change is directly proportional to the *product* of the populations.

This assumption of the law of mass action is also commonly used in chemical reactions - especially in modeling enzyme dynamics. For example let's say you have a substrate A that reactions with enzyme B to form a product S. Perhaps you might have seen this represented as a reaction equation:

$$A + B \to S \tag{3.7}$$

How we would write the product of formation, or $\frac{dS}{dt}$ is the following:

$$\frac{dS}{dt} = kAB, \tag{3.8}$$

where k is the proportionality constant or the rate constant associated with the reaction. If we wanted to represent this as a schematic we would have the following diagram (Figure 3.4):

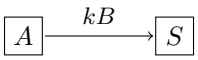


Figure 3.4: Schematic diagram of the law of mass action.

We could also consider if there was a constant decay of the substrate, which we might revise Figure 3.4 to Figure 3.5:

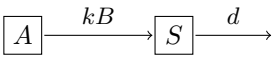


Figure 3.5: Revised schematic diagram of the law of mass action with decay.

For this case, the rate of change of S would then be:

$$\frac{dS}{dt} = kAB - dS, \tag{3.9}$$

You may be wondering about rates for A and B. When S is formed A and B are catalyzed, so the rate of formation for S (the positive term in Equation (3.9)) will be a loss for A and B (Equation (3.10)):

$$\begin{aligned} \frac{dA}{dt} &= -kAB \\ \frac{dB}{dt} &= -kAB \end{aligned} \tag{3.10}$$

Both equations are similar in this case, not pending any additional inputs or outputs.

3.3 Establishing species

Let's look at another example where from we will determine a differential equation model from a context:

> A newly introduced plant species is introduced to a region. It competes with another established species for nutrients (and is a better competitor). However, the growth rate of the new species is proportional to the difference between the current number of established species and the number of new species. You may assume that the number of established species is a constant E.

For this problem we will start by naming our variables. Let N represent number of new species and E the number of established species. We will break this down accordingly:

- *"the growth rate of the new species"* means $\frac{dN}{dt}$.
- *"is proportional to the difference between the current number of established species and the number of new species"* means $k \cdot (E - N)$, where k is the proportionality constant. Including this parameter helps to avoid assuming we have a 1:1 correspondence between the growth rate of the new species and the population difference.

- *"and is a better competitor"* helps to explain why the term is $k \cdot (E - N)$ insteady of $k \cdot (N - E)$. We know that the newly established species will start out in much smaller numbers than N. But since it is a better competitor, we would expect its rate to increase initially. So $\frac{dN}{dt}$ should be *positive* rather than negative. Assuming $N < E$, then $E - N > 0$, which guarantees that the new species will grow.

So this description could be modeled with Equation (3.11):

$$\frac{dN}{dt} = k(E - N) \tag{3.11}$$

Does Equation (3.11) seem familiar to you? It is similar to Equation (1.4) in Section 1 for the spread of Ebola! While this may seem surprising, it is often the case that similar equations appear in different contexts. So do you have to memorize every possible model for every possible context? The answer is emphatically *NO*! Rather it is more advantageous to learn how to analyze models qualitatively rather than memorize several different types of models and not see the connections between them.

An interesting solution to a differential equation is the *steady state* or *equilibrium* solution. We find this where the rate equals zero. Let's take a look how to do that for our establishing plant model

Example 3.1. What is the steady state value for the differential equation $\frac{dE}{dt} = k(E - N)$? (That is solve for E when $\frac{dE}{dt} = 0$.)

Solution. Let's solve $\frac{dE}{dt} = k(E - N) = 0$. For this equation we want to express the right hand side in terms of E. The parameter k is a constant $k > 0$, so really the steady state occurs when $E - N = 0$, or when $N = E$.

What this model tells us that eventually the new species will overtake the established species E.

3.4 Other types of functional responses

In several examples we have seen a rate of change proportional to the current population, as in the rate of growth of the hare population is rH. This is one example of what we would call a functional response. Another type of functional response assumes that the rate reaches a limiting value proportional to the population size, so $\frac{dH}{dt} = \frac{rH}{1 + arH}$. This is an example of a **type II functional response**. Finally, the type II response has also been generalized (a **type III functional response**) $\frac{dH}{dt} = \frac{rH^2}{1 + arH^2}$. Figure 3.6 shows all three functional responses together:

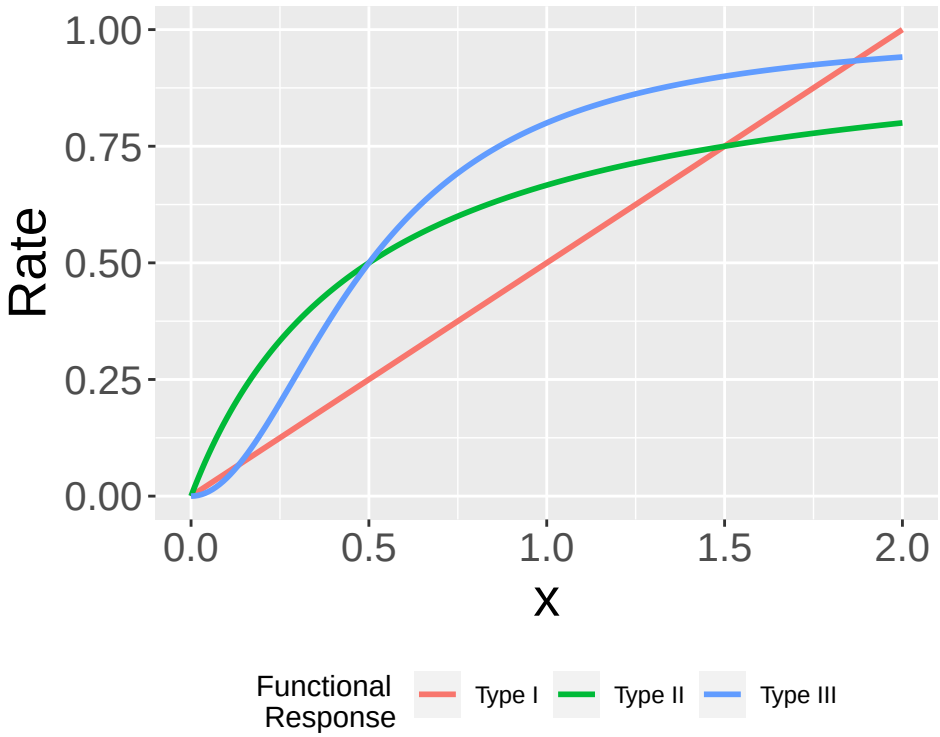


Figure 3.6: Comparison between Type I - Type III functional responses.

Notice the limiting behavior in the Type II and Type III functional responses. These responses are commonly used in ecology and predator-prey dynamics and in problems of how animals search for food.

3.5 Exercises

Exercise 3.1. Consider the following type of functional responses:

$$\text{Type I: } \frac{dP}{dt} = 0.1P \tag{3.12}$$

$$\text{Type II: } \frac{dP}{dt} = \frac{0.1P}{1 + .03P} \tag{3.13}$$

$$\text{Type III: } \frac{dP}{dt} = \frac{0.1P^2}{1 + .05P^2} \tag{3.14}$$

For each of the functional responses evaluate $\lim_{P\to\infty} \frac{dP}{dt}$. Since these functional responses represent a rate of change of a population, what are some examples (hypothetical or actual) would each of these responses be appropriate?

Exercise 3.2. A population grows according to the equation $\frac{dP}{dt} = \frac{0.1P}{1 + .05P} - P$.

a. On the same axis, plot the equations $f(P) = \frac{0.1P}{1 + .05P}$ and $g(P) = P$. What are the two positive values of P where $f(P)$ and $g(P)$ intersect?
b. Next algebraically determine the two steady state values of P, that is solve $\frac{dP}{dt} = 0$ for P. (*Hint:* factor a P out of the expression $\frac{0.1P}{1 + 5P} - P$.)
c. Does your algebraic solution match your graphical solutions?

Exercise 3.3. A population grows according to the equation $\frac{dP}{dt} = 2P - \frac{4P^2}{1 + P^2}$.

a. On the same axis, plot the equations $f(P) = 2P$ and $g(P) = \frac{4P^2}{1 + P^2}$. What are the two positive values of P where $f(P)$ and $g(P)$ intersect?
b. Next algebraically determine the two steady state values of P, that is solve $\frac{dP}{dt} = 0$ for P. (*Hint:* factor a P out of the expression $2P - \frac{4P^2}{1 + P^2}$.)
c. Does your algebraic solution match your graphical solutions?

Exercise 3.4. A population grows according to the equation $\frac{dP}{dt} = \frac{aP}{1 + abP} - dP$, where a, b and d are parameters. Determine the two steady state values of P, that is solve $\frac{dP}{dt} = 0$ for P.

Exercise 3.5. A chemical reaction takes two chemicals X and Y to form a substrate Z through the law of mass action. However the substrate can also disassociate. The reaction schematic is the following:

$$X + Y \rightleftharpoons Z, \tag{3.15}$$

where the proportionality constant k_+ is associated with the formation of the substrate Z and k_- the disassociation (Z decays back to X and Y).

Write down a differential equation that represents the rate of reaction $\frac{dZ}{dt}$.

Exercise 3.6. For each of the following exercises consider the following contextual situations modeling rates of change. Name variables for each situation and write down a differential equation describing the context. Be sure to identify and briefly describe any parameters you need for your model. For each problem you will need to:

- Name and describe all variables.
- Write down a differential equation.
- Identify and describe any parameters needed.
- Write a brief one-two sentence explanation of why your differential equation models the situation at hand.
- Hand sketch a rough graph of what you think the solution as a function of time - *note:* your solution needs to be consistent with your explanation and vice versa.

a. The rate of change of an animal's body temperature is proportional to the difference in temperature between the environment.
b. A plant grows propritional to its current length L. Assume this proportionality constant is μ, whose rate also decreases proportional to its current value. You will need to write down a system of two equation with variables L and μ.
c. A patient undergoing chemotherapy receives an injection at rate I. This injection decreases the rate that a tumor accumulates mass. Independent of the injection, the tumor accumulates mass at a rate proportional to the mass of the tumor.
d. A cell with radius r assimilates nutrients at a rate proportional to its surface area, but uses nutrients proportional to its volume. Determine an equation that represents the rate of change of the radius.
e. A patient undergoing chemotherapy receives an injection at rate I. This injection decreases the rate that a tumor accumulates mass. Independent of the injection, the tumor accumulates mass at a rate proportional to the mass of the tumor.
f. The rate that a cancer cell divides (increases in amount) is proportional to the amount of healthy cells in its surrounding environment. You may assume that a healthy cell has a mortality δ_H and a cancer cell has mortality δ_C. Be sure to write down a system of differential equations for the population of cancer cells C and healthy cells H.
g. The rate that a virus is spread to the population is proportional to the probability that a person is sick (out of N total sick and healthy individuals).

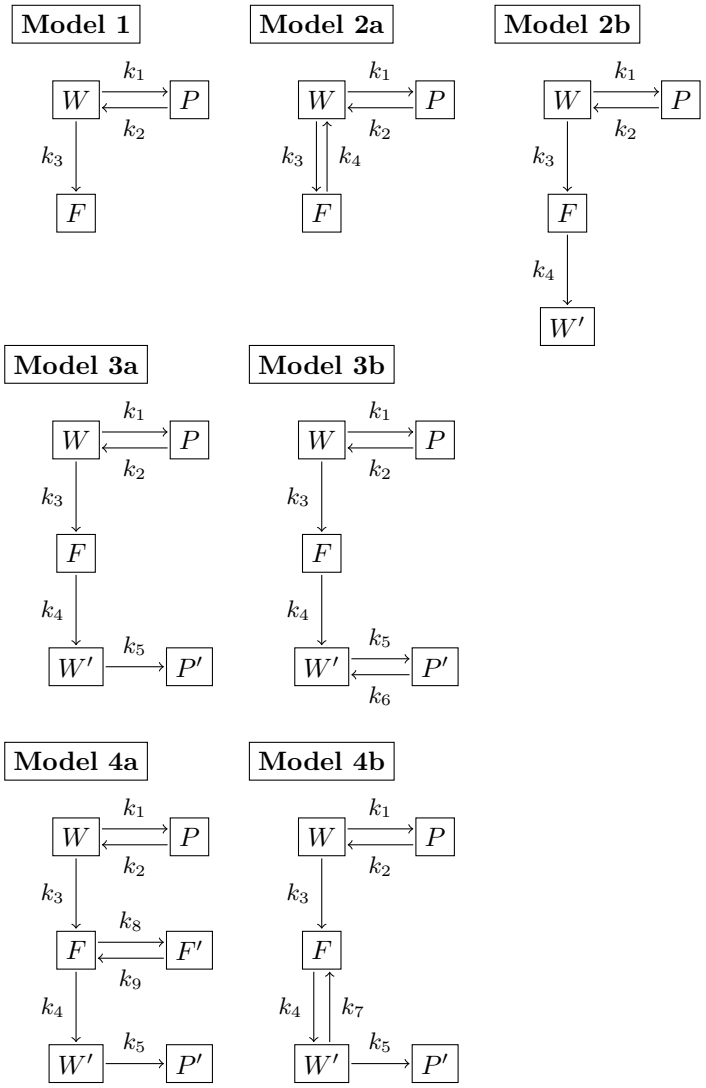


Figure 3.7: Reaction schemes.

Exercise 3.7. You are tasked with the job of investigating the effect of a pesticide on water quality, in terms of its effects on the health of the plants and fish in the ecosystem. Different models can be created that investigate the effect of the pesticide. Different types of reaction schemes for this system are shown in Figure 3.7, where F represents the amount of pesticide in the fish, W the amount of pesticide in the water, and S the amount of pesticide in the soil. The prime (e.g. F', W', and S' represent other bound forms of the respective state). In all seven different models can be derived. For each of the model schematics, apply the Law of Mass Action to write down a system of differential equations.

Chapter 4

Euler's Method

The focus of this section is on *approximation* of solutions to a differential equation via a numerical method. Typically a first numerical method one might learn to tackle this problem is *Euler's method*, which is so fundamental it was popularized in the movie Hidden Figures.

The way we are going to do this is through expansion of the idea of a *locally linear approximation* to the tangent line. Let's start with an example.

Example 4.1 (The flu bug)**.** The rate of change of the flu through a population is given by the number of people infected t days after infection is,

$$\frac{dI}{dt} = 3e^{-.025t}.$$

Assuming that $I(0) = 10$, what is a locally linear approximation to this infection? Second, using your linear approximation, what would you predict is the value after one day ($I(1)$)?

Solution. In order to solve this problem, we know that the locally linear approximation is to $I(t)$ at $t = 0$ is $L(t) = I(0) + I'(0) \cdot (t - 0)$. Here, $I(0) = 10$ and $I'(0) = 3$, so

$$L(t) = 10 + 3t$$

. Using $L(t) \approx I(t)$, we have $L(1) = 10 + 3 = 13$. So our model predicts there will be 13 people sick.

Notice in Example 4.1 we used two pieces of information: the (given) value of the function at $t = 0$ and the estimate of the derivative from the rate of change.

It may be helpful to compare our prediction from $L(1)$ to the actual value. The solution to the differential equation in Example 4.1 is $I(t) = 130 - 120e^{-.025t}$ (you should verify this is the case by differentation). Let's also our approximation to the actual solution in the following table:

t	Linear approximation	Actual Solution
0	10	10
1	13	12.96

Not too bad, huh? Our approximation at $L(1)$ is an *overestimate*, mainly because the actual solution is concave down, but it isn't that far off.

Let's build this solution out a little more by computing the rate of change at $t = 1$, assuming that thirteen people is a pretty close estimate for the number infected (I) at $t = 1$. What we could do is to build *another* linear approximation using the differential equation. Let's call this linear approximation $L_1(t)$ to distinguish it from $L(t)$. The formula for $L_1(t)$ (the locally linear approximation is to $I(t)$ at $t = 1$) is

$$L_1(t) = I(1) + I'(1) \cdot (t - 1)$$

.

Here, $I(1) = 13$ and $I'(1) = 2.92$, and $L(t) = 13 + 2.92(t - 1)$.

Assuming that $L_1(t) \approx I(t)$, we can evaluate $L_1(t)$ at $t = 2$ *as an approximation* for $I(2)$: have $L(2) = 13 + 2.92 = 15.92$. Comparing this to the actual solution at $t = 2$, we have $I(2) = 15.85$. Again, not too bad of a solution.

We can continue to build out the solution from there. Figure @ref(fig:eulers_ver1) shows what we would have for a solution if we continued to build out this approach:

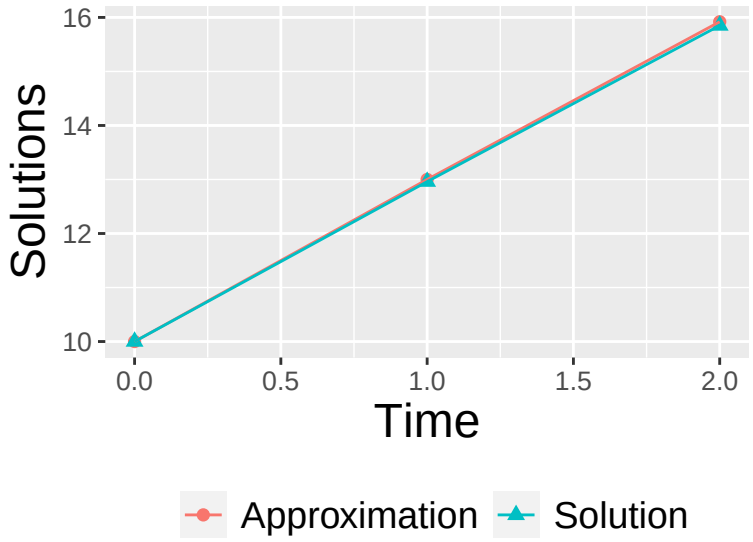


Figure 4.1: (#fig:eulers_ver1)Approximation of a solution using local linearity

When you plot them they do look indistinguishable from each other by eye. It looks like we are onto something here!

4.1 Defining an Algorithm

Here would be an algorithm that would describe our process to determine a solution to a differential equation:

- Determine the locally linear approximation at a given point.
- Forecast out to another time value.
- Repeat the locally linear approximation.

If we continue on in this way, let's take a look at how our approximation would do after several days:

t	Approximate Solution	Actual Solution
90	118.4	117
95	119.9	118.6

Now it seems that our approximation isn't so accurate as time goes on. What if we updated the infection rate every half day? I know this means that we would be doing additional work (more iterations), but taking smaller timesteps goes hand in hand with more accurate solutions. Let's start out smaller with the first few timesteps:

t	I	$\frac{dI}{dt}$	$\frac{dI}{dt} \cdot \Delta t$
0	10	3	1.5
0.5	$= 10 + 1.5 = 11.5$	2.96	1.48
1	$= 11.5 + 1.48 = 12.98$	2.92	1.46
1.5	$= 12.92 + 1.46 = 14.38$	2.88	1.44
2	$= 14.38 + 1.44 = 15.82$		

Notice how we have started to build up a way to organize how to compute the solution. Each column is a "step" of the method, computing the solution at a new timestep based on our step size Δt. The third column just computes the value of the derivative for a particular time and I, and then the fourth column is the *increment* size, or the amount we are forecasting the solution will grow by to the next timestep. (There are other ways to think about this, but if you have a *rate of change* multiplied by a time increment this will give you an approximation to the net change in a function.)

This idea of *approximate, forecast, repeat* is the heart of many numerical methods that approximate solutions to differential equations. The particular method that we have developed here is called *Euler's Method*. We display the results from additional steps in Figure 4.2.

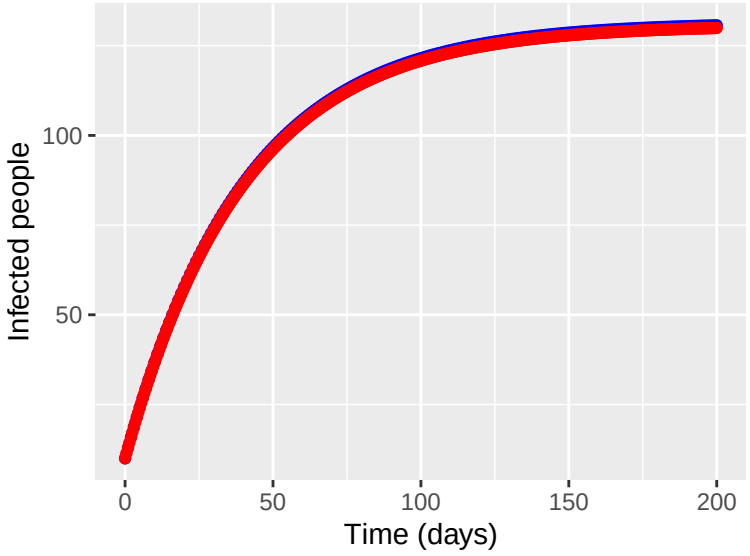


Figure 4.2: Approximation of a solution using local linearity

You may notice that the approximation in Figure 4.2 compares very favorably to the actual solution function. At the end, we have the following comparisons:

t	Euler's Method ($\Delta t = 1$)	Euler's Method ($\Delta t = 0.5$)	Actual Solution
190	130.5	129.7	129
195	130.6	129.8	129.1
200	130.7	129.9	129.2

There is a tradeoff here - the smaller stepsizes you have the more work it will take to compute you solution. You may have seen a similar tradeoff in Calculus when you explored numerical integration and Riemann sums.

4.2 Building an iterative method

Now that we have worked on an example, let's carefully formulate Euler's method with another example. Consider the following equation that describes the rate of change of the spread of a disease (such as Ebola, as we covered in the first section):

$$\frac{dI}{dt} = 0.003I \cdot (4000 - I)$$

Let's call the function $f(I) = 0.03I \cdot (4000 - I)$. In order to numerically approximate the solution, we will need to recall some concepts from calculus. One way that we can approximate the derivative is through a difference function:

$$\frac{dI}{dt} = \lim_{\Delta t \to 0} \frac{I(t + \Delta t) - I(t)}{\Delta t}$$

As long as we consider the quantity Δt to be small (say for this problem 0.1 days if you would like to have units attached to this), we can approximate the derivative with difference function on the right hand side. With this information, we have a reasonable way to organize the problem:

$$\begin{aligned}\frac{I(t+\Delta t)-I(t)}{\Delta t} &= 0.003I\cdot(4000-I)\\ I(t+\Delta t)-I(t) &= 0.003I\cdot(4000-I)\cdot\Delta t\\ I(t+\Delta t) &= I(t)+0.003I\cdot(4000-I)\cdot\Delta t\end{aligned}$$

The last equation $I(t+\Delta t) = I(t)+0.03I\cdot(4000-I)\cdot\Delta t = f(I)\cdot\Delta t$ is a reasonable way to define an iterative system, especially if we have a spreadsheet program. Here is some code in `R` that can define a `for` loop to do this iteratively and then plot the solution:

```
# Define your timestep and time vector
deltaT <- 0.1
t <- seq(0, 2, by = deltaT)

# Define the number of steps we take.  This is equal to 10 / dt (why?)
N <- length(t)

# Define the initial condition
i_approx <- 10

# Define a vector for your solution:the derivative equation
for (i in 2:N) { # We start this at 2 because the first value is 10
  didt <- .003 * i_approx[i - 1] * (4000 - i_approx[i - 1])
  i_approx[i] <- i_approx[i - 1] + didt * deltaT
}


# Define your data for the solution into a tibble:
solution_data <- tibble(
  time = t,
  infected = i_approx
)
```

Next we will plot our solution in Figure 4.3.

```
# Now plot your solution:
ggplot(data = solution_data) +
  geom_line(aes(x = time, y = infected)) +
labs(
  x = "Time",
  y = "Infected"
)
```

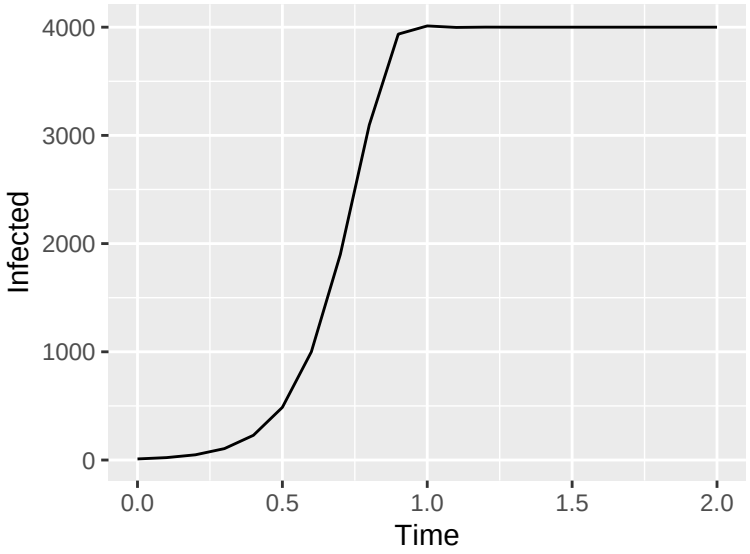


Figure 4.3: An iterative method

Ok, let's break this code down step by step:

- `deltaT <- 0.1` and `t <- seq(0,2,by=deltaT)` define the timesteps (Δt) and the output time vector `t`. We also define `N <- length(t)` so we know how many steps we take.
- `i_approx <- 10` defines the initial condition to the system, in other words $I(0) = 10$.
- The `for` loop goes through this system - first computing the value of $\frac{dI}{dt}$ and then forecasing out the next timestep $I(t + \Delta t) = f(I) \cdot \Delta t$
- The remaining code plots the dataframe, like we learned in Section 2.

Let's recap what we've learned to summarize Euler's method. The most general form of a differential equation is:

$$\frac{d\vec{y}}{dt} = f(\vec{y}, \vec{\alpha}, t),$$

where $\vec{y}$ is the vector of state variables you want to solve for, and $\vec{\alpha}$ is your vector of parameters.

At a given initial condition, Euler's method applies locally linear approximations to forecast the solution forward Δt time units:

$$\vec{y}_{n+1} = y_n + f(\vec{y}_n, \vec{\alpha}, t_n) \cdot \Delta t$$

To generate Figure 4.3 we created the solution directly in `R` - but you don't want to copy and paste the code. The `demodeler` package has a function called `euler` that does the same process to generate the output solution:

```
library(demodelr)

# Define the rate equation:
infection_eq <- c(didt ~ .003 * i * (4000 - i))

# Define the initial condition (as a named vector)
infection_init <- c(i = 10)

# Define deltaT and the time steps:
deltaT <- 0.2
n_steps <- 10

# Compute the solution via Euler's method:
out_solution <- euler(system_eq = infection_eq,
                      initial_condition = infection_init,
```

```
                      deltaT = deltaT,
                      n_steps = n_steps
                      )


# Now plot your solution:
ggplot(data = out_solution) +
  geom_line(aes(x = t, y = i)) +
labs(
  x = "Time",
  y = "Infected"
)
```

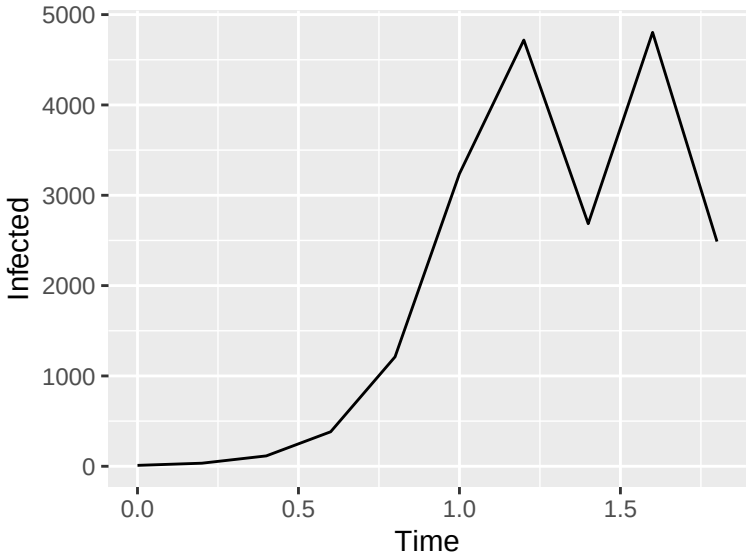


Figure 4.4: Euler's method solution

Let's talk through the steps of this code as well:

- The line `infection_eq <- c(didt ~ .003 * i * (4000-i))` represents the differential equation, written in formula notation. So $\frac{dI}{dt} \rightarrow$ `didt` and $f(I) \rightarrow$ `.003 * i * (4000-i))`, with the variable `i`.
- The initial condition $I(0) = 10$ is written as a **named vector:** `infection_init <- c(i=10)`. Make sure the name of the variable is consistent with your differential equation.
- As before we need to identify Δt and the number of steps N.

The command `euler` then computes the solution applying Euler's method, returning a data frame so we can plot the results. Note the columns of the data frame are the variables t and i that have been named in our equations.

4.3 Euler's method applied to systems

Now that we have some experience with Euler's method, let's see how we can apply the function `euler` to a system of differential equations. Here is a sample code that shows the dynamics for the Lotka-Volterra equations, as studied in Section 3:

$$\begin{aligned} \frac{dH}{dt} &= rH - bHL \\ \frac{dL}{dt} &= ebHL - dL \end{aligned} \tag{4.1}$$

We are going to use Euler's method to solve this differential equation. Similar to the previous example we will need to determine the f

```r
# Define the rate equation:
lynx_hare_eq <- c(
  dHdt ~ r * H - b * H * L,
  dLdt ~ e * b * H * L - d * L
)


# Define the parameters (as a named vector)
lynx_hare_params <- c(r = 2, b = 0.5, e = 0.1, d = 1) # parameters: a named vector

# Define the initial condition (as a named vector)
lynx_hare_init <- c(H = 1, L = 3)

# Define deltaT and the time steps:
deltaT <- 0.05 # timestep length
timeSteps <- 200 # must be a number greater than 1

# Compute the solution via Euler's method:
out_solution <- euler(system_eq = lynx_hare_eq,
                      parameters = lynx_hare_params,
                      initial_condition = lynx_hare_init,
                      deltaT = deltaT,
                      n_steps = n_steps
                      )

# Make a plot of the solution, using different colors for lynx or hares.
ggplot(data = out_solution) +
  geom_line(aes(x = t, y = H), color = "red") +
  geom_line(aes(x = t, y = L), color = "blue") +
  labs(
    x = "Time",
    y = "Lynx (red) or Hares (blue)"
  )
```

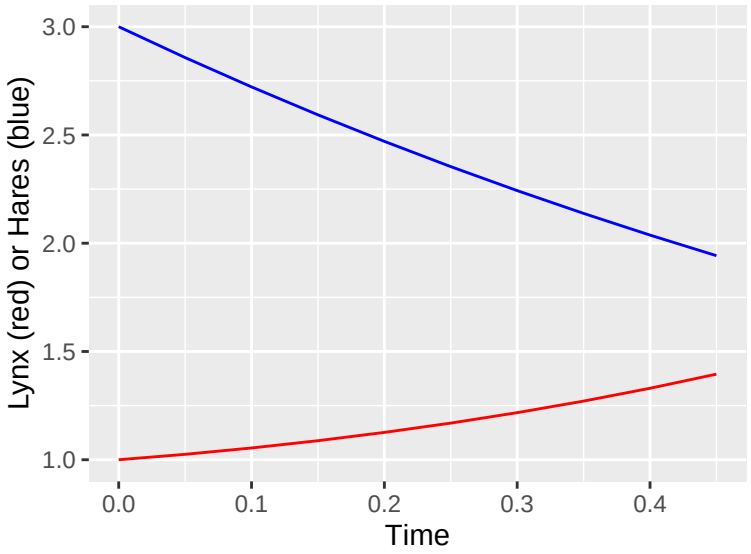


Figure 4.5: Euler's method solution for Lynx-Hare system

This example is structured similarly as a single variable differential equation, with some key changes:

- The variable `lynx_hare_eq` is now a vector, with each entry one of the rate equations.
- We need to identify both variables in their initial condition.

- Most importantly, Equation (4.1) has parameters, which we define as a named vector `lynx_hare_params <- c(r = 2, b = 0.5, e = 0.1, d = 1)` that we pass through to the command `euler` with the option `parameters`. If your equation does not have any parameters you do not need to worry about specifying this input.
- We plot both solutions together at the end, or you can make two separate plots. Remember that you can choose the color in your plot.

Thankfully the `euler` code is pretty easy to adapt for systems of equations!

4.4 More refined numerical solvers

Perhaps in the course of working with Euler's method you encounter a differential equation that produces some nonsensible results. Take for example the following which is the implementation of our quarantine model:

```
quarantine_eq <- c(
  dSdt ~ -k * S * I,
  dIdt ~ k * S * I - beta * I
)


deltaT <- .1 # timestep length
timeSteps <- 15 # must be a number greater than 1

quarantine_parameters <- c(k = .05, beta = .2) # parameters: a named vector

quarantine_init <- c(S = 300, I = 1) # Be sure you have enough conditions as you do variables.

# Compute the solution via Euler's method:
out_solution <- euler(system_eq = quarantine_eq,
                      parameters = quarantine_parameters,
                      initial_condition = quarantine_init,
                      deltaT = deltaT,
                      n_steps = n_steps
                      )


ggplot(data = out_solution) +
  geom_line(aes(x = t, y = S), color = "red") +
  geom_line(aes(x = t, y = I), color = "blue") +
  labs(
    x = "Time",
    y = "Susceptible (red) or Infected (blue)"
  )
```

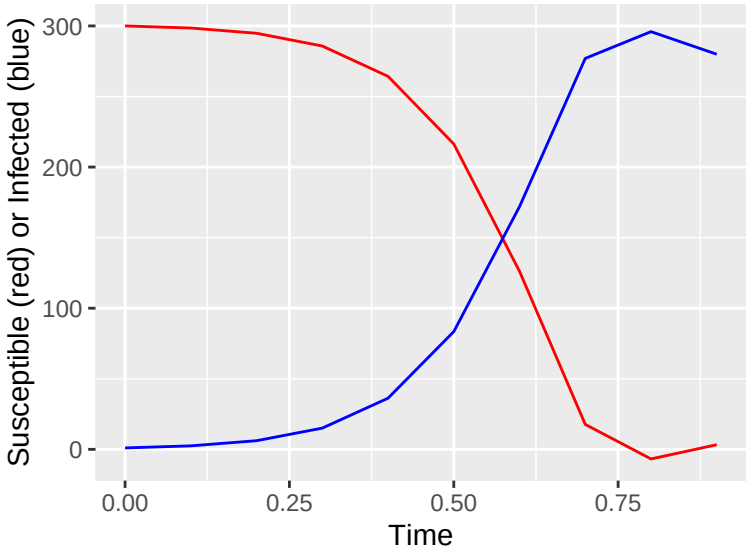


Figure 4.6: Surprising results with Euler's method.

You may notice the solution for S wiggles around $t = 0.75$ and is negative. This is concerning because we know there can't be negative people! This requires a little more investigation.

If we take a look at $t = 0.75$ the value for $S \approx 1$ and the value for $I \approx 280$. If we let $k = 0.05$ and $\beta = 0.2$, this means that $\frac{dS}{dt} = -14$ and $\frac{dI}{dt} = -42$. The values of S and I are both decreasing! We know that our Euler's method update is one where the new value is the old value plus any change. So the new value for $S = 1 - 14 \cdot 0.1 = -0.4$. Mathematically, Euler's method working correctly, but we know realistically that neither S or I can be negative.

In turns out that this can easily be overcome. While Euler's method is useful, it does quite poorly in cases where the solution is changing rapidly - or we might need to make some smaller step sizes.

Another way to remedy this is to use a *higher order solver*, and one such method is called the Runge-Kutta Method. If you take a course in numerical analysis you might study these, but for the moment you see the difference between the Runge-Kutta solver implemented in the `demodelr` package, which by all intents and purposes is replaces the command `euler` with `rk4`:

```
quarantine_eq <- c(
  dSdt ~ -k * S * I,
  dIdt ~ k * S * I - beta * I
)


deltaT <- .1 # timestep length
timeSteps <- 15 # must be a number greater than 1

quarantine_parameters <- c(k = .05, beta = .2) # parameters: a named vector

quarantine_init <- c(S = 300, I = 1) # Be sure you have enough conditions as you do variables.

# Compute the solution via a Runge-Kutta method:
out_solution <- rk4(system_eq = quarantine_eq,
                    parameters = quarantine_parameters,
                    initial_condition = quarantine_init,
                    deltaT = deltaT,
                    n_steps = n_steps
                    )


ggplot(data = out_solution) +
  geom_line(aes(x = t, y = S), color = "red") +
```

```
  geom_line(aes(x = t, y = I), color = "blue") +
  labs(
    x = "Time",
    y = "Susceptible (red) or Infected (blue)"
  )
```

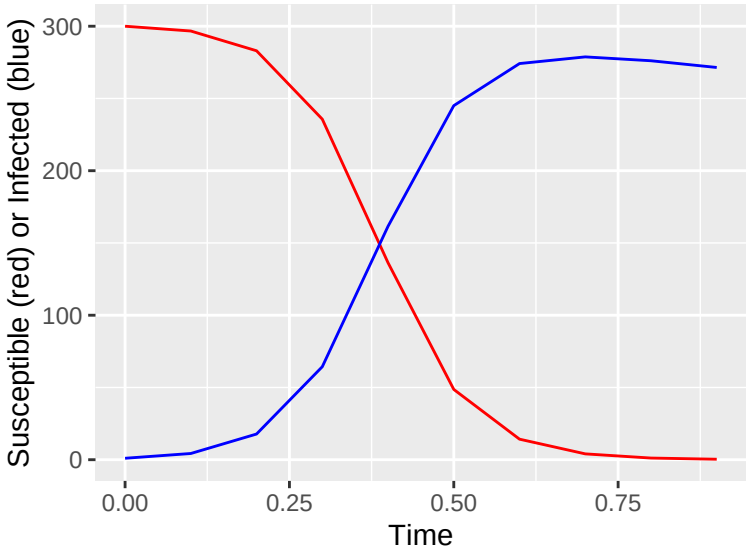


Figure 4.7: Better results with the Runge-Kutta method.

So what is going on here? Briefly, the differences between the two methods have to do with the error in the numerical method. The error is quantified as the difference between the actual solution and the numerical solution. Euler's method has an error on the order of the stepsize Δt, whereas the Runge-Kutta method has an error of $(\Delta t)^4$. For this example, the error in Euler's method is 0.1 ($\Delta t = .1$), but for the Runge-Kutta method it is 0.0001 ($(\Delta t)^4 = .0001$). The error is noticeably different! While we can improve Euler's method by taking a smaller timestep - BUT that means we need a larger number of steps N - which may take more computational time.

Does this discussion sound familiar? Perhaps you examined similar when you took calculus and studied Riemann sums to approximate the area underneath a curve (left sum, right sum, trapezoid, midpoint). It turns out that these two problems are closely related. Numerical analysis is a great field of study to examine these topics and others!

Moving ahead, I might switch between Euler's method or the Runge-Kutta method when solving a differential equation. Thankfully the bulk of the work is "under the hood" - the setup will be the same for both!

4.5 Exercises

Exercise 4.1. Verify that $I(t) = 130 - 120e^{-0.25t}$ is a solution to the differential equation

$$\frac{dI}{dt} = 130 - 0.025I$$

with $I(0) = 10$.

Exercise 4.2. Apply the `rk4` solver with $\Delta t = 0.1$ with $N = 10$ to the initial value problem $\frac{dI}{dt} = 0.003I \cdot (4000 - I)$ $I(0) = 10$. Compare your graph to Figure 4.3. What differences do you observe? Which solution method (`euler` or `rk4`) is better (and why)?

Exercise 4.3. The following exercise will help you explore the relationships between stepsize, ending points, and number of steps needed. You may assume that we will start at $t = 0$ in all parts.

a. If we wish to do an Euler's method solution with step size 1 second and ending at $t = 5$ seconds, how many steps will we take?
b. If we wish to do an Euler's method solution with step size 0.5 seconds and ending at $t = 5$ seconds, how many steps will we take?
c. If we wish to do an Euler's method solution with step size 0.1 seconds and ending at $t = 5$ seconds, how many steps will we take?
d. If we wish to do an Euler's method solution with step size Δt and go to ending value of T, what is an expression that relates the number steps N as a function of Δt and T?

Exercise 4.4. To get a rough approximation between error and step size, let's say for a particular differential equation that we are starting at $t = 0$ and going to $t = 2$, with $\Delta t = 0.2$. We know that the Runge-Kutta error will be on the order of $(\Delta t)^4 = 0.0016$. If we want to use Euler's method with the same order of error, we could say $\Delta t = .0001$. For that case, how many steps will we need to take?

Exercise 4.5. For each of the following differential equations, apply Euler's method to generate a numerical solution to the differential equation and plot your solution. The stepsize (Δt) and number of iterations (N) are listed.

a. Differential equation: $\frac{dS}{dt} = 3 - S$. Set $\Delta t = 0.1$, $N = 50$. Initial conditions: $S(0) = 0.5$, $S(0) = 5$.
b. Differential Equation: $\frac{dS}{dt} = \frac{1}{1 - S}$. Set $\Delta t = 0.01$, $N = 30$. Initial conditions: $S(0) = 0.5$, $S(0) = 2$.
c. Differential equation: $\frac{dS}{dt} = 0.8 \cdot S \cdot (10 - S)$. Set $\Delta t = 0.1$, $N = 50$. Initial conditions: $S(0) = 3$, $S(0) = 10$.

Exercise 4.6. For each of the following differential equations, apply the Runge-Kutta method method to generate a numerical solution to the differential equation and plot your solution. The stepsize (Δt) and number of iterations (N) are listed. Contrast your answers with Exercise 4.5.

a. Differential equation: $\frac{dS}{dt} = 3 - S$. Set $\Delta t = 0.1$, $N = 50$. Initial conditions: $S(0) = 0.5$, $S(0) = 5$.
b. Differential Equation: $\frac{dS}{dt} = \frac{1}{1 - S}$. Set $\Delta t = 0.01$, $N = 30$. Initial conditions: $S(0) = 0.5$, $S(0) = 2$.
c. Differential equation: $\frac{dS}{dt} = 0.8 \cdot S \cdot (10 - S)$. Set $\Delta t = 0.1$, $N = 50$. Initial conditions: $S(0) = 3$, $S(0) = 10$.

Exercise 4.7. Complete the following steps:

a. Apply the code `euler` to generate a numerical solution to the differential equation:

- Differential equation: $\frac{dS}{dt} = r \cdot S \cdot (K - S)$.
- Set $r = 1.2$ and $K = 3$.

- Set $\Delta t = 0.1$, $N = 50$.
- Initial conditions (three different ones): $S(0) = 1$, $S(0) = 3$, $S(0) = 5$.

b. Plot your Euler's method solutions with the three initial conditions on the same plot. What do you notice when you do plot them together?
c. Make a hypothesis regarding the long term behavior of this system. Then plot a few more solution curves to verify your guess.

Exercise 4.8. Complete the following steps:

a. Apply the code `euler` to generate a numerical solution to the differential equation:

- Differential equation: $\dfrac{dS}{dt} = K - S$.
- Set $K = 2$.
- Set $\Delta t = 0.1$, $N = 50$.
- Initial conditions (three different ones): $S(0) = 0$, $S(0) = 2$, $S(0) = 5$.

b. Plot your Euler's method solutions with the three initial conditions on the same plot. What do you notice when you do plot them together?
c. Make a hypothesis regarding the long term behavior of this system. Then plot a few more solution curves to verify your guess.

Exercise 4.9. Let's do some more work with Euler's method for $\dfrac{dS}{dt} = 0.8 \cdot S \cdot (10 - S)$. This time set $S(0) = 15$, $\Delta t = 0.1$, $N = 10$. When you examine your solution, what is incorrect about the Euler's method solution based on your qualitative knowledge of the underlying dynamics? Now calculate Euler's method for the same differential equation for the following conditions: $S(0) = 15$, $\Delta t = 0.01$, $N = 100$. What has changed in your solution?

Exercise 4.10. Let's do some more work with Euler's method for $\dfrac{dS}{dt} = \dfrac{1}{1 - S}$. This time set $S(0) = 1.5$, $\Delta t = 0.1$, $N = 10$ and also $S(0) = 1.5$, $\Delta t = 0.01$, $N = 100$. Between these two solutions, what has changed? Do you think it is numerically possible to calculate a reasonable solution for Euler's method near $S = 1$? (*note: this differential equation is an example of finite time blow up*)

Exercise 4.11. Similar to Exercise 4.10, let's apply the `rk4` method for $\dfrac{dS}{dt} = \dfrac{1}{1 - S}$. This time set $S(0) = 1.5$, $\Delta t = 0.1$, $N = 10$ and also $S(0) = 1.5$, $\Delta t = 0.01$, $N = 100$. Between these two solutions, what has changed? Does this numerical solver do a better job in computing solutions compared to the Euler method?

Exercise 4.12. One way to model the growth rate hares is with $f(H) = \dfrac{rH}{1 + kH}$, where r and k are parameters. This is in constrast to exponential growth, which assumes $f(H) = rH$.

a. First evaluate $\lim\limits_{H \to \infty} rH$.
b. Then $\lim\limits_{H \to \infty} \dfrac{rH}{1 + kH}$.
c. Compare your two answers. Discuss how the growth rate $f(H) = \dfrac{rH}{1 + kH}$ seems to be a more realistic model.

Exercise 4.13. In the lynx hare example we can also consider an alternative system where the growth of the hare is not exponential:

$$\begin{aligned} \frac{dH}{dt} &= \frac{2H}{1 + kH} - 0.5HL \\ \frac{dL}{dt} &= 0.05HL - dL \end{aligned} \tag{4.2}$$

Set the number of timesteps to be 2000. Apply Euler's method to numerically solve this system of equations when $k = 0.1$ and $k = 1$. Plot your simulation results.

Exercise 4.14. Consider the differential equation $\frac{dS}{dt} = \frac{1}{1-S}$. Notice that at $S = 1$ the rate $\frac{dS}{dt}$ is not defined.

a. If you applied Euler's method solution with initial condition $S(0) = 0.9$, what do you think your solution would approach as the number of timesteps increased?
b. If you applied Euler's method solution with initial condition $S(0) = 1.1$, what do you think your solution would approach as the number of timesteps increased?
c. Explain how you could come to the same conclusion as the previous two problems if you graphed $f(S) = \frac{1}{1-S}$.

Exercise 4.15. Building on Exercise 4.4, let's say for a particular differential equation we have N steps from $0 \leq t \leq b$. An error of ϵ is desired.

a. What is the ratio $\frac{N_E}{N_{RK4}}$, where N_{RK4} represents the number of steps needed for the Runge-Kutta method, and N_E the number of steps for Euler's method?
b. Make a plot of the ratio $\frac{N_E}{N_{RK4}}$ for $0 \leq \epsilon \leq 1$. How many more steps does Euler's method need to do to achieve the same level of error, compared to the Runge-Kutta method?

Chapter 5

Phase Lines and Equilibrium Solutions

In modeling with differential equations, we want to understand how a system develops both qualitatively and quantitatively. Euler's method (and other associated numerical methods for solving differential equations) illustrate solution behavior numerically. One key thing about the qualitative analysis is we are interested in the *motion* or the "flow" of the solution at a given point. Is the solution increasing, decreasing, or staying the same?

For this section we will discuss qualitative aspects of a differential equation. We are going to focus on differential equations in one variable. Section 6 will build your understanding of the same idea with coupled systems of equations.

5.1 Equilibrium solutions

A great place to start is where the rate of change for a differential equation is zero, or in other words there is *no* flow. Borrowing ideas from calculus, this occurs when the rate of change is zero. We solve this by setting the left hand side of $\frac{dy}{dt} = f(y)$ equal to zero and solving for y (or whatever dependent variable describes the problem).

Example 5.1. What are the equilibrium solutions to $\frac{dy}{dt} = -y$?

Solution. For this example we know that when the rate of change is zero, this means that $\frac{dy}{dt} = 0$, or when $0 = -y$. So $y = 0$ is the equilibrium solution.

The general solution to the differential equation $\frac{dy}{dt} = -y$ is when $y(t) = Ce^{-t}$, where C is an arbitrary constant. Figure 5.1 plots different solution curves, with the equilibrium solution shown as a horizontal line:

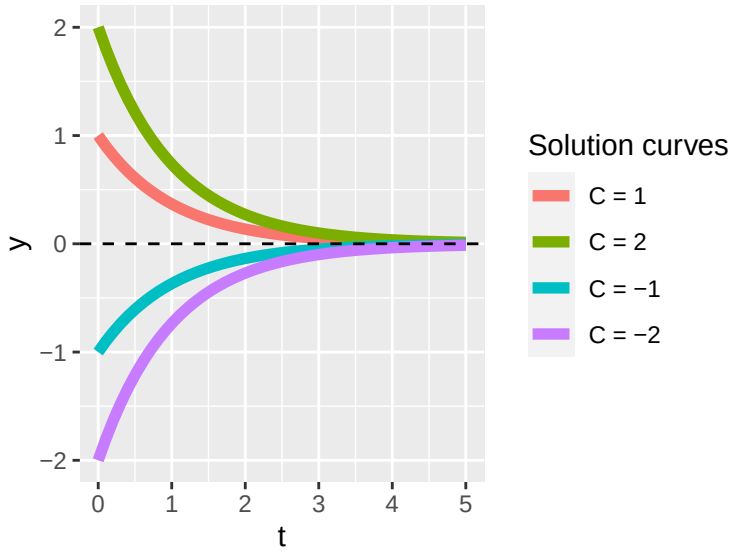


Figure 5.1: Solution curves to $y' = -y$

Notice that as t increases, all solutions approach the equilibrium solution $y = 0$ as t no matter if the initial condition is positive or negative. A second way that we can verify that solutions approach the equilibrium solution is by evaluate the following limit: $\lim_{t\to\infty} Ce^{-t} = 0$.

Example 5.2. What are the equilibrium solutions to $\dfrac{dN}{dt} = N \cdot (1 - N)$?

Solution. In this case the equilibrium solutions occur when $N \cdot (1 - N) = 0$, or when $N = 0$ or $N = 1$.

The generic solution to this differential equation is

$$N(t) = \frac{N_0}{N_0 + (1 - N_0)e^{-t}}.$$

Figure 5.2 displays several different solution curves.

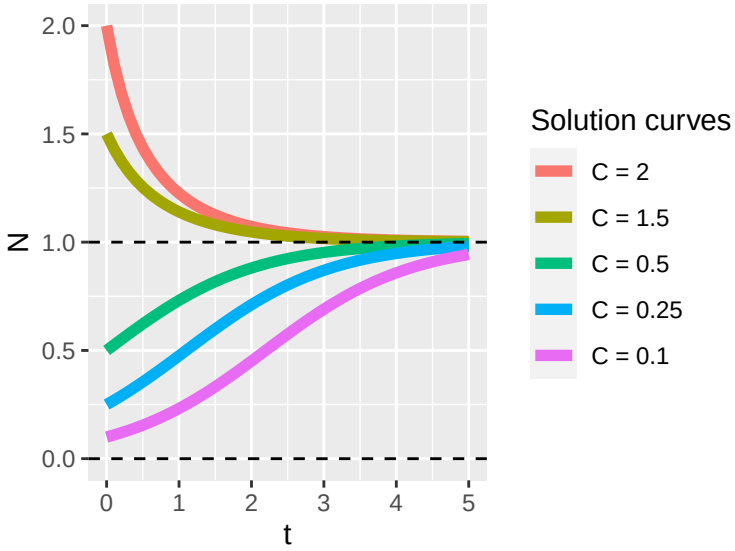


Figure 5.2: Solution curves for $N' = N(1 - N)$

In Figure 5.2 notice how all the solutions tend towards $N = 1$, but even solutions that start close to $N = 0$ seem to move away from this equilibrium solution. This example brings us to understanding classifying the *stability* of the equilibrium solutions.

5.2 Phase lines for differential equations

While it is one thing to determine where the equilibrium solutions are, we are also interested in classifying the **stability** of the equilibrium solutions. To do this investigate the behavior of the differential equation around the equilibrium solutions, using facts from calculus:

- If $\dfrac{dy}{dt} < 0$, the solution $y(t)$ is decreasing.
- If $\dfrac{dy}{dt} > 0$, the solution $y(t)$ is increasing.

Let's apply this logic to our differential equation $\dfrac{dy}{dt} = -y$. We know that if $y = 3$, $\dfrac{dy}{dt} = -3 < 0$, so we say the function is *decreasing* to $y = 0$. If $y = -2$, $\dfrac{dy}{dt} = -(-2) = 2 > 0$, so we say the function is *increasing* to $y = 0$. This can be represented neatly in the phase line diagram for Figure 5.3:

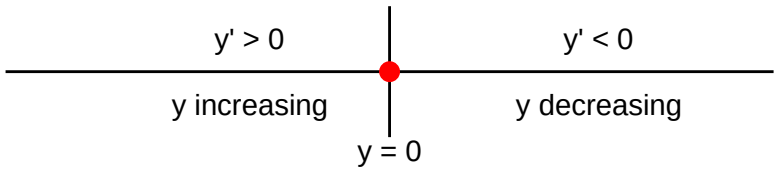


Figure 5.3: Phase line to $y' = -y$.

Because the solution is *increasing* to $y = 0$ when $y < 0$, and *decreasing* to $y = 0$ when $y > 0$, we say that the equilibrium solution is **stable**, which is also confirmed by the solutions we plotted above.

Example 5.3. Classify the stability of the equilibrium solutions to $\frac{dy}{dt} = k \cdot y$, where k is a parameter.

Solution. In this case the equilibrium solution is still $y = 0$. We will need to consider three different cases for the stability depending on the value of k ($k > 0$, $k < 0$, and $k = 0$):

- When $k < 0$, the phase line will be similar Figure 5.3.
- When $k > 0$ the phase line will be is shown in the Figure 5.4. We say in this case that the equilibrium solution is *unstable*, as all solutions flow away from the equilibrium. Several different solutions are shown in Figure 5.5
- When $k = 0$ we have the differential equation $\frac{dy}{dt} = 0$, which has $y = C$ as a general solution. For this special case the equilibrium solution is neither stable or unstable[1].

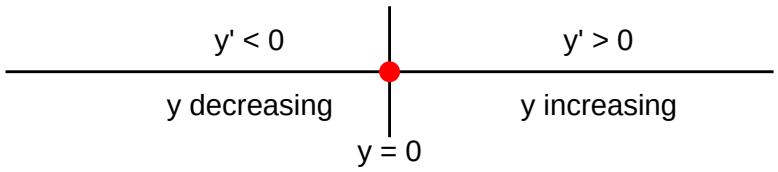


Figure 5.4: Phaseline for $y' = ky$, with $k > 0$.

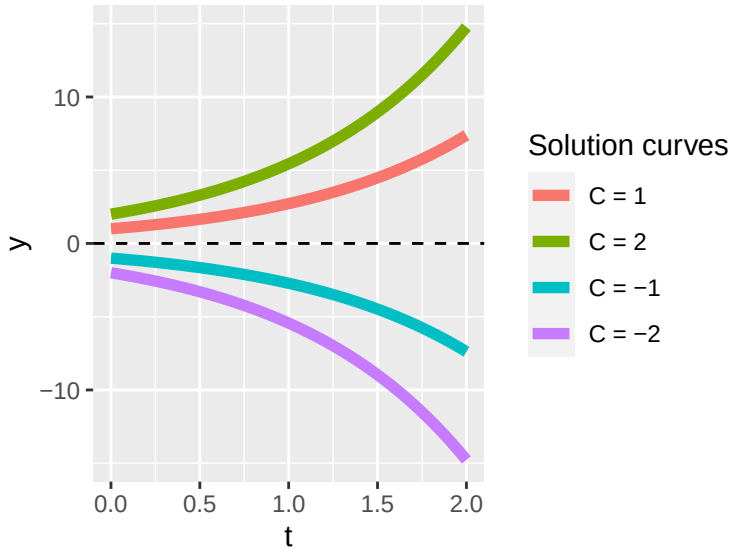


Figure 5.5: Solution curves for $y' = ky$, with $k > 0$.

Example 5.4. Let's investigate the phase line for the differential equation $\frac{dN}{dt} = N \cdot (1 - N)$ and classify stability of the equilibrium solutions.

[1]By all intents and purposes this is a different differential equation than $\frac{dy}{dt} = k \cdot y$; something peculiar is going on here - which we come back to when discuss bifurcations in Section 20.

Solution. This differential equation has equilbrium solutions when $N(1-N)=0$, or $N=0$ or $N=1$. We evaluate the stability of the solutions in the following table:

Test point	**Sign of N'**	**Tendency of solution**
$N=-1$	Negative	Decreasing
$N=0$	Zero	Equilibrium solution
$N=0.5$	Positive	Increasing
$N=1$	Zero	Equilibrium solution
$N=2$	Negative	Decreasing

Notice how the selected test points in the first column are either the the *left* or the *right* of the equilbrium solution. We can also represent the information in the table using a phase line diagram (Figure 5.6), but in this case we need to include *two* equilibrium solutions.

The table and Figure 5.6 confirms that N is moving *away* from $N=0$ (either decreasing when N is less than 0 and increasing when N is greater than 0) and moving *towards* $N=1$ (either increasing when N is between 0 and 1 and decreasing when N is greater than one.

These results suggest that equilibrium solution at $N=0$ to be *unstable* and at $N=1$ to be *stable*.

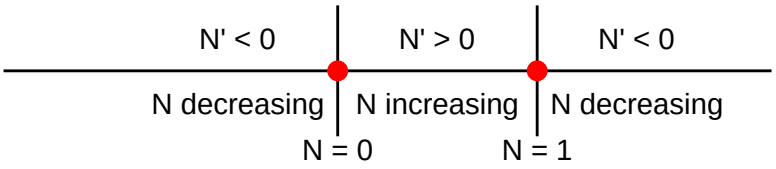


Figure 5.6: Phase line diagram for the differential equation $N'=N(1-N)$.

Other than writing the words in the phase line diagram, we also use arrows to signify increasing or decreasing in the solutions.

5.3 A stability test for equilibrium solutions

Notice how when constructing the phase line diagram we relied on the behavior of solutions *around* the equilibrium solution to classify the stability. As an alternative we can also use the point at the equilibrium solution itself.

To do this we are going to consider the general differential equation $\frac{dy}{dt}=f(y)$. We are going to assume that we have an equilibrium solution at $y=y_*$.

We use local linearization to construct a locally linear approximation to $L(y)$ to $f(y)$ at $y=y_*$:

$$L(y)=f(y_*)+f'(y_*)\cdot(y-y_*) \tag{5.1}$$

Once we have the local linearization, there are two follow-on steps to simplify Equation (5.1). First, because we have an equilibrium solution, $f(y_*)=0$. Second, Equation (5.1) can be written with a new variable P, defined by variable $P=y-y_*$. Then Equation (5.1) translates to

$$\frac{dP}{dt}=f'(y_*)\cdot P \tag{5.2}$$

Does Equation (5.2) look familiar? It should! This equation is similar to the example where we classified the stability of $\frac{dy}{dt}=k\cdot y$ – cool! So let's use what we learned in Example 5.3 above to classify the stability:

Local linearization stability test for equilibrium solutions: For a differential equation $\frac{dy}{dt}=f(y)$ with equilibrium solution y_*, we can classify the stability of the equilibrium solution through the following:

- If $f'(y_*)'>0$ at an equilibrium solution, the equilibrium solution $y=y_*$ will be *unstable*.

- If $f'(y_*) < 0$ at an equilibrium solution, the equilibrium solution $y = y_*$ will be *stable*.
- If $f'(y_*) = 0$, we cannot conclude anything about the stability of $y = y_*$.

Example 5.5. Apply local linearization to classify the stability of the equilibrium solutions of $\dfrac{dN}{dt} = N \cdot (1 - N)$

Solution. The locally linear approximation is $L(N) = 1 - 2N$. We have $L(0) = 1 > 0$, so $N = 0$ is unstable. Similarly $L(1) = -1$, so $N = 1$ is stable.

5.4 Exercises

Exercise 5.1. What are the equilibrium solutions to the following differential equations?

a. $\frac{dS}{dt} = 0.3 \cdot (10 - S)$
b. $\frac{dP}{dt} = P \cdot (P - 1)(P - 2)$

Then classify the stability of the equilbrium solutions using the local linearization stability test.

Exercise 5.2. Using your results from Exercise 5.1, construct a phase line for each of the differential equations and classify the stability of the equilibrium solutions.

Exercise 5.3. A population grows according to the equation $\frac{dP}{dt} = \frac{P}{1+2P} - 0.2P$.

a. Determine the equilibrium solutions for this differential equation.
b. Classify the stability of the equilibrium solutions using the local linearization stability test.

Exercise 5.4. (Inspired by Logan and Wolesensky (2009)) A cell with radius r assimilates nutrients at a rate proportional to its surface area, but uses nutrients proportional to its volume, according to the following differential equation:

$$\frac{dr}{dt} = 4\pi r^2 - \frac{4}{3}\pi r^3.$$

a. Determine the equilibrium solutions for this differential equation.
b. Construct a phase line for this differential equation to classify the stability of the equilibrium solutions.

Exercise 5.5. (Inspired by Thornley and Johnson (1990)) The Chanter equation of growth is the following, where W is the weight of an object:

$$\frac{dW}{dt} = W(3 - W)e^{-Dt}, \tag{5.3}$$

Use this differential equation to answer the following questions.

a. What happens to the rate of growth ($\frac{dW}{dt}$) as t grows large?
b. What are the equilibrium solutions to this model? Are they stable or unstable?
c. Notice how the equilbrium solutions are the same as those for the logistic model. Based on your previous work, do why would this model be a more realistic model of growth than the logistic model $\frac{dW}{dt} = W(3 - W)$?

Exercise 5.6. Red blood cells are formed from stem cells in the bone marrow. The red blood cell density r satisfies an equation of the form

$$\frac{dr}{dt} = \frac{br}{1 + r^n} - cr, \tag{5.4}$$

where $n > 1$ and $b > 1$ and $c > 0$. Find all the equilibrium solutions r_* to this differential equation. *Hint:* can you factor an r from your equation first?

Exercise 5.7. (Inspired by Hugo van den Berg (2011)) Organisms that live in a saline environment biochemically maintain the amount of salt in their blood stream. An equation that represents the level of S in the blood is the following:

$$\frac{dS}{dt} = I + p \cdot (W - S)$$

Where the parameter I represents the active uptake of salt, p is the permeability of the skin, and W is the salinity in the water.

a. First set $I = 0$. Determine the equilibrium solutions for this differential equation. Express your answer S_* in terms of the parameters p, and W.
b. Next consider $I > 0$. Determine the equilibrium solutions for this differential equation. Express your answer S_* in terms of the parameters p, W, and I. Why should your new equilbrium solution be greater than the equilibrium solution from the previous problem?
c. Classify the stability of both equilibrium solutions using the local linearization stability test.

Exercise 5.8. (Inspired by Logan and Wolesensky (2009)) The immigration rate of bird species (species per time) from a mainland to an offshore island is $I_m \cdot (1 - S/P)$, where I_m is the maximum immigration rate, P is the size of the source pool of species on the mainland, and S is the number of species already occupying the island. Further, the extinction rate is $E \cdot S/P$, where E is the maximum extinction rate. The growth rate of the number of species on the island is the immigration rate minus the extinction rate, given by the following differential equation:

$$\frac{dS}{dt} = I_m \left(1 - \frac{S}{P}\right) - \frac{ES}{P} \tag{5.5}$$

a. Determine the equilibrium solutions S_* for this differential equation. Expression your answer in terms of I_M, P, and E.
b. Classify the stability of the equilibrium solutions using the local linearization stability test.

Exercise 5.9. A colony of bacteria growing in a nutrient-rich medium deplete the nutrient as they grow. As a result, the nutrient concentration $x(t)$ is steadily decreasing. The equation describing this decrease is the following:

$$\frac{dx}{dt} = -\mu \frac{x \cdot (\xi - x)}{\kappa + x},$$

where μ, κ, and ξ are all parameters greater than zero.

a. Determine the equilibrium solutions x_* for this differential equation.
b. Construct a phase line for this differential equation and classify the stability of the equilibrium solutions.

Exercise 5.10. Can a solution curve cross an equilibrium solution of a differential equation?

Chapter 6

Coupled Systems of Equations

In this section we will learn how to qualitatively understand systems of differential equations. When analyzing a single differential equation we used the idea of a phase line to understand if a solution was stable or unstable. Here we extend that to equations of more than one variable and investigate what we will call the *phase plane*.

6.1 Model redux: flu with quarantine

In Section 1 we studied the following model for the flu as a coupled system of equations:

$$\begin{aligned} \frac{dS}{dt} &= -kSI \\ \frac{dI}{dt} &= kSI \end{aligned} \tag{6.1}$$

In this scenario we are also going to consider that those who are infected are quarantined, proportional to the number infected, according to the following schematic:

Which gives us the following system of equations:

$$\begin{aligned} \frac{dS}{dt} &= -kSI \\ \frac{dI}{dt} &= kSI - \beta I \end{aligned} \tag{6.2}$$

To find the equilibrium solutions we want to find values of S and I where the rates $\frac{dS}{dt}$ and $\frac{dI}{dt}$ are *both* zero. This can be done by algebraically solving the system of equations:

$$\begin{aligned} 0 &= -kSI \\ 0 &= kSI - \beta I \end{aligned} \tag{6.3}$$

Let's examine the first equation ($0 = -kSI$), which we can see is consistent when either $S = 0$ and $I = 0$. These give us two options, which we then use in the second equation ($0 = kSI - \beta I$). When $S = 0$, then $0 = k \cdot 0 \cdot I - \beta I \to 0 = -\beta I$, which is consistent when $I = 0$. So $(S_*, I_*) = (0, 0)$ is one equilibrium solution. The other case when $I = 0$ gives us that any value of S would be an equilibrium solution. Can you explain why?

Equation (6.2) and its equilibrium solutions are an interesting example. We call the equations $S = 0$ and $I = 0$ when $\frac{dS}{dt} = 0$ as *nullclines* for S. In a similar manner, the equations in S and I when $\frac{dI}{dt} = 0$ are called *nullclines* for I. Let's try to determine formulas for these equations:

$$\begin{aligned} 0 &= kSI - \beta I \\ 0 &= I \cdot (kS - \beta) \end{aligned} \tag{6.4}$$

Because the last equation is factored as a product, nullclines for I are either $I = 0$ or $S = \frac{\beta}{k}$.

Nullclines are not equilibrium solutions by themselves - it is the *intersection* of two different nullclines that determine equilibrium solutions. Figure 6.1 shows the nullclines in the $S - I$ plane (since we have two equations), with S on the horizontal axis and I on the vertical axis.

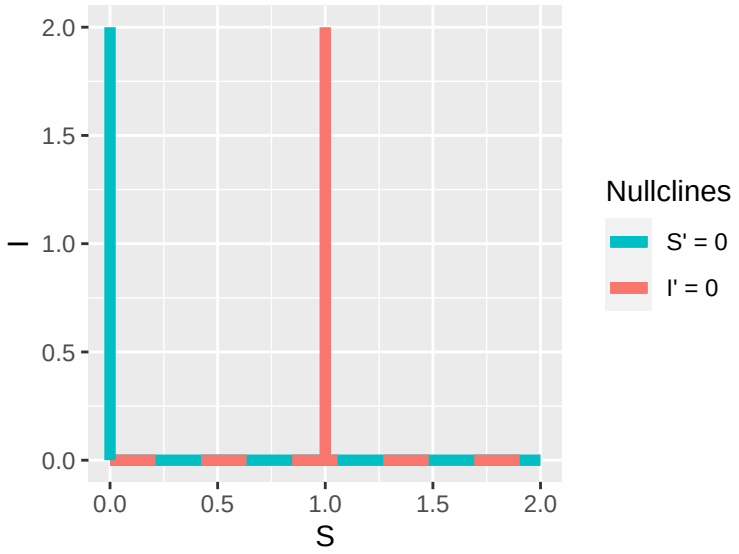


Figure 6.1: Nullclines for Equation (6.2). To generate the plot we assumed $\beta = 1$ and $k = 1$

A key thing to note is that where two different nullclines cross is an *equilibrium solution* to the system of equations. This means that **both** $\frac{dS}{dt}$ and $\frac{dI}{dt}$ are zero at this point. Examining Figure 6.1, there are three possibilities:

1. There is an equilibrium solution at $S = 0$ and $I = 0$ (otherwise known as the origin). This equilibrium solution makes biological sense: if there is nobody susceptible or infected there are no flu cases (everyone is perfectly healthy - yay!) .
2. The entire horizontal axis is an equilibrium solution because $I = 0$, which makes both $\frac{dS}{dt}$ and $\frac{dI}{dt}$ both zero. There is a practical interpretation of this nullcline - whenever $I = 0$, meaning there are no infected people around, so infection cannot occur.
3. There is also a third possibility where the vertical line at $S = 1$ crosses the horizontal axis ($S = 1$, $I = 0$), but that also falls under the second equilibrium solution.

Now that we have identified our nullclines and equilibrium solutions, we will add additional context with the *flow* of the solution.

6.1.1 Adding context to our phase plane: slope fields

Let's go back to the idea of a phase plane, but this time we are going to add more context to our nullcline graph by evaluating different values of S and I into our system of equations and plot the *slope field*.

Table 6.1 evaluates the derivatives $\frac{dS}{dt}$ and $\frac{dI}{dt}$ in Equation (6.2) for different values of S and I.

Notice how the different values of $\frac{dS}{dt}$ and $\frac{dI}{dt}$ at each of the S and I values. We can plot each of the coordinate pairs of $\left(\frac{dS}{dt}, \frac{dI}{dt}\right)$ as a vector in the (S, I) plane. We associate $\frac{dS}{dt}$ with left-right motion, so positive $\frac{dS}{dt}$ means pointing to the right. Likewise, we associate $\frac{dI}{dt}$ with up-down motion, so positive $\frac{dI}{dt}$ means the vector points up. At the point $(S, I) = (1, 1)$, we

Table 6.1: Values of $\frac{dS}{dt}$ and $\frac{dI}{dt}$ for the flu with quarantine model.

S	I	dS_dt	dI_dt
0.0000000	0.0000000	0.0000000	0.0000000
0.4444444	0.4444444	-0.1975309	-0.2469136
0.8888889	0.8888889	-0.7901235	-0.0987654
1.3333333	1.3333333	-1.7777778	0.4444444
1.7777778	1.7777778	-3.1604938	1.3827160
2.2222222	2.2222222	-4.9382716	2.7160494
2.6666667	2.6666667	-7.1111111	4.4444444
3.1111111	3.1111111	-9.6790123	6.5679012
3.5555556	3.5555556	-12.6419753	9.0864198
4.0000000	4.0000000	-16.0000000	12.0000000

have an arrow that points directly to the west because and $\frac{dI}{dt} < 0$ and $\frac{dI}{dt} = 0$. Continuing on in this manner, by sequentially sampling points in the (S, I) plane we get a vector field plot (Figure 6.2), superimposed with the nullclines.

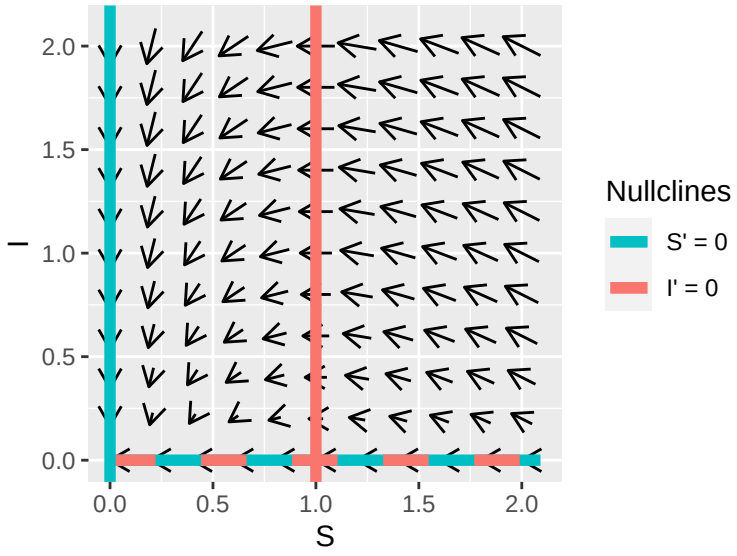


Figure 6.2: Phaseplane for Equation (6.2). To generate the plot we assumed $\beta = 1$ and $k = 1$

6.1.2 Motion around the nullclines

We can also extend the motion around the nullclines to investigate the stability of an equilbrium solution. With a one dimensional differential equation we used a number line to quantify values where the solution is increasing / decreasing. The problem with several differential equations is that the notion of "increasing" or "decreasing"" becomes difficult to understand - there is an additional degree of freedom! Simply put, in a plane you can move left/right *or* up/down. The benefit for having nullclines is that they **isolate** the motion in one direction. When $\frac{dS}{dt} = 0$ the only allowed motion is up and down; when $\frac{dI}{dt} = 0$ the only allowed motion is left and right.

In general for a two dimensional system: - When a horizontal axis variable has a nullcline, the only allowed motion is up/down. - When a vertical axis variable has a nullcline, the only motion is up/down.

Applying this knowledge to Equation (6.2), if we choose points where $I' = 0$ then we know that the only motion is to the left and the right because S can still change along that curve. If we choose points where $S' = 0$ then we know that the only motion is to the up/down because I can still change along that curve (Figure 6.3).

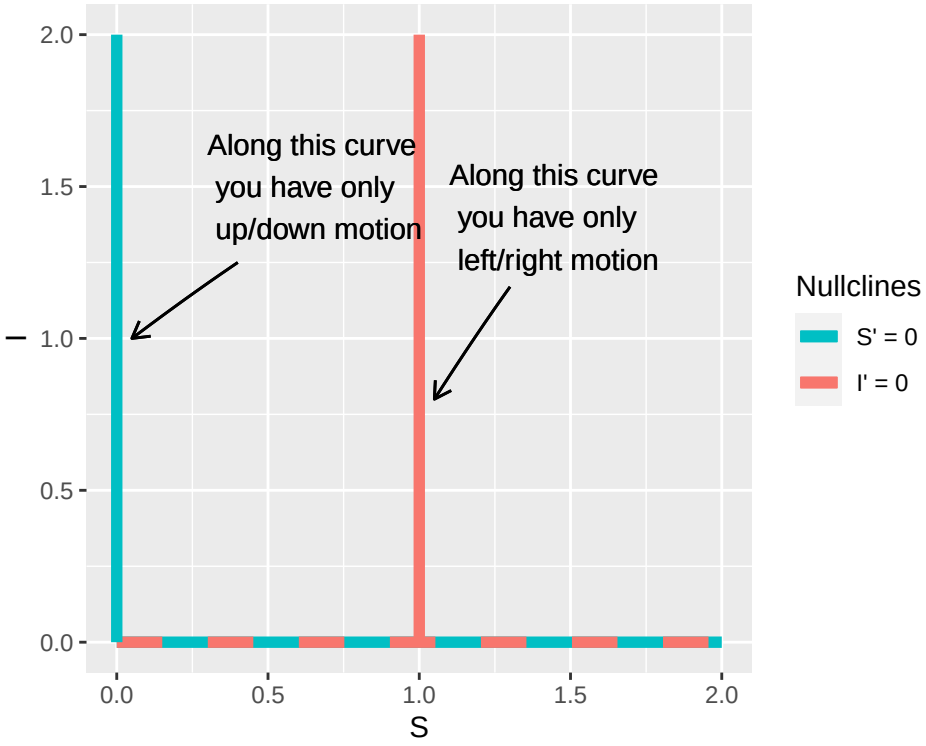


Figure 6.3: Nullclines for Equation (6.2) with context on the direction of the motion.

6.2 Determining stability of an equilbrium solution

The picture of the phase plane with the nullcline qualitatively tells us about the stability of an equilibrium point. Once of the equilibrium solutions is at the origin $(S, I) = (0, 0)$. As before we want to investigate if the equilibrium solution is stable or unstable.

Figure 6.4 zooms in the phaseplane at the equilibrium solution at $S = 0$, $I = 0$:

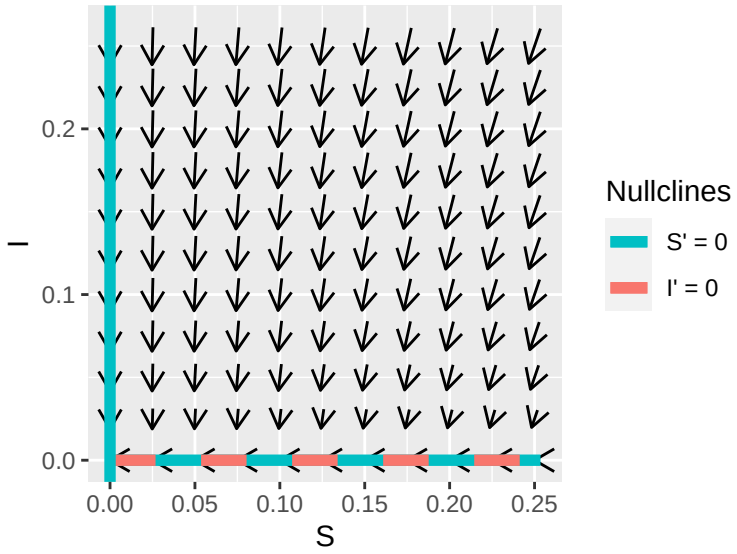


Figure 6.4: Zoomed in view of the phaseplane for Equation (6.2)

As you can see the arrows appear to be pointing into and towards the equilibrium solution. So we would classify this equilbrium solution as *stable*.

6.3 Generating a phase plane in R

Let's take what we learned from the case study of the flu model with quarantine to qualitatively analyze a system of differential equations:

- We determine nullclines by setting the derivatives equal to zero.
- Equilibrium solutions occur where nullclines for the two different equations intersect.
- The arrows in the phase plane help us characterize the stability of the equilibrium solution.

To determine the phaseplane diagram `demodelr` package has some basic functionality to generate a phase plane. Consider the following system of differential equations (Equation (6.5)):

$$\begin{aligned} \frac{dx}{dt} &= x - y \\ \frac{dy}{dt} &= -x + y \end{aligned} \tag{6.5}$$

In order to generate a phaseplane diagram for Equation (6.5) we need to define functions for x' and y', which I will annotate as dx and dy respectively. We are going to collect these equations in one vector called `system_eq`, using the tilde (~) as a replacement for the equals sign:

```
system_eq <- c(dx ~ x-y,
               dy ~ x+y)
```

Then what we do is apply the command `phaseplane`, which will generate a vector field over a domain:

```
phaseplane(system_eq,'x','y')
```

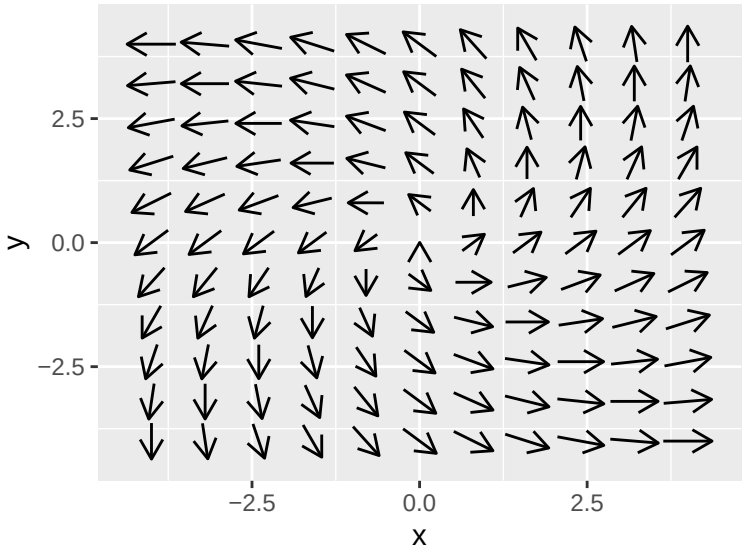


Figure 6.5: Phaseplane diagram for Equation (6.5)

```
# The values in quotes are the labels for the axes and to identify the variables - they are needed!
```

The command `phaseplane` has an option called `eq_soln` that will if there are any equilibrium solutions to be found and report them to the console. For example try running `phaseplane(system_eq,'x','y',eq_soln=TRUE)` and see what gets output to console. While this option lists equilibrium solutions, you should confirm them with the differential equation through direct solving.

6.3.1 Generating a phase line in R:

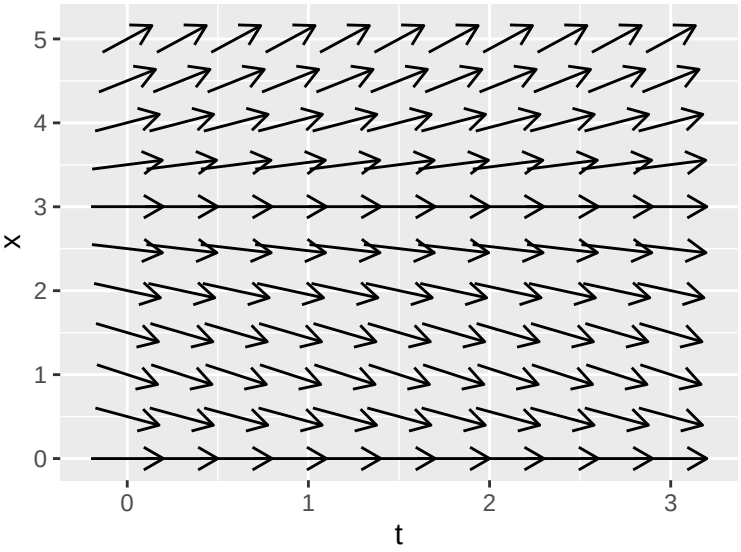
From Section 5 we discussed how to construct phaselines by hand. It turns out that the command `phaseplane` can also plot phase lines. Let's take a look at an example first and then discuss how that it works.

Example 6.1. A colony of bacteria growing in a nutrient-rich medium deplete the nutrient as they grow. As a result, the nutrient concentration $x(t)$ is steadily decreasing. Determine the phaseline for the following differential equation:

$$\frac{dx}{dt} = -0.7 \cdot \frac{x \cdot (3-x)}{1+x} \tag{6.6}$$

The R code to generate this phaseline is the following:

```
# Define the windows where we make the plots
t_window <- c(0,3)
x_window <- c(0,5)

# Define the differential equation
system_eq <- c(dt ~ 1,
               dx ~ -0.7 * x*(3-x)/(1+x))

phaseplane(system_eq,"t","x",t_window,x_window)
```



Notice how we have the equation $dt = 1$. What we are doing is re-writing the differential equation with a new variable s (Equation (6.7)):

$$\begin{aligned} \frac{dt}{ds} &= 1 \\ \frac{dx}{ds} &= -0.7 \cdot \frac{x \cdot (3-x)}{1+x} \end{aligned} \tag{6.7}$$

The differential equation $\frac{dt}{ds} = 1$ has solution $s = t$, so in essence is the same as Equation (6.6) (perhaps a little more complicated). However re-writing this system was a quick and handy workaround to re-use code.

6.4 Exercises

Exercise 6.1. Determine equilibrium solutions for Equation (6.5).

Exercise 6.2. This problem considers the following system of differential equations:

$$\begin{aligned} \frac{dx}{dt} &= y \\ \frac{dy}{dt} &= -x \end{aligned} \tag{6.8}$$

a. Determine the equations of the nullclines and equilibrium point of this system of differential equations.
b. Modify the function `phaseplane` to generate a phaseplane of this system.
c. For each point along a nullcline, determine the resulting motion (up-down or left-right).
d. Based on the work you generated, determine if the equilibrium solution is *stable* or *unstable*.
e. Verify that the functions $x(t) = \sin(t)$ and $y = \cos(t)$ is one solution to this system of differential equations.

Exercise 6.3. Consider the following system of differential equations:

$$\begin{aligned} \frac{dx}{dt} &= y \\ \frac{dy}{dt} &= 3x^2 - 1 \end{aligned} \tag{6.9}$$

a. Determine the equations of the nullclines and equilibrium solutions for this system of differential equations.
b. For each point along a nullcline, determine the resulting motion (up-down or left-right).
c. Modify the function `phaseplane` to generate a phaseplane of this system.
d. Make a hypothesis to classify if the equilibrium point is *stable* or *unstable*.

Exercise 6.4. (Inspired by Thornley and Johnson (1990)) A plant grows propritional to its current length L. Assume this proportionality constant is μ, whose rate also decreases proportional to its current value. The system of equations that models this plant growth is the following:

$$\begin{aligned} \frac{dL}{dt} &= \mu L \\ \frac{d\mu}{dt} &= -0.1\mu \end{aligned} \tag{6.10}$$

a. Explain why $L = 0$ and $\mu = 0$ is the only equilibrium solution to this differential equation.
b. Modify the function `phaseplane` to generate a phaseplane of this system.
c. Is the origin a stable equilibrium solution?

Exercise 6.5. (Inspired by Logan and Wolesensky (2009)) Red blood cells are formed from stem cells in the bone marrow. The red blood cell density r satisfies an equation of the form

$$\frac{dr}{dt} = \frac{0.2r}{1 + r^2} - 0.1r, \tag{6.11}$$

a. What are the equilibrium solutions for this differential equation?
b. Modify the function `phaseplane` to generate a phaseline for this differential euqation for $0 \leq t \leq 5$ and $0 \leq r \leq 5$.
c. Based on the phaseline, are the equilibrium solutions stable or unstable?

Exercise 6.6. (Inspired by Hugo van den Berg (2011)) Organisms that live in a saline environment biochemically maintain the amount of salt in their blood stream. An equation that represents the level of S in the blood is the following:

$$\frac{dS}{dt} = 1 + 0.3 \cdot (3 - S)$$

a. What are the equilibrium solutions for this differential equation?
b. Modify the function `phaseplane` to generate a phaseline for this differential euqation for $0 \leq t \leq 10$ and $0 \leq S \leq 10$.
c. Based on the phaseline, are the equilibrium solutions stable or unstable?

Exercise 6.7. (Inspired by Hugo van den Berg (2011)) The core body temperature (T) of a mammal is coupled to the heat production (scaled by heat capacity Q) with the following system of differential equations:

$$\begin{aligned} \frac{dT}{dt} &= Q + 0.5 \cdot (20 - T) \\ \frac{dQ}{dt} &= 0.1 \cdot (38 - T), \end{aligned} \tag{6.12}$$

a. Determine the equations of the nullclines and equilibrium point of this system of differential equations.
b. For each point along a nullcline, determine the resulting motion (up-down or left-right).
c. Make a hypothesis to classify if the equilibrium point is *stable* or *unstable*.

Exercise 6.8. Consider the following system of differential equations for the lynx-hare model:

$$\frac{dH}{dt} = rH - bHL \tag{6.13}$$
$$\frac{dL}{dt} = ebHL - dL \tag{6.14}$$

a. Determine the steady states of this system of differential equations.
b. Determine equations for the nullclines, expressed as L as a function of H. There should be two nullclnes for each rate.

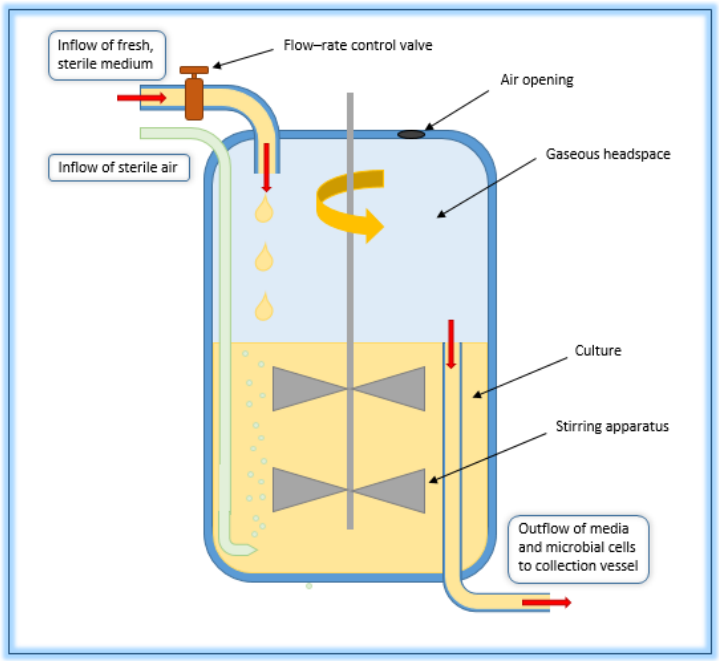


Figure 6.6: An example of a chemostat.

Exercise 6.9. (Inspired by Hugo van den Berg (2011)) A chemostat is a tank used to study microbes and ecology, where microbes grow under controlled conditions. Think of this like a large tank with nutrient-rich water being continuously cycled through, as shown in the following Figure 6.6. (Source: Wikipedia). Equations that describe the microbial biomass W and the nutrient concentration C (in the culture) are the following:

$$\begin{aligned} \frac{dW}{dt} &= \mu W - F\frac{W}{V} \\ \frac{dC}{dt} &= D \cdot (C_R - C) - S\mu\frac{W}{V}, \end{aligned} \tag{6.15}$$

where we have the following parameters: μ is the per capita reproduction rate, F is the flow rate, V is the volume of the culture solution, D is the dilution rate, C_R is the concentration of nutrients entering the culture, and S is a stoichiometric conversion of nutrients to biomass.

a. Write the equations of the nullclines for this differential equation.
b. Determine the equilibrium solutions for this system of differential equations.
c. Generate a phaseplane for this differential equation with the values $\mu = 1$, $D = 1$, $C_R = 2$ and $S = 1$ and $V = 1$.
d. Classify the stability of the equilbrium solutions.

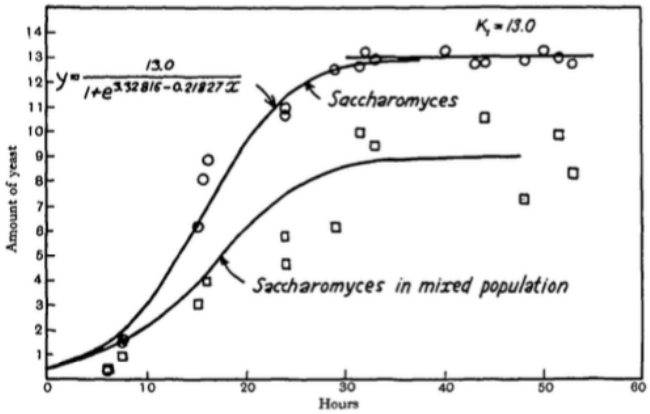
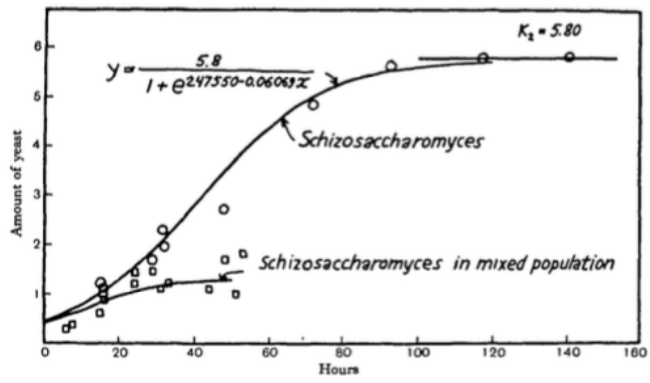


Fig. 2. The growth of the volume of *Saccharomyces cerevisiae* cultivated separately and in the mixed population according to the first and second series of experiments.



\begin{figure} \caption{Population results from two yeast species growing in competition. From Gause (1932).} \end{figure}

Exercise 6.10. A classical paper *Experimental Studies on the Struggle for Existence: I. Mixed Population of Two Species of Yeast* by Gause (1932) examined two different species of yeast growing in competition with each other. The differential equations given for two species in competition are:

$$\begin{aligned} \frac{dy_1}{dt} &= -b_1 y_1 \frac{(K_1 - (y_1 + \alpha y_2))}{K_1} \\ \frac{dy_2}{dt} &= -b_2 y_2 \frac{(K_2 - (y_2 + \beta y_1))}{K_2}, \end{aligned} \tag{6.16}$$

where y_1 and y_2 are the two species of yeast with the parameters $b_1,\ b_2,\ K_1,\ K_2,\ \alpha,\ \beta$ describing the characteristics of the yeast species.

a. Determine the equilibrium solutions for this differential equation. Express your answer in terms of the parameters $b_1,\ b_2,\ K_1,\ K_2,\ \alpha,\ \beta$.
b. Gause computed the following values of the parameters: $b_1 = 0.21827,\ b_2 = 0.06069,\ K_1 = 13.0,\ K_2 = 5.8,\ \alpha = 3.15,\ \beta = 0.439$. Using these values, what would be the predicted values for the equilibrium solutions?
c. Use the function `systems` to solve this system of differential equations numerically.
d. Figure 6.4 is from Gause (1932) and shows the experimental population values "in mixed population''. Use the graph to estimate the equilibrium solutions for both species. How close (or how far from) the equilibrium solutions are Gause's results to your computed equilibrium solutions?

Chapter 7

Exact Solutions to Differential Equations

We have already discussed some tools that can analyze differential equations numerically (Section 4) and qualitatively (Sections 5 and 6). The phase plane allowed us to evaluate equilibrium solutions and their stability. Beyond this graphical approach it is also helpful to know the exact solution to an equation. In this section we will study a few techniques to find exact solutions to differential equation. We will apply some of the tools you may have learned from calculus.

7.1 Separable Differential Equations

One technique to solve differential equations is the method of *separation of variables*. Let's look at an example:

What is the general solution to $\frac{dy}{dx} = yx^2$? To solve this expression we collect the variables involving x and one side of the equation, and the variables involving y on the other:

$$\frac{1}{y}dy = x^2 dx. \tag{7.1}$$

Now the next step is to determine the antiderivative of both sides of expression:

$$\begin{aligned} \int \frac{1}{y}dy &= \ln(y) + C. \\ \int x^2 dx &= \frac{1}{3}x^3 + C. \end{aligned} \tag{7.2}$$

Finally since both sides are equal we can solve for the dependent variable y. One thing to note: usually for antiderivatives we always include a $+C$. For solving differential equations it is okay just to keep only one $+C$, which usually is best on the side of the independent variable:

$$\ln(y) = \frac{1}{3}x^3 + C \rightarrow e^{\ln(y)} = e^{\frac{1}{3}x^3 + C} \rightarrow y = Ce^{\frac{1}{3}x^3}. \tag{7.3}$$

We are in business! So here is a general technique approach to solving a differential equation via separation of variables:

1. **Separate** the variables on one side of the equation.
2. **Integrate** both sides individually.
3. **Solve** for the dependent variable.

If we solve this equation using separation of variables we havUsing your work above as a guide, solve this differential equation to determine a solution $y(x)$.

7.2 Integrating factors

One model that we have looked at is the the SI model where the spread of the disease is proportional to the number infected:

$$\frac{dI}{dt} = .03(1000 - I) = 30 - .03I \tag{7.4}$$

While this differential equation can be solved via separation of variables, let's try a different approach as an illustration of another useful technique. First let's write the terms involving I on one side of the equation:

$$\frac{dI}{dt} + .03I = 30. \tag{7.5}$$

What we are going to do is multiply both sides of this equation by $e^{.03t}$ (I'll explain more about that later):

$$\frac{dI}{dt} \cdot e^{.03t} + .03I \cdot e^{.03t} = 30 \cdot e^{.03t} \tag{7.6}$$

Hmmm - this seems like we are making our equation harder to solve, doesn't it? However the left hand side is actually the derivative of the expression $I \cdot e^{kt}$! Let's take a look:

$$\frac{d}{dt}\left(I \cdot e^{.03t}\right) = \frac{dI}{dt} \cdot e^{.03t} + I \cdot .03e^{.03t} \tag{7.7}$$

This derivative is courtesy of the product rule from calculus. Ok, so what does this do to the differential equation? Well, by re-writing the differential equation as a derivative and integrating:

$$\begin{gathered} \frac{d}{dt}\left(I \cdot e^{.03t}\right) = 30 \cdot e^{.03t} \rightarrow \\ \int \frac{d}{dt}\left(I \cdot e^{.03t}\right)\, dt = \int 30 \cdot e^{.03t}\, dt \rightarrow \\ I \cdot e^{.03t} = 30 \cdot e^{.03t} + C \end{gathered} \tag{7.8}$$

Notice how by writing the left hand side in terms of the product rule and integrating we could find the solution. We added the $+C$ to the right hand side. All that is left to do is to solve in terms of $I(t)$ by dividing by e^{kt}. We will label this solution $I_1(t)$:

$$I_1(t) = 1000 + Ce^{-.03t} \tag{7.9}$$

Cool! The function $f(t) = e^{.03t}$ is called an *integrating factor*.

To see what is meant by that, let's try one more example.

Example 7.1. Modifying Equation (7.4), now let's assume that the rate of infection is time dependent, or $k(t) = .03t$. Apply the method of integrating factors to determine a solution to this differential equation.

Solution. How this would work in practice is that initially (at $t = 0$) there is no infection, but the infection rate increases as time goes on.

Our differential equation in this case is:

$$\frac{dI}{dt} + .03t \cdot I = 30t.$$

So if we want to write the left hand side as a product, what we will do is multiply the *entire* differential equation by $e^{\int .03t\, dt} = e^{0.015t^2}$ This term is called the *integrating factor*:

$$\frac{dI}{dt} \cdot e^{0.015t^2} + .03t \cdot I \cdot e^{0.015t^2} = 30t \cdot e^{0.015t^2} \tag{7.10}$$

First we rewrite the left hand side using the product rule:

$$\frac{dI}{dt} \cdot e^{0.015t^2} + .03t \cdot I \cdot e^{0.015t^2} = \frac{d}{dt}\left(I \cdot e^{0.015t^2}\right). \tag{7.11}$$

Now we can integrate this equation with the product rule in reverse:

$$\begin{aligned} \frac{d}{dt}\left(I \cdot e^{0.015t^2}\right) &= 30t \cdot e^{0.015t^2} \rightarrow \\ \int \frac{d}{dt}\left(I \cdot e^{0.015t^2}\right)\, dt &= \int 30t \cdot e^{0.015t^2}\, dt \rightarrow \\ I \cdot e^{0.015t^2} &= 1000 \cdot e^{0.015t^2} + C \end{aligned} \tag{7.12}$$

All right! So the last step is to write the equation in terms of $I(t)$, which we will label $I_2(t)$:

$$I_2(t) = 1000 + Ce^{-0.015t^2} \tag{7.13}$$

Figure 7.1 compares solutions $I_1(t)$ and $I_2(t)$ when the initial condition (in both cases) is 10 (so $I_1(0) = I_2(0) = 10$).

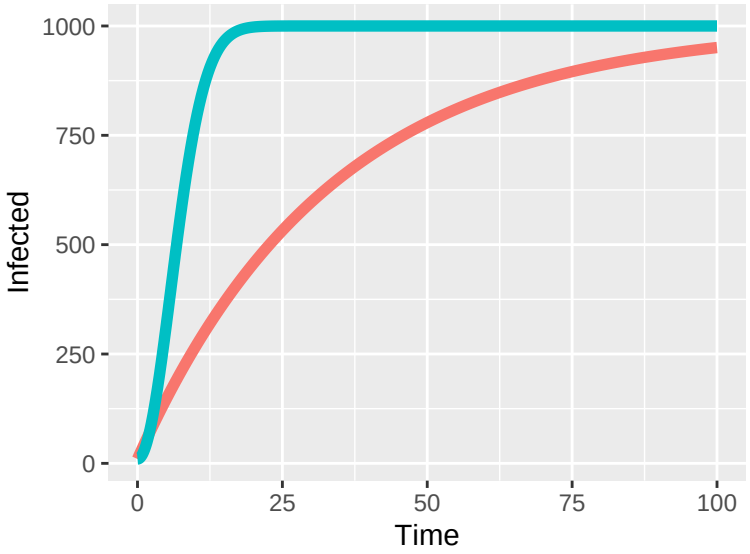


Figure 7.1: Comparison of two integrating factor solutions, Equation (7.9) in red and Equation (7.13) in blue.

Ok, let's summarize this integrating factor approach for differential equations that can be written in the form

$$\frac{dy}{dt} + f(t) \cdot y = g(t)$$

1. Calculate the *integrating factor* $e^{\int f(t)\, dt}$. Hopefully the integral $\int f(t)\, dt$ is easy to compute!
2. Next multiply the integrating factor across your equation to rewrite the differential equation as $\frac{d}{dt}\left(y \cdot e^{\int f(t)\, dt}\right) = g(t) \cdot e^{\int f(t)\, dt}$.
3. Then compute the integral $H(t) = \int g(t) \cdot e^{\int f(t)\, dt}\, dt$. This looks intimidating - but hopefully is manageable to compute! Don't forget the $+C$!
4. Then solve for $y(t)$: $y(t) = H(t) \cdot e^{-\int f(t)\, dt} + Ce^{-\int f(t)\, dt}$.

This technique is a handy way to work with equations that aren't easily separable.

7.3 Guess and Check

A final approach is called the guess and check method. Say for example we have the following equation that describes the rate of change above: The first approach if a function is a solution to a differential equation is the guess and check method, or by direct substitution.

$$\frac{dS}{dt} = 0.7S$$

We know that we can apply separation of variables, but instead let's try to see if the function $\tilde{S}(t) = 5e^{0.7t}$ is a solution in order to do that, we need to differentiate $\tilde{S}(t)$, which using our knowledge of calculus is $0.7 \cdot 5e^{0.7t}$. If we note that $\frac{d\tilde{S}}{dt} = 0.7\tilde{S} = 0.7e^{0.7t}$ than the function $\tilde{S}$ does solve the differential equation.

Super cool! Let's try an example:

Example 7.2. Verify the following functions are solutions to the differential equation $\frac{dS}{dt} = 0.7S$:

- $\tilde{R}(t) = 10e^{0.7t}$
- $\tilde{P}(t) = e^{0.7t}$
- $\tilde{Q}(t) = 5e^{0.7t}$
- $\tilde{F}(t) = 3$
- $\tilde{G}(t) = 0$

Solution. Let's apply direct differentiation to each of these functions:

- $\tilde{R}(t) = 10e^{0.7t} \to \tilde{R}'(t) = 7e^{0.7t}$
- $\tilde{P}(t) = e^{0.7t} \to \tilde{P}'(t) = e^{0.7t}$
- $\tilde{Q}(t) = 5e^{0.7t} \to \tilde{Q}'(t) = 3.5e^{0.7t}$
- $\tilde{F}(t) = 3 \to \tilde{F}'(t) = 0$
- $\tilde{G}(t) = 0 \to \tilde{G}'(t) = 0$

Now we will compare each of these solutions to the right hand side:

- $0.7\tilde{R}(t) = 0.7 \cdot 10e^{0.7t} \to 7e^{0.7t}$
- $0.7\tilde{P}(t) = 0.7e^{0.7t}$
- $0.7\tilde{Q}(t) = 0.7 \cdot 5e^{0.7t} \to = 3.5e^{0.7t}$
- $0.7\tilde{F}(t) = 0.7 \cdot 3 \to 2.1$
- $0.7\tilde{G}(t) = 0.7 \cdot 0 \to 0$

Notice how the right hand sides of each equation equals the left hand sides. When that is the case, our candidate functions are indeed solutions to the differential equation!

7.4 Superposition of solutions

Related to the Guess and Check method is this concept called superposition of solutions. Here how this works: if you have two known solutions to a differential equation, then the sum (or difference) is a solution as well. Let's look at an example:

Example 7.3. Show that $\tilde{S}(t) = 5e^{0.7t} + e^{0.7t}$ is a solution to the differential equation $\frac{dS}{dt} = 0.7S$

Solution. By direct differentiation, $\tilde{S}'(t) = 3.5e^{0.7t} + 0.7e^{0.7t}$. Also we have that $0.7 \cdot \tilde{S}(t) = 0.7 \cdot (5e^{0.7t} + e^{0.7t}) = 3.5e^{0.7t} + 0.7e^{0.7t}$, which equals $\tilde{S}$.

What this example illustrates is the principle that if you have two solutions to a differential equation, they can be added together and produce a new solution. This is an example of a *linear combinations* of solutions, and we can state this more formally:

> If $x(t)$ and $y(t)$ are solutions to the differential equation $z' = f(t, z)$, then $c(t) = a \cdot x(t) + b \cdot y(t)$ are also solutions, where a and b are constants.

Hopefully you were able to verify that $\tilde{R}$ and $\tilde{Q}$ and $\tilde{G}$ all were solutions to the differential equation, and that $\tilde{R} + \tilde{Q}$ was a solution as well. The most general solution to this differential equation is $S(t) = Ce^{0.7t}$, where the initial condition would determine the value of C.

7.5 Applying guess and check more broadly

As noted earlier, the guess and check method may seem to be trivial - if you have a differential equation, and solution, why verify it? Well, this method helps to introduce a useful solution technique to a differential equation, and one that we can build up through direct verification.

We are going to revisit the lynx hare model, but simplified a little bit. Here we are going to assume that lynx and hares both decline at a rate proportional to the population size, but the lynx population increases according to the rate of hare decline:

$$\begin{aligned} \frac{dH}{dt} &= -bH \\ \frac{dL}{dt} &= bH - dL \end{aligned} \tag{7.14}$$

Based on these simplified assumptions a good approach is to assume a solution that is exponential for both H and L:

$$\begin{aligned} \tilde{H}(t) &= C_1 e^{\lambda t} \\ \tilde{L}(t) &= C_2 e^{\lambda t} \end{aligned} \tag{7.15}$$

The form of this solution has three unknowns: C_1, C_2, and λ. If you have taken a course in Linear Algebra, you may recognize that we are assuming the solution is a vector of the form $\vec{v} = \vec{C}e^{\lambda t}$. Let's apply Guess and Check to solve these equations. By differentiation, we have the following:

$$\begin{aligned} \frac{d\tilde{H}}{dt} &= \lambda C_1 e^{\lambda t} \\ \frac{d\tilde{L}}{dt} &= \lambda C_2 e^{\lambda t}. \end{aligned} \tag{7.16}$$

Comparing to our differential equation we can show that

$$\begin{aligned} \lambda C_1 e^{\lambda t} &= -bC_1 e^{\lambda t} \rightarrow (\lambda + \delta) C_1 e^{\lambda t} = 0 \\ \lambda C_2 e^{\lambda t} &= bC_1 e^{\lambda t} - dC_2 e^{\lambda t} \end{aligned} \tag{7.17}$$

Let's rearrange this expression a little bit:

$$\begin{aligned} (\lambda + b) C_1 e^{\lambda t} &= 0 \\ (\lambda + d) C_2 e^{\lambda t} &= bC_1 e^{\lambda t} \end{aligned} \tag{7.18}$$

Notice that for the second equation we can solve for $C_1 e^{\lambda t}$, or $C_1 e^{\lambda t} = \dfrac{(\lambda + d)}{b} C_2 e^{\lambda t}$.

This allows for something neat to happen. We can substitute this expression for $C_1 e^{\lambda t}$ into the first equation:

$$(\lambda + b)\frac{(\lambda + d)}{b} C_2 e^{\lambda t} = 0 \tag{7.19}$$

If we assume that $b \neq 0$, then we have the following simplified expression:

$$(\lambda + b)(\lambda + d) C_2 e^{\lambda t} = 0 \tag{7.20}$$

Because the exponential function never equals zero, with this new equation, the only possibility is that $(\lambda + b)(\lambda + d) = 0$, or that $\lambda = -b$ or $\lambda = -d$. Remember: if expressions multiply to zero, then the only possibility is that at least one of them is

zero. The process outlined here finds the *eigenvalues* and *eigenvectors* of a system of equations. We will study these concepts in Section 18.

Next we need to determine values of C_1 and C_2. We can do this by going back to the equation $(\lambda + d)C_2e^{\lambda t} = bC_1e^{\lambda t}$, or $(\lambda + d)C_2e^{\lambda t} - bC_1e^{\lambda t} = 0$ rearranged.

Let's analyze this equation for each of the values of λ:

7.5.1 Case $\lambda = -d$

For this situation, we have

$$(-d + d)C_2e^{-dt} - bC_1e^{-dt} = 0 \rightarrow -bC_1e^{-dt} = 0.$$

The only way for this equation to be consistent and remain zero is if $C_1 = 0$. We don't have any restrictions on C_2, so the general solution will be

$$\tilde{H}(t) = 0 \tag{7.21}$$
$$\tilde{L}(t) = C_2e^{-dt}. \tag{7.22}$$

7.5.2 Case $\lambda = -d$

For this situation, we have $(-d + b)C_2e^{-dt} - dC_1e^{-dt} = 0$ which leads to the following equation:

$$\left((-d + b)C_2 - dC_1\right)e^{-dt} = 0 \tag{7.23}$$

The only way for this equation to be consistent and remain zero is if $((-d + b)C_2 - dC_1) = 0$, or if $C_2 = \left(\frac{d}{-d+b}\right)C_1$. In this case, the general solution will be

$$\tilde{H}(t) = C_1e^{-dt} \tag{7.24}$$
$$\tilde{L}(t) = \left(\frac{d}{-d+b}\right)C_1e^{-dt}, \tag{7.25}$$

The parameter C_2 can be determined by the initial condition. Notice that we need to have $d \neq b$ or our solution will be undefined.

Now we can write down a general solution to the system by combining our two solutions together. Here we can you the fact that two solutions can be added together (superposition) to generate a solution.

$$H(t) = C_1e^{-dt} \tag{7.26}$$
$$L(t) = \left(\frac{d}{-d+b}\right)C_1e^{-dt} + C_2e^{-bt} \tag{7.27}$$

This method only works on *linear* differential equations (i.e. it wouldn't work if there was a term such as kHL in our dynamics. Later on in the course we will look a more systematic method (i.e eigenvalues) to determine solutions to linear systems of equations.

7.6 Exercises

Exercise 7.1. Determine the value of C when $I(0) = 10$ for the two equations:

$$\begin{aligned} I_1(t) &= 1000 + Ce^{-.03t} \\ I_2(t) &= 1000 + Ce^{-0.015t^2} \end{aligned} \tag{7.28}$$

Exercise 7.2. Verify that $I_2(t) = N + Ce^{-0.5kt^2}$ is the solution to the differential equation $\dfrac{dI}{dt} = kt(N - I)$. Plot your dolution for various values of k ranging from .001 to .1. What effect does k have on the solution?

Exercise 7.3. A chemical reaction $2A \to C + D$ can be modeled with the following differential equation (Scholz and Scholz 2014):

$$\frac{dA}{dt} = -2kA^2 \tag{7.29}$$

Apply the method of separation of variables to determine a general solution for this differential equation.

Exercise 7.4. Which of the following differential equations be solved via separation of variables?

a. $\dfrac{dy}{dx} = x \cdot (y^2 + 2)$
b. $\dfrac{dy}{dx} = x^2 + xy$
c. $\dfrac{dy}{dx} = e^{x+y}$
d. $\dfrac{dy}{dx} = y \cdot \cos(2 + x)$
e. $\dfrac{dy}{dx} = \ln x + \ln y$

Once you have identified which ones can be solved via separation of variables, apply that technique to solve each differential equation.

Exercise 7.5. Solve the following differential equations by separation of variables:

a. $\dfrac{dy}{dx} = \dfrac{y^3}{x}$
b. $\dfrac{dy}{dx} = 1 + y^2$
c. $\dfrac{dy}{dx} = 8 - y$

Exercise 7.6. Consider the following differential equation $\dfrac{dP}{dt} = -\delta P$, $P(0) = P_0$, where δ is a constant parameter.

a. Solve this equation using the method of separation of variables.
b. Solve this euqation using an integrating factor.
c. Your two solutions from the two methods should be the same - are they?

Exercise 7.7. Here we return to the problem of how animals consume food. A differential equation that relates a consumer's nutrient content (denoted as y) to the nutrient content of food (denoted as x) is given by:

$$\frac{dy}{dx} = \frac{1}{\theta}\frac{y}{x}, \tag{7.30}$$

where $\theta \geq 1$ is a constant. Apply separation of variables to determine the general solution to this differential equation.

Exercise 7.8. Apply separation of variables to determine general solutions to the following systems of differential equations:

$$\begin{aligned} \frac{dx}{dt} &= x \\ \frac{dy}{dt} &= y \end{aligned} \tag{7.31}$$

(This system is an example of an *uncoupled* system of equations.)

Exercise 7.9. A plant grows propritional to its current length L. Assume this proportionality constant is μ, whose rate also decreases proportional to its current value. The system of equations that models this plant growth is the following:

$$\begin{aligned} \frac{dL}{dt} &= \mu L \\ \frac{d\mu}{dt} &= -k\mu \end{aligned} \tag{7.32}$$

(k is a constant parameter)

Apply separation of variables to determine the general solutions to this system of equations.

Exercise 7.10. Use the method developed in this section determine the general solution to the following system of differential equations:

$$\begin{aligned} \frac{dx}{dt} &= x - y \\ \frac{dy}{dt} &= 2y \end{aligned} \tag{7.33}$$

Exercise 7.11. Apply the method of integrating factors to determine the solution to the differential equation $\frac{dI}{dt} = (N - I) = kN - kI$, where k and N are parameters.

Exercise 7.12. For each of the following differential equations:

- Determine equilibrium solutions for the differential equation.
- Apply separation of variables to determine general solutions to the following differential equations:
- Choose reasonable values of any parameters and plot the solution curve for an initial condition that you select.

a. $\frac{dy}{dx} = -\frac{x}{y}$
b. $\frac{dy}{dx} = 8 - y$
c. $\frac{dW}{dt} = k(N - W)$ (k and N are constant parameters)
d. $\frac{dR}{dt} = -aR \ln \frac{R}{K}$ (a and K are constant parameters)

Exercise 7.13. Consider the following differential equation, where M represents a population of mayflies and t is time (given in months), and δ is a mortality rate (units % mayflies / month):

$$\frac{dM}{dt} = -\delta \cdot M \tag{7.34}$$

Determine the general solution to this differential equation and plot a few different solution curves with different values of δ. Assume that $M(0) = 10,000$. Also identify the equilibrium solution to the differential equation and classify the stability of the equilibrium solution based on your solution curves.

Exercise 7.14. An alternative model of mayfly mortality is the following:

$$\frac{dM}{dt} = -\delta(t) \cdot M, \tag{7.35}$$

where $\delta(t)$ is a time dependent mortality function. Determine a solution and plot a solution curve (assuming $M(0) = 10,000$ and over the interval from $0 \leq t \leq 1$) for this differential equation when $\delta(t)$ has the following forms:

a. $\delta(t) = t^2$
b. $\delta(t) = 1 - t^2$

Provide a reasonable biological explanation justifying the use of this alternative mayfly model.

Part II

Parameterizing Models with Data

Chapter 8

Linear Regression and Curve Fitting

8.1 What is parameter estimation?

Over the next several sections we will examine aspects of *parameter estimation*, which can be generally stated as the following process:

> Parameter estimation is the process of determining values of parameters $\vec{\alpha}$ for a function $f(\vec{x}, \vec{\alpha})$. Usually these parameters are determinined by minimizing the square difference between data $\vec{y}$ and the output of the function $f(\vec{x}, \vec{\alpha})$.

Example 8.1. The function $f(x) = ax + b$ has parameters a and b. In our notation above, $\vec{\alpha} = [a\ b]^T$. Usually these parameters can be determined through a set of measurements $(\vec{x}, \vec{y})$ (in other words a scatterplot).

Example 8.1 is an example of a *linear parameter estimation* problem. I did use matrix notation to denote the $\vec{\alpha}$ - although I will say matrix notation might be a little more formal for our purposes to start. To solve this problem we can address it from several different mathematical areas: *calculus* (optimization), *statistics* (likelihood functions), and *linear algebra* (systems of linear equations).

In this section I will show you how to apply `R` to determine the unknown parameters and interpret the results. A few section later we will explore *how* to approach the parameter estimation problem with likelihood and cost functions. We will use `R` a lot in this section to make plots - so please visit Section 2 if you need some reminders on plotting in R.

8.2 Fitting temperature data

Let's take a look at a specific example. Consider the following dataset of average global temperature over time (Table 8.1:

```
library(tidyverse)
library(demodelr)

kable(global_temperature[1:20, ], caption = "Global Temperature from 1880")
```

(This dataset can be found in the `demodelr` package with the name `global_temperature`.) To name our variables let $Y =$ Year since 1880 and $T =$ Temperature . First let's visualize the data, with time on the horizontal axis and temperature on the vertical axis:

```
ggplot(data = global_temperature, ) +
  geom_point(aes(x = yearSince1880, y = globalTemp),
    color = "red",
    size = 2
  ) +
  labs(x = "Y", y = "T")
```

Table 8.1: Global Temperature from 1880

yearSince1880	globalTemp
0	13.50
1	13.53
2	13.62
3	13.60
4	13.32
5	13.47
6	13.33
7	13.26
8	13.49
9	13.74
10	13.39
11	13.38
12	13.48
13	13.45
14	13.55
15	13.63
16	13.68
17	13.76
18	13.60
19	13.68

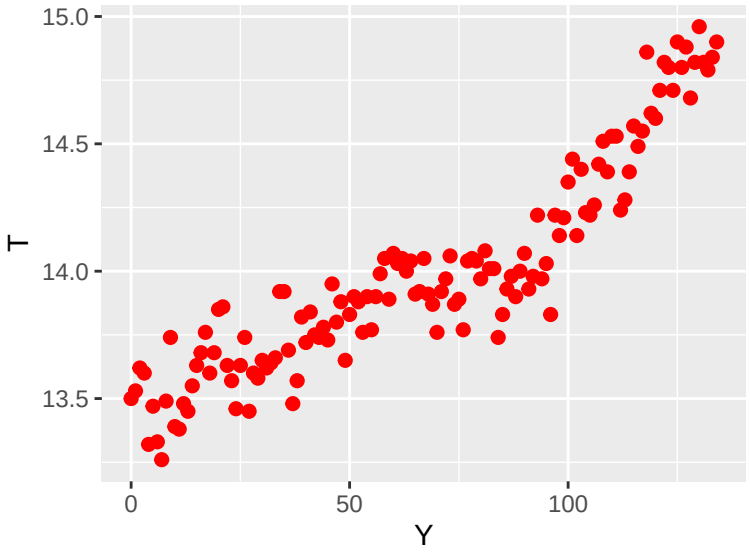


Figure 8.1: Scatterplot of global temperature data. The variable Y represents the Year since 1880 and T the temperature in degrees Celsius.

We will be working with these data to fit a function $f(Y, \vec{\alpha}) = T$. In order to fit a function in `R` we need three essential elements:

- We need **data** for the formula to interpret. This needs to be a two column data table, such as `global_temperature`.
- The **regression formula** we will use for the fit is given in the text `regressionFormula <- y ~ 1 + x`. We adopt the convention that y signifies the "dependent variable" and x signifies the "independent variable," but are *named columns in a data frame*. For the `global_temperature` dataset we would write the `regression_formula` as `regression_formula <- globalTemp ~ 1 + yearSince1880` Said differently, what this regression formula "does a linear regression where the factors are a constant term and one proportional to the independent variable."
- The **command** `lm` stands for *l*inear *m*odel. This is the main function that does our fitting procedure, where we need to specify the dataset we are using.

That's it! So if we need to do a linear regression of global temperature against year since 1880 the code to do this is the following:

```
regression_formula <- globalTemp ~ 1 + yearSince1880

linear_fit <- lm(regression_formula, data = global_temperature)

summary(linear_fit)
```

```
##
## Call:
## lm(formula = regression_formula, data = global_temperature)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.45013 -0.11632 -0.00849  0.11326  0.36865
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   13.358442   0.028832  463.32   <2e-16 ***
## yearSince1880  0.009601   0.000372   25.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1684 on 133 degrees of freedom
## Multiple R-squared:  0.8336, Adjusted R-squared:  0.8323
## F-statistic: 666.2 on 1 and 133 DF,  p-value: < 2.2e-16
```

What is printed on the console is the summary of the fit results. This summary contains several interesting things that you would study in advanced courses in statistics, but here is what we will focus on:

- The estimated **coefficients** (`Coefficients:`) of the linear regression. The column `Estimate` lists the constants in front of our regression formula $y = a + bx$. What follows is the error on that estimate by formulas from statistics. The other additional columns concern statistical tests that show significance of the estimate.
- One helpful thing to look at is the **Residual standard error** (`Residual standard error`), which represents the overall, total effect of the differences between the model predicted values of $\vec{y}$ and the measured values of $\vec{y}$. The goal of linear regression is to minimize this model-data difference.

To plot the fitted equation with the regression coefficients we are going to borrow from another package called `broom`, which helps produce model output in what is called "tidy" data format. You can read more about `broom` here.

Since we are only going to use one or two functions from this package, I am going to refer to the functions I need with the syntax `PACKAGE_NAME::FUNCTION`.

First we will make a data frame with the predicted coefficients from our linear model:

```
global_temperature_model <- broom::augment(linear_fit, data = global_temperature)

glimpse(global_temperature_model)
```

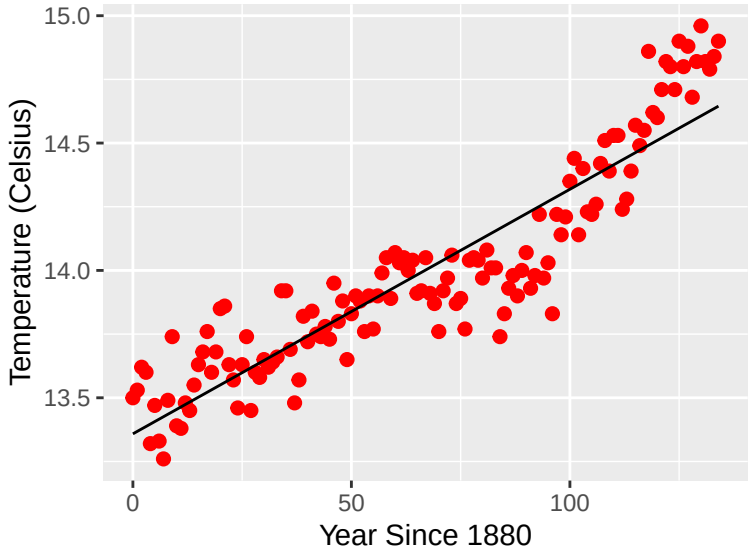
```
## Rows: 135
## Columns: 8
## $ yearSince1880 <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16~
## $ globalTemp    <dbl> 13.50, 13.53, 13.62, 13.60, 13.32, 13.47, 13.33, 13.26, ~
## $ .fitted       <dbl> 13.35844, 13.36804, 13.37764, 13.38725, 13.39685, 13.406~
## $ .resid        <dbl> 0.141557734, 0.161956817, 0.242355900, 0.212754983, -0.0~
## $ .hat          <dbl> 0.02930283, 0.02865412, 0.02801515, 0.02738595, 0.026766~
## $ .sigma        <dbl> 0.1686042, 0.1684612, 0.1677079, 0.1680214, 0.1689313, 0~
## $ .cooksd       <dbl> 1.098342e-02, 1.403997e-02, 3.069797e-02, 2.309589e-02, ~
## $ .std.resid    <dbl> 0.85304308, 0.97564433, 1.45949662, 1.28082181, -0.46247~
```

Notice how the augment command takes the results from `linear_fit` with the data `global_temperature`. I like appending `_model` to the original name of the data frame to signify that there are modeled components here to work with. There is a lot

to unpack with this new data frame, but the important ones are the columns `yearSince1880` (the independent variable) and `.fitted`, which represents the fitted coefficients.

Now we are ready to graph the data along with the fitted regression line.

```
ggplot(data = global_temperature) +
  geom_point(aes(x = yearSince1880, y = globalTemp),
    color = "red",
    size = 2
  ) +
  geom_line(
    data = global_temperature_model,
    aes(x = yearSince1880, y = .fitted)
  ) +
  labs(
    x = "Year Since 1880",
    y = "Temperature (Celsius)"
  )
```



8.3 Moving beyond linear models

We can also fit additional polynomial models such as the equation $y = a + bx + cx^2 + dx^3$... (estimated parameters a, b, c, d, ...). There is a key distinction here: the equation is *nonlinear* in the variable x, but *linear* with respect to the parameters. How we do that in `R` is pretty simple, it just depends on how we enter in the regression formula. Here are few templates:

Equation	Regression Formula
$y = a + bx$	`y ~ 1 + x`
$y = a$	`y ~ 1`
$y = bx$	`y ~ -1+x`
$y = a + bx + cx^2$	`y ~ 1 + x + I(x^2)`
$y = a + bx + cx^2 + dx^3$	`y~ 1 + x + I(x^2) + I(x^3)`

Note: the structure `I(..)` is needed for `R` to signify a factor of the form x^n.

8.4 Can you linearize your model?

We can also fit and plot nonlinear models in cases where the function can be transformed mathematically to a linear equation. Here is one example: while the equation $y = ae^{bx}$ non linear with respect to the parameters, it can be made linear by a

logarithmic transformation of the data:

$$\ln(y) = \ln(ae^{bx}) = \ln(a) + \ln(e^{bx}) = \ln(a) + bx \tag{8.1}$$

The advantage to this approach is that the growth rate parameter is easily identifiable from the data, and the value of a is found by exponentiation of the fitted intercept value. The disadvantage is that you need to interpret the do a log transform of the y variable first before doing any fits.

Example 8.2. A common equation in enzyme kinetics is the *Michaelis-Menten* law, which states that the rate of the uptake of a substrate V is given by the equation:

$$V = \frac{V_{max} s}{s + K_m}, \tag{8.2}$$

where s is the amount of substrate, K_m is half-saturation constant, and V_{max} the maximum reaction rate. (Typically V is used to signify the "velocity" of the reaction.)

Say you have the following data:

s (mM)	V (mM / s)
0.1	0.04
0.2	0.08
0.5	0.17
1.0	0.24
2.0	0.32
3.5	0.39
5.0	0.42

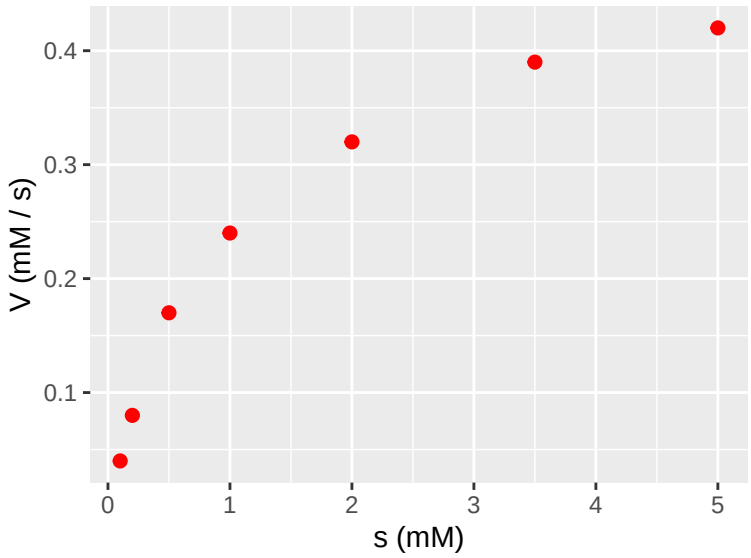
Let's explore these data in `R`.

Solution. First thing that we will need to do is to define a data frame (`tibble`) of these data:

```
enzyme_data <- tibble(
  s = c(0.1, 0.2, 0.5, 1.0, 2.0, 3.5, 5.0),
  V = c(0.04, 0.08, 0.17, 0.24, 0.32, 0.39, 0.42)
)
```

Next let's do some exploratory data analysis:

```
ggplot(data = enzyme_data) +
  geom_point(aes(x = s, y = V),
    color = "red",
    size = 2
  ) +
  labs(
    x = "s (mM)",
    y = "V (mM / s)"
  )
```



Definitely looks non-linear. But take a look at what happens if we plot the reciprocal of s and the reciprocal of V:

```
ggplot(data = enzyme_data) +
  geom_point(aes(x = 1 / s, y = 1 / V),
    color = "red",
    size = 2
  ) +
  labs(
    x = "1/s (1/mM)",
    y = "1/V (s / mM)"
  )
```

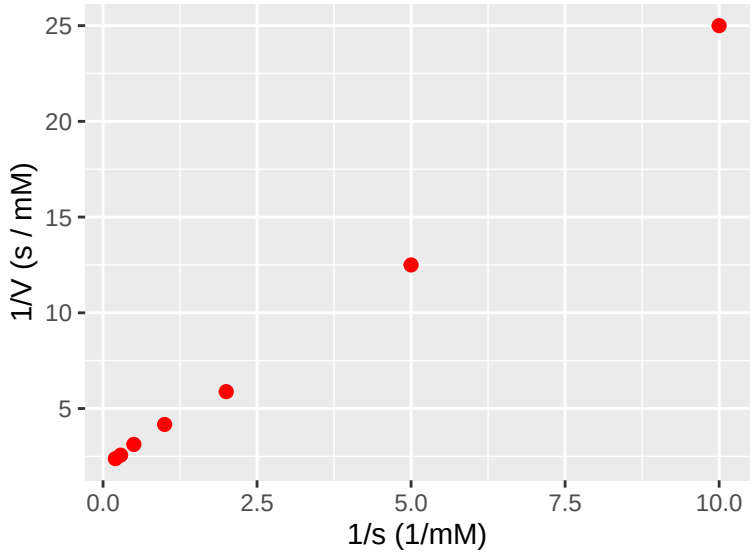


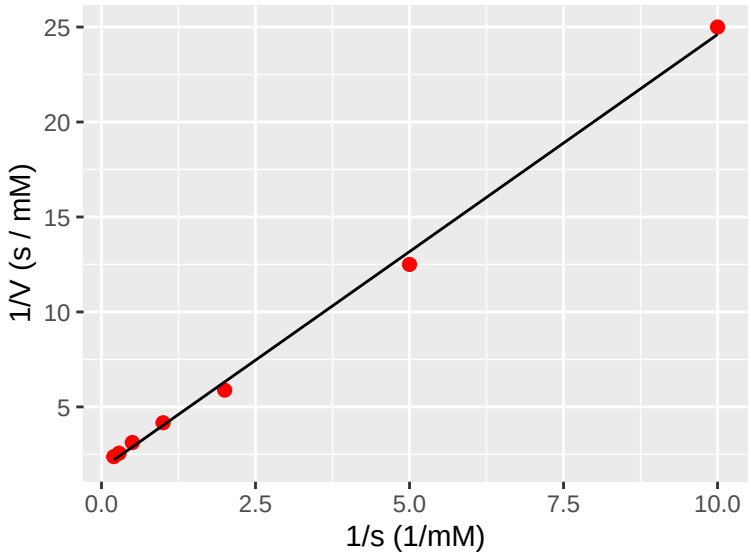
Figure 8.2: A plot of the transformed enzyme data.

Figure 8.2 really looks linear! Notice how easy it was to do that data transformation inside the `ggplot` command. In order to do a linear fit to the transformed data we will use the regression formulas defined above and the handy structure `I(VARIABLE)`:

```
enzyme_fit <- lm(I(1 / V) ~ 1 + I(1 / s),
  data = enzyme_data
)


summary(enzyme_fit)
```

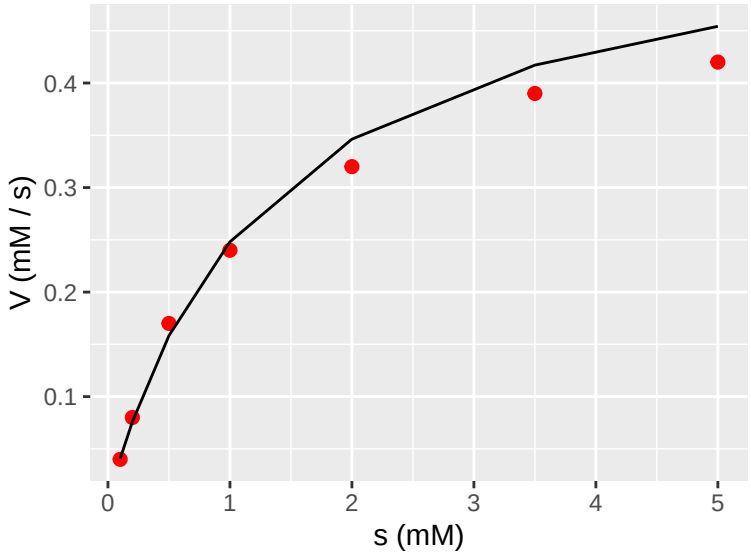
```
##
## Call:
## lm(formula = I(1/V) ~ 1 + I(1/s), data = enzyme_data)
##
## Residuals:
##       1       2       3       4       5       6       7
##  0.3913 -0.6764 -0.4347  0.1360  0.2376  0.1667  0.1795
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.74417    0.21009   8.302 0.000414 ***
## I(1/s)       2.28645    0.04868  46.968 8.26e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4323 on 5 degrees of freedom
## Multiple R-squared:  0.9977, Adjusted R-squared:  0.9973
## F-statistic:  2206 on 1 and 5 DF,  p-value: 8.263e-08
```

```
enzyme_data_model <- broom::augment(enzyme_fit, data = enzyme_data)

ggplot(data = enzyme_data) +
  geom_point(aes(x = 1 / s, y = 1 / V),
    color = "red",
    size = 2
  ) +
  geom_line(
    data = enzyme_data_model,
    aes(x = 1 / s, y = .fitted)
  ) +
  labs(
    x = "1/s (1/mM)",
    y = "1/V (s / mM)"
  )
```



Notice when plotting the fitted model we didn't need to take the reciprocal of `.fitted` because the linear model already did the inverse. However if we wanted to plot the model with the original data, then we need to take the reciprocal (confusing - I know!)

```
ggplot(data = enzyme_data) +
  geom_point(aes(x = s, y = V),
    color = "red",
    size = 2
  ) +
  geom_line(
    data = enzyme_data_model,
    aes(x = s, y = 1 / .fitted)
  ) +
  labs(
    x = "s (mM)",
    y = "V (mM / s)"
  )
```



8.5 Nonlinear models

Many cases you will not be able to write your model in a linear format. You can still do a non-linear curve fit using the function `nlm`, however you will need to specify the function along with the formula you are using. For example if we try to fit the weight of the dog Wilson over time to the logistic equation we would have the following:

$$W = f(D, a, b, c) = a - be^{ct}, \tag{8.3}$$

where we have the parameters a, b, and c. Notice how W is a function of D and the parameters.

```
nonlinear_fit <- nls(mass ~ a - b * exp(c * days),
  data = wilson,
  start = list(a = 75, b = 30, c = -0.01)
)


summary(nonlinear_fit)
```

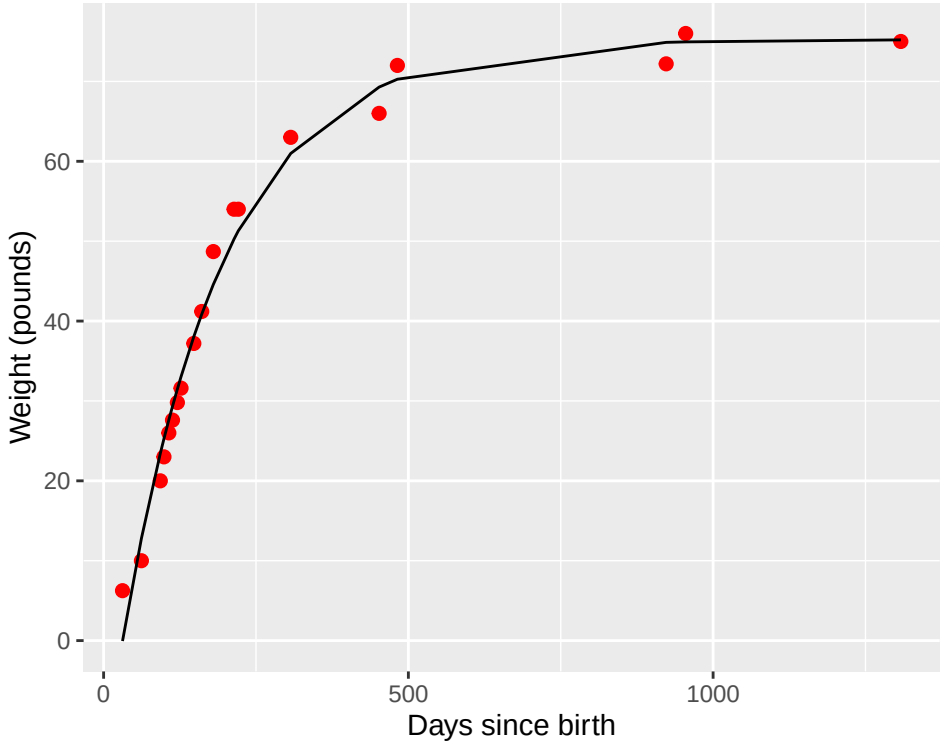
```
##
## Formula: mass ~ a - b * exp(c * days)
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
## a 75.2349873  1.5726835   47.84  < 2e-16 ***
## b 90.7696994  3.3999731   26.70 1.07e-14 ***
## c -0.0060311  0.0004324  -13.95 2.26e-10 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.961 on 16 degrees of freedom
##
## Number of iterations to convergence: 11
## Achieved convergence tolerance: 6.9e-06
```

The tricky part for a nonlinear model is that you need a starting value for the parameters (it is an iterative method). This can be tricky and takes some trial and error.

However once you have your fitted model, you can still plot the fitted values with the coefficients:

```
wilson_model <- broom::augment(nonlinear_fit, data = wilson)

ggplot(data = wilson) +
  geom_point(aes(x = days, y = mass),
    color = "red",
    size = 2
  ) +
  geom_line(
    data = wilson_model,
    aes(x = days, y = .fitted)
  ) +
  labs(
    x = "Days since birth",
    y = "Weight (pounds)"
  )
```



We will revisit these data later when we are making likelihood functions.

8.6 Exercises

Exercise 8.1. Determine if the following equations are linear with respect to the parameters. For the purposes of this problem we assume that y is a function of x.

a. $y = a + bx + cx^2 + dx^3$
b. $y = a\sin(x) + b\cos(x)$
c. $y = a\sin(bx) + c\cos(dx)$
d. $y = a + bx + a \cdot bx^2$
e. $y = ae^{-x} + be^{x}$
f. $y = ae^{-bx} + ce^{-dx}$

Exercise 8.2. Each of the following equations can be written as linear with respect to the parameters, through applying some elementary transformations to the data. Write each equation as a linear function with respect to the parameters.

a. $y = ae^{-bx}$
b. $y = (a + bx)^2$
c. $y = \dfrac{1}{a + bx}$
d. $y = cx^n$

Exercise 8.3. Use the dataset `global_temperature` and the function `lm` to answer the following questions:

a. Complete the following table, which represents various regression fits to global temperature T (in degrees Celsius) and years since 1880 (denoted by Y). In the table **Coeffiicients** represent the values of the parameters a, b, c, etc from your fitted equation; **P** = number of parameters; **RSE** = Residual standard error.

Equation	**Coefficients**	**P**	**RSE**
$T = a + bY$			
$T = a + bY + cY^2$			
$T = a + bY + cY^2 + dY^3$			
$T = a + bY + cY^2 + dY^3 + eY^4$			
$T = a + bY + cY^2 + dY^3 + eY^4 + fY^5$			
$T = a + bY + cY^2 + dY^3 + eY^4 + fY^5 + gY^6$			

b. After making this table, choose the polynomial of the function that you believe fits the data best. Provide reasoning and explanation why you chose the polynomial that you did.
c. Finally show the plot of your selected polynomial with the data.

Exercise 8.4. An equation that relates a consumer's nutrient content (denoted as y) to the nutrient content of food (denoted as x) is given by: $y = cx^{1/\theta}$, where $\theta \geq 1$ and c are both constants is a constant.

a. Show that you can write this equation as linear equation by applying a logarithm to both sides and simplifying.
b. Use the dataset `phosphorous` to determine a linear regression fit for your new linear equation.
c. Determine the value of c and θ in the original equation with the parameters from the linear fit.

Exercise 8.5. Following on from the last exercise, do a non-linear least squares fit for the dataset `phosphorous` to the equation $y = cx^{1/\theta}$, where $\theta \geq 1$ and c are both constants is a constant. For a starting point, you may use the values of c and θ from the previous exercise. Finally make a plot of the original phosophorous data and the fitted model.

Exercise 8.6. A common equation in enzyme kinetics is the *Michaelis-Menten* law, which states that the rate of the uptake of a substrate V is given by the equation:

$$V = \frac{V_{max} s}{s + K_m}, \tag{8.4}$$

where s is the amount of substrate, K_m is half-saturation constant, and V_{max} the maximum reaction rate. (Typically V is used to signify the "velocity" of the reaction.)

Say you have the following data:

s (mM)	V (mM / s)
0.1	0.04
0.2	0.08
0.5	0.17
1.0	0.24
2.0	0.32
3.5	0.39
5.0	0.42

a. Using algebra, show that this equation can be written as $\dfrac{1}{V} = \dfrac{1}{V_{max}} + \dfrac{K_m}{V_{max}} \cdot \dfrac{1}{s}$
b. The text determined the fitted coefficients for these transformed data. Determine values of K_m and V_{max}.
c. Make a plot of the actual data to the fitted model curve you found.

Note: The process outlined here is a *Lineweaver-Burk* plot.

Exercise 8.7. Following on from the last exercise, let's do a nonlinear least squares fit of the enzyme data to the equation:

$$V = \frac{V_{max} s}{s + K_m}, \tag{8.5}$$

where s is the amount of substrate, K_m is half-saturation constant, and V_{max} the maximum reaction rate.

a. Determine a non-linear least squares fit to the data for the given equation. You may use the values of K_m and V_{max} that you determined in the last exercise.
b. Make a plot of the actual data to the fitted model curve you found.

Exercise 8.8. Consider the following data which represents the temperature over the course of a day:

Hour	0	1	2	3	4	5	6	7	8	9	10	11	12
Temperature (°F)	54	53	55	54	58	58	61	63	67	66	67	69	68
Hour	13	14	15	16	17	18	19	20	21	22	23	24	
Temperature (°F)	68	66	67	63	60	59	57	56	53	52	54	53	

a. Make a scatterplot of these data, with the variable **Hour** on the horizontal axis.
b. A function that describes these data is $T = A + B \sin\left(\dfrac{\pi}{12} \cdot H\right) + C \cos\left(\dfrac{\pi}{12} \cdot H\right)$, where H is the hour and T is the temperature. Explain why this equation is linear for the parameters A, B, and C.
c. Define a `tibble` that include the variables T, $\sin\left(\dfrac{\pi}{12} \cdot H\right)$ $\cos\left(\dfrac{\pi}{12} \cdot H\right)$.
d. Do a linear fit on your new data frame to report the values of A, B, and C.
e. Add your fitted curve to the scatterplot.

Chapter 9

Probability and Likelihood Functions

The problem we examined in the last sectionr was the following:

> Determine the set of parameters $\vec{\alpha}$ that minimize the difference between data $\vec{y}$ and the output of the function $f(\vec{x}, \vec{\alpha})$ and measured error $\vec{\sigma}$.

We are going to examine the linear regression problem again using a smaller dataset by applying *likelihood functions*, which is a topic from probability and statistics.

This section will introduce likelihood functions but also discuss some interesting visualization techniques of multivariable functions and contour plots. We will also see a technique to evaluate a continuous function. We are starting to build out some `R` skills and techniques that you can apply in other context. Let's get started!

9.1 Linear regression, part 2

Assume we have the following (limited) number of points where we wish to fit a function of the form $y = bx$.

x	y
1	3
2	5
4	4
4	10

For this example we are forcing the intercept term a to equal zero - for most cases you will just fit the linear equation (see Exercise 9.7 where you will consider the intercept a). Figure 9.1 displays a quick scatterplot of these data:

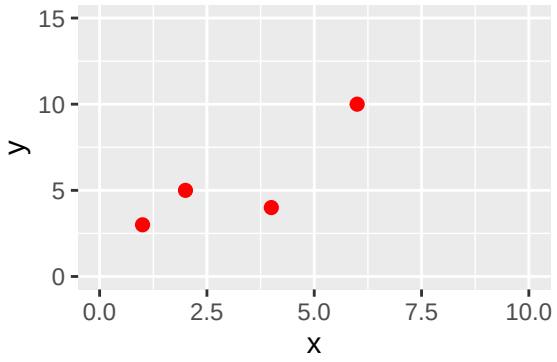


Figure 9.1: A scatterplot of a small dataset.

The goal here is to work to determine the value of b that is most *likely* (in other words, consistent) with the data. However, before we tackle this further we need to understand how to quantify *more likely* in a mathematical sense. In order to do this,

we need to take a quick excursion into probability distributions. Let's go!

9.2 Probability

In order to understand likelihood functions, first I am going to review very essential information about probability and probability distributions. Probability is the association of a set of observable events to a quantitative scale between 0 to 1. (Zero means that event is not possible, 1 means that it definitely can happen). This definition could be refined somewhat (Devore, Berk, and Carlton 2021). Events can be considered as discrete events (think counting or combinatorial problems) or continuous events. For our purposes we are only going to consider continuous events, specifically in this case the probability of a parameter obtaining a particular value.

Consider this graphic, which may be familiar to you as the normal distribution or the bell curve:

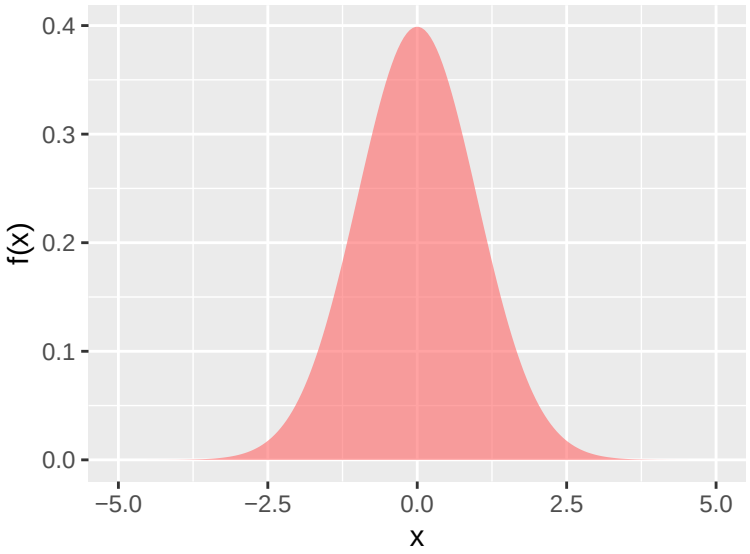


Figure 9.2: The normal distribution

We tend to think of the plot and the associated function $f(x)$ as something with input and output (such as $f(0) = 0.3989$). However because it is a probability density function, the *area* between two points gives yields the probability of an event to fall within two values:

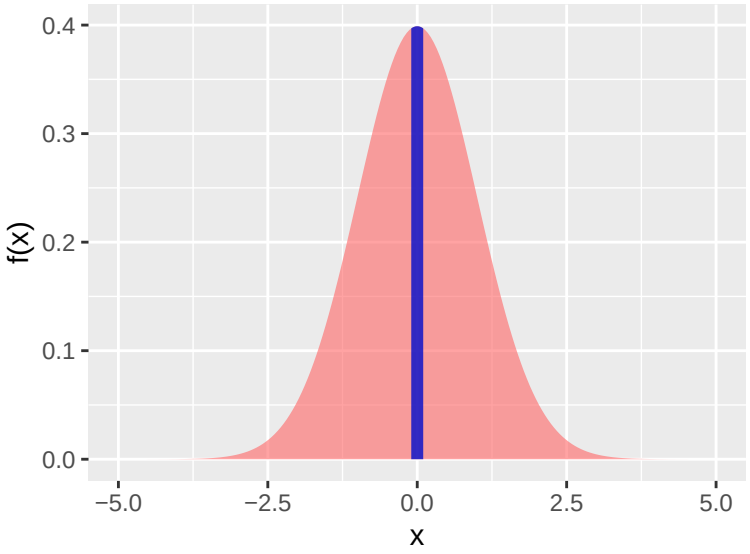


Figure 9.3: The area between two values, normally distributed

In this case, the shaded area tells us the probability that our measurement is between $x = -0.1$ and $x = 0.1$. The value of the

area, or the probability is 0.07966. When you took calculus the area was expressed as a definite integral: $\int_{-0.1}^{0.1} f(x)\,dx = 0.07966$, where $f(x)$ is the formula for the probability density function for the normal distribution.

The basic idea is that we can assign values to an outcomes as a way of displaying our belief (confidence) in the result. With this intuition we can summarize key facts about probability density functions:

- $f(x) \geq 0$ (this means that probability density functions are positive values)
- Area integrates to one (in probability, this means we have accounted for all of our outcomes)

Probability density functions also have formulas. For example, the formula for the normal distribution is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/(2\sigma^2)} \tag{9.1}$$

Where μ is the mean and σ is the standard deviation

9.2.1 Other probability distributions

Beyond the normal distribution some of the more common ones we utilize in the parameter estimation are the following:

- **Uniform:** For this distribution we must specify between a minimum value a and maximum value b.

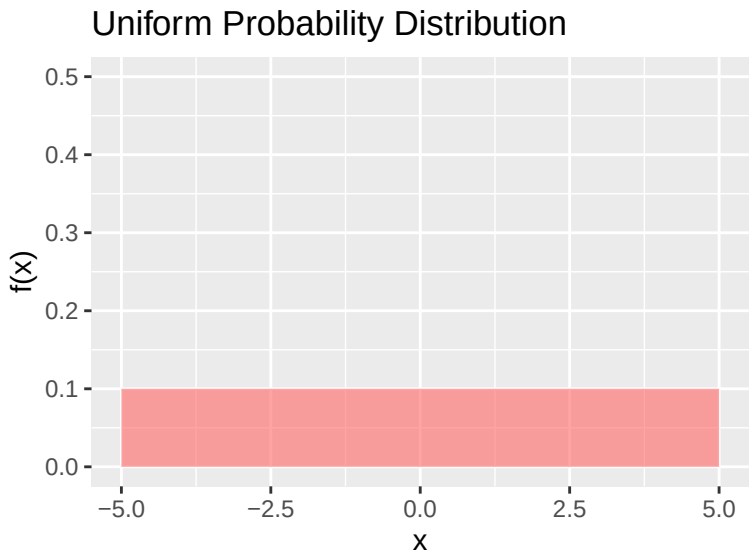


Figure 9.4: The uniform distribution

The formula for the uniform distribution is

$$f(x) = \frac{1}{b-a} \text{ for } a \leq x \leq b \tag{9.2}$$

- **Exponential:** For this distribution we must specify between a rate parameter λ.

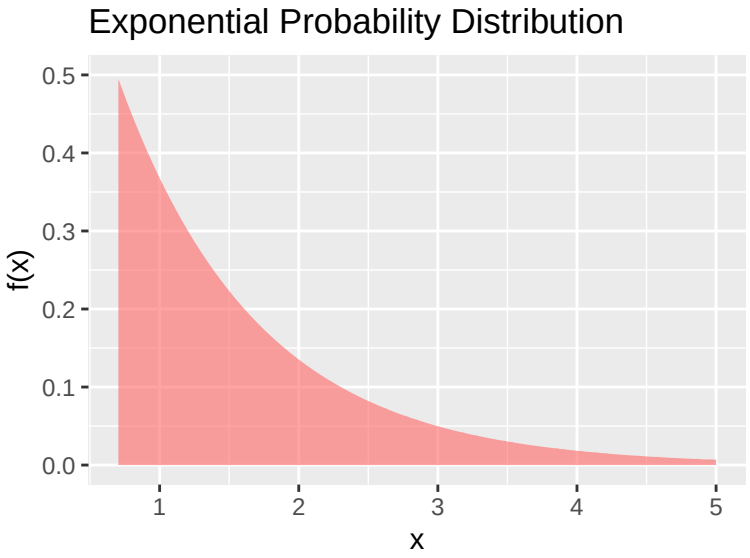


Figure 9.5: The exponential distribution

The formula for the exponential distribution is

$$f(x) = \lambda e^{-\lambda x} \text{ for } x \geq 0 \tag{9.3}$$

where λ is the rate parameter

- **Lognormal:** This distirbution is for positive values, with mean μ and standard deviation σ.

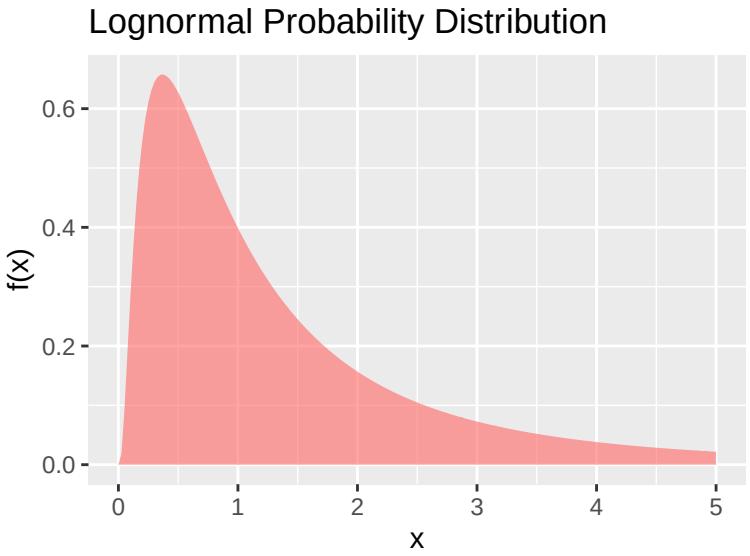


Figure 9.6: The lognormal distribution

The formula for the lognormal distribution is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma x} e^{-(\ln(x)-\mu)^2/(2\sigma^2)} \text{ for } x \geq 0 \tag{9.4}$$

9.2.2 Computing and graphic probabilities in R

Here is the good news with R: the commands to generate densities and cumulative distributions are already included! There are a variety of implementations: both for the density, cumulative distribution, random number generation, and lognormal distributions from these. For the moment, Table 9.2 summarizes some common probability distributions in R.

Distribution	Key Parameters	R command	Density Example
Normal	$\mu \to$ `mean`, $\sigma \to$ `sd`	`norm`	`dnorm(mu=0,sd=1,seq(-5,5,length=200))`
Uniform	$a \to$ `min`, $b \to$ `max`	`unif`	`dunif(seq(-5,5,length=200),min = -5,max=5)`
Exponential	$\lambda \to$ `rate`	`exp`	`dexp(seq(0,5,length=200))`
Normal	$\mu \to$ `meanlog`, $\sigma \to$ `sdlog`	`lnorm`	`dlnorm(seq(0,5,length=200))`

Table 9.2: Summary of common probability distributions in R

To make the graphs of these density functions in R we use the prefix `d` + the name (`norm`, `exp`) etc of the distribution we wish to specify, including any of the key parameters. If we don't include any of the parameters then it will just use the defaults (which you can see by typing `?NAME` where `NAME` is the name of the command (i.e. `?dnorm`).

Example 9.1. Make a graph of the lognormal density function with $\mu = 0$ and $\sigma = 1$ from $0 \leq x \leq 5$.

Solution. For this case we are using the defaults of the lognormal distribution. The code is the following, and plotted in Figure 9.7.

```
x <- seq(0, 5, length = 200)
y <- dlnorm(x) # Just use the mean defaults

# Define your data frame to plot
lognormal_data <- tibble(x, y)

ggplot() +
  geom_line(
    data = lognormal_data,
    aes(x = x, y = y)
  ) +
  labs(x = "x", y = "Lognormal density")
```

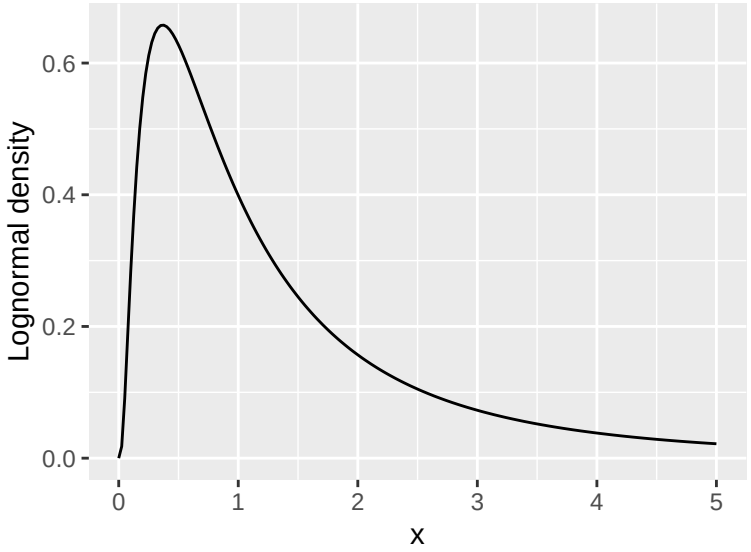


Figure 9.7: Code to plot the lognormal distribution

To determine the area between two values in a density function we use the prefix `p`.

Example 9.2. Use R to evaluate $\int_1^2 e^{-x}\, dx$.

Solution. The function e^{-x} is the exponential probability distribution with $\lambda = 1$. For this example if we wanted to find the area between two values in the exponential density in the shaded graph we would type `pexp(2)-pexp(1)` at the R console, which would give the value of 0.233. A visual representation of this area is shown in Figure 9.8.

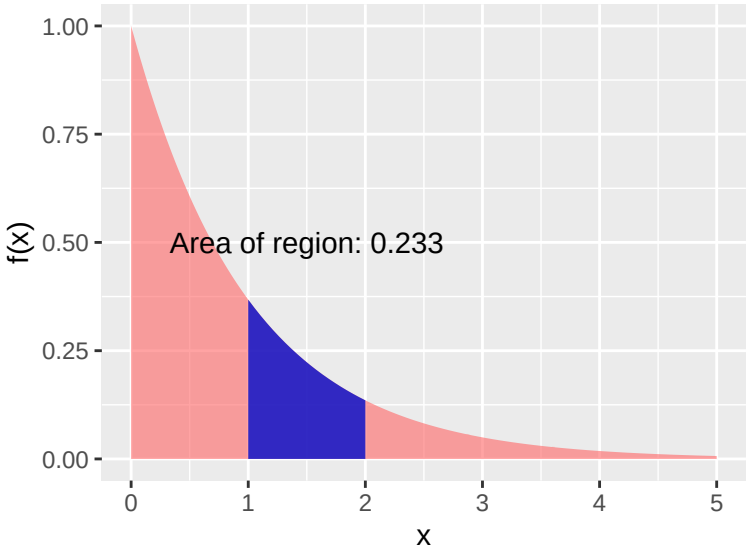


Figure 9.8: The area for the exponential distribution

9.3 Connecting probabilities to linear regression

Now that we have made that small excursion into probablity, let's start to return back to the linear regression problem. Another way to phrase this the linear regression problem studied in the last section is to examine the probability distribution of the model-data residual ϵ:

$$\epsilon_i = y_i - f(x_i, \vec{\alpha}). \tag{9.5}$$

The approach with likelihood functions assumes a particular probability distribution on each residual. One common assumption is that the residual *distribution* is normal with mean $\mu = 0$ and standard deviation σ (which could be specified as measurement error, etc).

$$L(\epsilon_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\epsilon_i^2/2\sigma^2}, \tag{9.6}$$

To extend this further across all measurement, we use the idea of *independent, identically distributed* measurements so the joint likelihood of **all** the residuals is the product of the likelihoods. The assumption of independent, identically distributed is is a common one. As a note of caution you should always evaluate if this is a valid assumption for more advanced applications.

$$L(\vec{\epsilon}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} e^{-\epsilon_i^2/2\sigma^2}, \tag{9.7}$$

We are making progress here, but in order to fully characterize the solution we need to specify the parameters $\vec{\alpha}$. A simple redefining of the likelihood function where we specify the measurements (x and y) and parameters ($\vec{\alpha}$) is all we need:

$$L(\vec{\alpha}|\vec{x}, \vec{y}) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi}\sigma} \exp(-(y_i - f(x_i, \vec{\alpha}))^2/2\sigma^2) \tag{9.8}$$

Now we have a function where the best parameter estimate is the one that optimizes the likelihood.

To return to our original linear regression problem (Figure 9.1, as a reminder we wanted to fit the function $y = bx$ to the following set of points:

x	y
1	3

x	y
2	5
4	4
4	10

The likelihood $L(\epsilon_i)$ $N(0, \sigma)$ characterizing these data are the following:

$$L(b) = \left(\frac{1}{\sqrt{2\pi}\sigma}\right)^4 e^{-\frac{(3-b)^2}{2\sigma}} \cdot e^{-\frac{(5-2b)^2}{2\sigma}} \cdot e^{-\frac{(4-4b)^2}{2\sigma}} \cdot e^{-\frac{(10-4b)^2}{2\sigma}} \tag{9.9}$$

For the purposes of our argument here, we will assume $\sigma = 1$. Figure 9.9 shows a plot of the likelihood function $L(b)$.

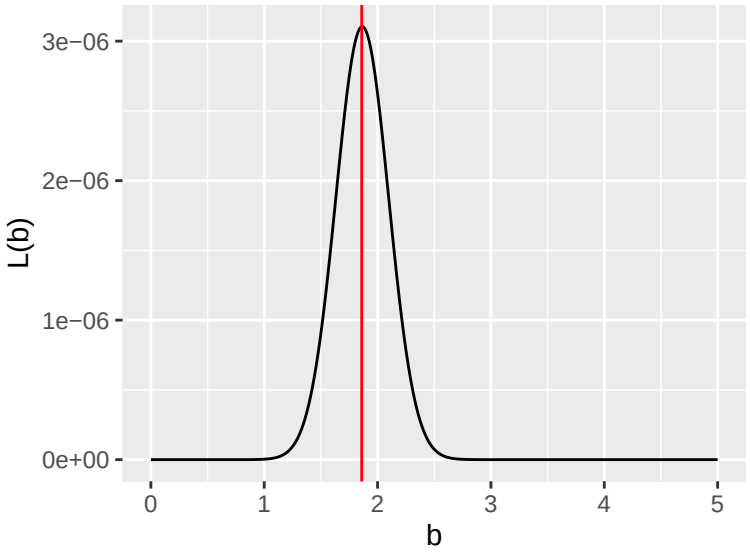


Figure 9.9: The likelihood function for the small dataset

Note that in Figure 9.9 the y values of $L(b)$ are really small (this may be the case), the likelihood function is maximized at $b = 1.86$. An alternative to the small numbers in $L(b)$ is to use the log likelihood (Equation (9.10)):

$$\begin{aligned} \ln(L(\vec{\alpha}|\vec{x}, \vec{y})) &= N \ln\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \sum_{i=1}^{N} \frac{(y_i - f(x_i, \vec{\alpha})^2}{2\sigma^2} \\ &= -\frac{N}{2}\ln(2) - \frac{N}{2}\ln(\pi) - N\ln(\sigma) - \sum_{i=1}^{N} \frac{(y_i - f(x_i, \vec{\alpha})^2}{2\sigma^2} \end{aligned} \tag{9.10}$$

In the homework you will be working on how to transform the likelihood function $L(b)$ to the log-likelihood $\ln(L(b))$.

9.4 Plotting likelihood surfaces

Ok, we are going to examine a second example from Gause (1932) which modeled the growing of yeast in solution. This classic paper examines the biological principal of *competitive exclusion*, how one species can out compete another one for resources. The data from Gause (1932) is encoded in the data frame `yeast` in the `demodelr` package. For this example we are going to examine a model for one species growing without competition. Figure 9.10 shows a scatterplot of the `yeast` data.

```
### Make a quick ggplot of the data

ggplot() +
  geom_point(
```

```
    data = yeast,
    aes(x = time, y = volume),
    color = "red",
    size = 2
  ) +
  labs(x = "Time", y = "Volume")
```

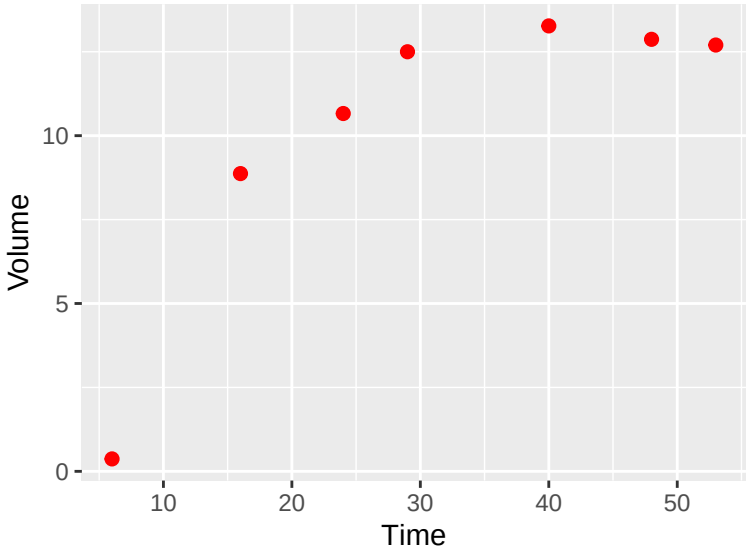


Figure 9.10: Scatterplot of *Sacchromyces* volume growing by itself in a container.

We are going to assume the population of yeast (represented with the measurement of volume) over time changes according to the equation:

$$\frac{dy}{dt} = -by\frac{(K-y)}{K}, \tag{9.11}$$

where y is the population of the yeast and b represents the growth rate and K is the carrying capacity of the population. It can be shown that the solution to this differential equation is $y = \dfrac{K}{1+e^{a-bt}}$, where the additional parameter a can be found through application of the initial condition y_0. In Gause (1932) the value of a was determined by solving the initial value problem $y(0) = 0.45$. In Exercise 9.1 you will show that $a = \ln\left(\dfrac{K}{0.45} - 1\right)$.

Equation (9.11) has two parameters: K and b. Here we are going to explore the likelihood function to try to determine the best set of values for the two parameters K and b using the function `compute_likelihood`. Inputs to the `compute_likelihood` function are the following:

- A function $y = f(x, \vec{\alpha})$
- A dataset (x, y)
- Ranges of your parameters $\vec{\alpha}$.

The `compute_likelihood` function also has an optional input `logLikely` that allows you to specify if you want to compute the likelihood or the log likelihood. The default is that `logLikely` is `FALSE`, meaning that the normal likelihoods are plotted.

First we will define the equation used to compute our model in the likelihood. As with the functions `euler` or `systems` we need to define this function:

```
library(demodelr)

# Gause model equation
gause_model <- volume ~ K / (1 + exp(log(k / 0.45 - 1) - b * time))
```

```
# Identify the ranges of the parameters that we wish to investigate
kParam <- seq(5, 20, length.out = 100)
bParam <- seq(0, 1, length.out = 100)


# Allow for all the possible combinations of parameters
gause_parameters <- expand.grid(K = kParam, b = bParam)

# Now compute the likelihood
gause_likelihood <- compute_likelihood(
  model = gause_model,
  data = yeast,
  parameters = gause_parameters,
  logLikely = FALSE
  )
```

Ok, let's break this code down step by step:

- The line `gause_model <- volume ~ K/(1+exp(log(k/0.45-1)-b*time))` identifies the formula that relates the variables `time` to `volume`.
- We define the ranges (minimum and maximum values) for our parameters by defining a sequence. Because we want to look at *all possible combinations* of these parameters we use the command `expand.grid`.
- The input `logLikely = FALSE` to `compute_likelihood` reports back likelihood values.

Some care is needed in defining the number of points that we want to evaluate - we will have 100^2 different combinations of K and b, which do take time to evaluate.

The output to `compute_likelihood` is a list - this is a flexible data structure. You can think of this as a collection of items. In this case, what gets returned are two data frames: `likelihood`, which is a data frame of likelihood values for each of the parameters and `opt_value`, which reports back the values of the parameters that optimize the likelihood function. Note that the optimum value is *an approximation*, as it is just the optimum from the input values. Let's take a look at the reported optimum values, which we can do with the syntax `LIST_NAME$VARIABLE_NAME`, where the dollar sign ($) helps identify which variable from the list you are investigating.

```
gause_likelihood$opt_value
```

```
## # A tibble: 1 x 4
##       K      b       l_hood log_lik
##   <dbl>  <dbl>        <dbl> <lgl>
## 1  12.7 0.0707 0.0000000492 FALSE
```

It is also important to visualize this likelihood function. For this dataset we have the two parameters K and b, so the likelihood function will be a likelihood surface. The code to generate the plot looks a little different from previous plots we have used previously:

```
# Define the likelihood values
my_likelihood <- gause_likelihood$likelihood

# Make a contour plot
ggplot(data = my_likelihood) +
  geom_tile(aes(x = K, y = b, fill = l_hood)) +
  stat_contour(aes(x = K, y = b, z = l_hood))
```

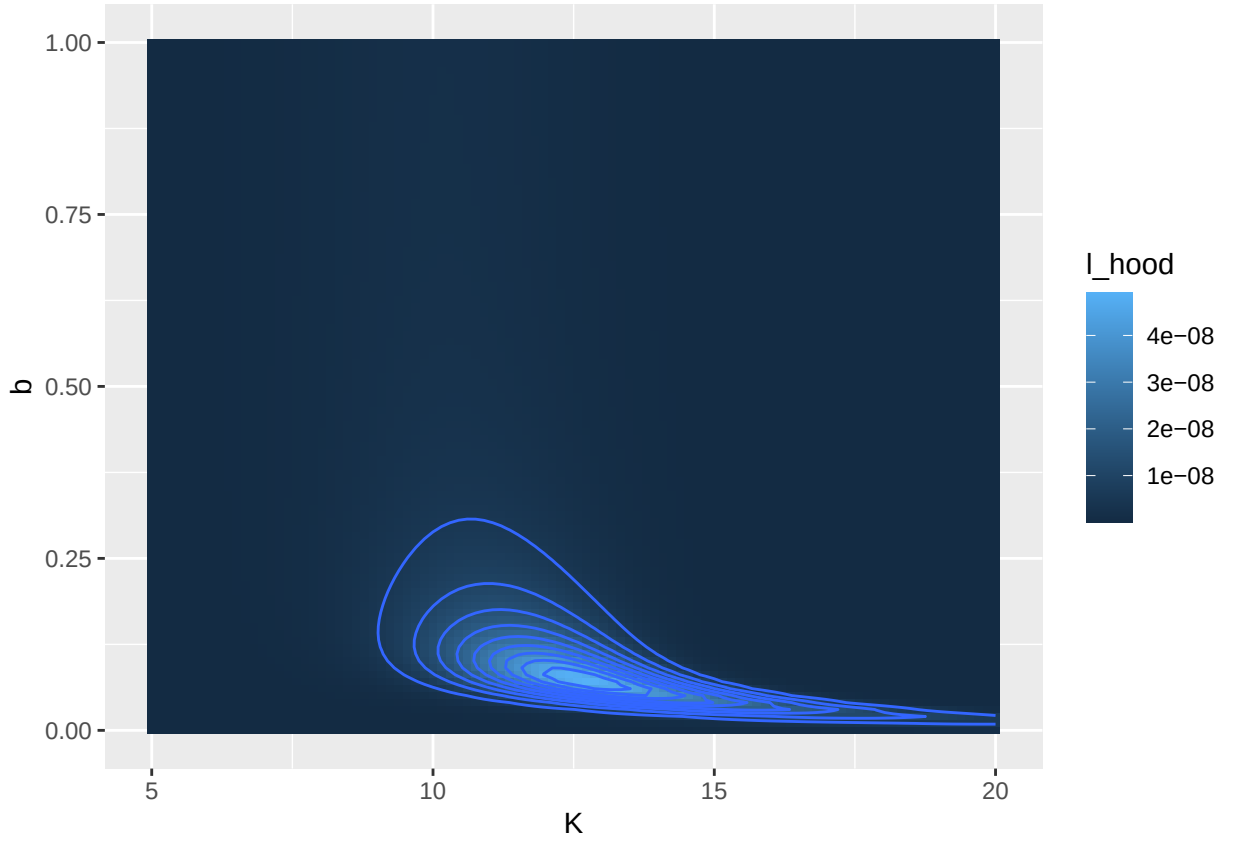


Figure 9.11: Likelihood surface and contour lines for the Gause dataset.

Similar to before, let's take this step by step:

- The command `my_likelihood` just puts the likelihood values in a data frame.
- The `ggplot` command is similar as before.
- We use `geom_tile` to visualize the likelihood surface. There are three required inputs from the `my_likelihood` data frame: the `x` and `y` axis data values and the `fill` value, which represents the height of the likelihood function.
- The command `stat_contour` draws the contour lines, or places where the likelihood function is the same. Notice how we used `z = l_hood` rather than `fill` here.

I chose some broad parameter ranges at first, so let's make a likelihood plot, exploring parameters closer to the last optimum value:

```
# Gause model equation
gause_model <- volume ~ K / (1 + exp(log(K / 0.45 - 1) - b * time))


# Identify the (new) ranges of the parameters that we wish to investigate
kParam <- seq(11, 14, length.out = 100)
bParam <- seq(0.1, 0.3, length.out = 100)


# Allow for all the possible combinations of parameters
gause_parameters_rev <- expand.grid(K = kParam, b = bParam)


gause_likelihood_rev <- compute_likelihood(
  model = gause_model,
  data = yeast,
  parameters = gause_parameters_rev,
  logLikely = FALSE
```

```
  )

# Report out the optimum values
opt_value_rev <- gause_likelihood_rev$opt_value

opt_value_rev
```

```
## # A tibble: 1 x 4
##       K     b   l_hood log_lik
##   <dbl> <dbl>    <dbl> <lgl>
## 1  12.8 0.241 0.000349 FALSE
```

```
# Define the likelihood values
my_likelihood_rev <- gause_likelihood_rev$likelihood

# Make a contour plot
ggplot(data = my_likelihood_rev) +
  geom_tile(aes(x = K, y = b, fill = l_hood)) +
  stat_contour(aes(x = K, y = b, z = l_hood)) +
  geom_point(data = opt_value_rev, aes(x = K, y = b), color = "red")
```

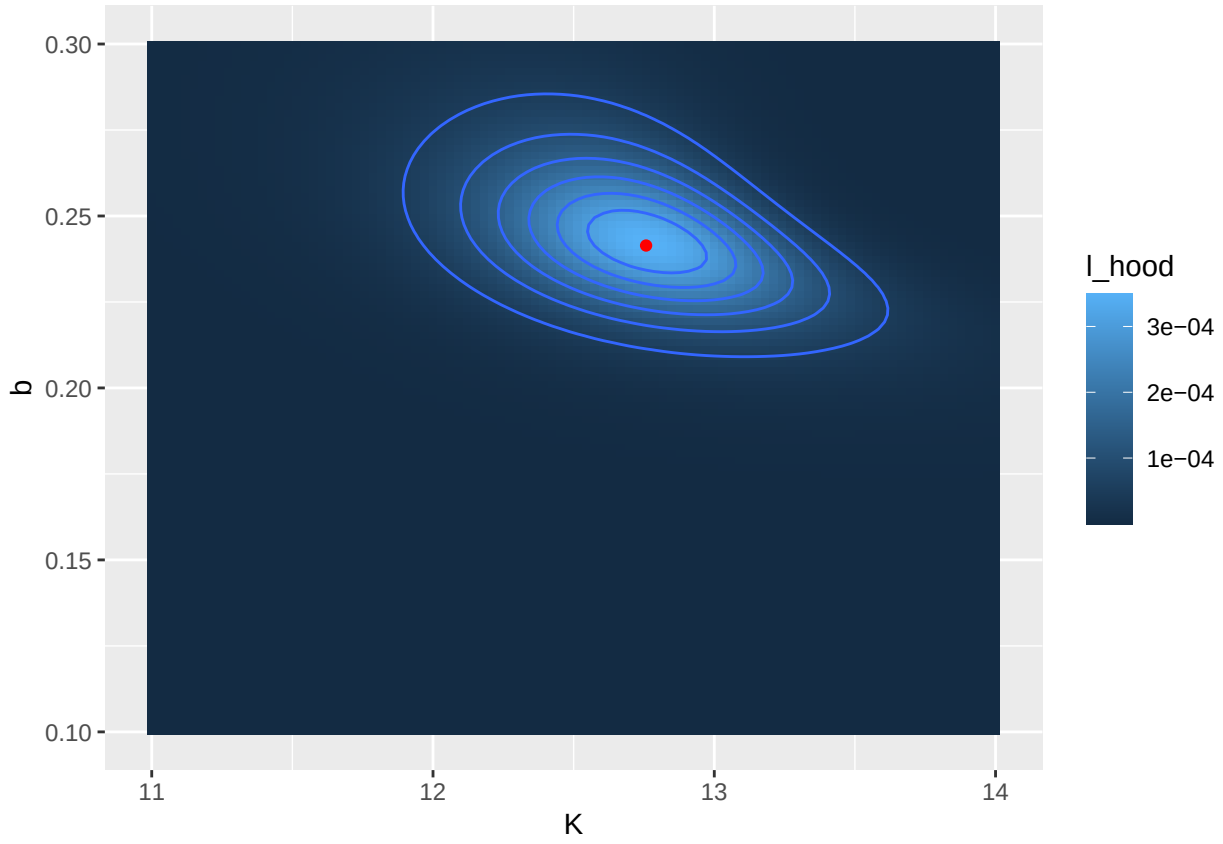


Figure 9.12: Revised likelihood surface. The computed location of the optimum value is shown as a red point.

The reported values for K (12.8) and b (0.241) may be close to what was reported from Figure 9.11. Notice that in Figure 9.12 I also added in the location of the optimum point with the code `geom_point(data=opt_value_rev,aes(x=k,y=b),color='red')`.

Finally we can use the optimized parameters to compare the function against the data (Figure 9.13.

```
# Define the parameters and the times that you will evaluate the equation
my_params <- gause_likelihood_rev$opt_value
time <- seq(0, 60, length.out = 100)
```

```
# Get the right hand side of your equations
new_eq <- gause_model %>%
  formula.tools::rhs()

# This collects the parameters and data into a list
in_list <- c(my_params, time) %>% as.list()

# The eval command evaluates your model
out_model <- eval(new_eq, envir = in_list)


# Now collect everything into a dataframe:
my_prediction <- tibble(time = time, volume = out_model)


ggplot() +
  geom_point(
    data = yeast,
    aes(x = time, y = volume),
    color = "red",
    size = 2
  ) +
  geom_line(
    data = my_prediction,
    aes(x = time, y = volume)
  ) +
  labs(x = "Time", y = "Volume")
```

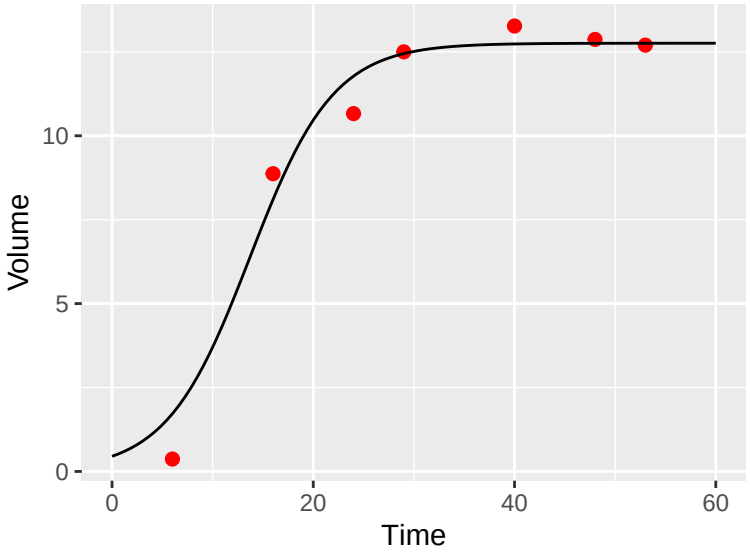


Figure 9.13: Model and data comparison of the `yeast` dataset from maximum likelihood estimation.

All right, this code block has some new commands and techniques that need explaining. Once we have the parameter estimates we need to compute the modeled values.

- First we define the `params` and the `time` we wish to evaluate with our model.
- We need to evaluate the right hand side of $y = \frac{K}{1+e^{a+bt}}$, so the definition of `new_eq` helps to do that, using the package `formula.tools`.
- The `%>%` is the `tidyverse` https://r4ds.had.co.nz/pipes.html#pipes. This is a very useful command to help make code more readable!
- `in_list <- c(params,my_time) %>% as.list()` collects the parameters and input times in one list to evaluate the

model with `out_model <- eval(new_eq,envir=in_list)`
- In order to plot we make a data frame `my_prediction`

And the rest of the plotting commands you should be used to.

9.5 Exercises

Exercise 9.1. Algebraically solve the equation $0.45 = \dfrac{K}{1+e^a}$ for K.

Exercise 9.2. Evaluate $\displaystyle\int_0^5 2e^{-2x}\,dx$ by hand. Then use `R` to compute the value of $\displaystyle\int_0^5 2e^{-2x}\,dx$. Does your computed answer match with what you found in `R`?

Exercise 9.3. Make a plot of the normal density distribution with $\mu = 2$ and $\sigma = 0.1$ for $0 \leq x \leq 4$. Then use `R` to compute the following integral: $\displaystyle\int_0^4 f(x)\,dx$, where $f(x)$ is the normal density function.

Exercise 9.4. Visualize the likelihood function for the `yeast` dataset, but in this case report out and visualize the loglikelihood. (This means that you are setting the option `logLikely = TRUE` in the `compute_likelihood` function.) Compare the loglikelihood surface to Figure 9.12.

Exercise 9.5. When we generated our plot of the likelihood function in Figure 9.9 we assumed that $\sigma = 1$ in Equation (9.9). For this exercise you will explore what happens in Equation (9.9) as σ increases or decreases.

a. Use desmos to generate a plot of Equation (9.9), but let σ be a slider. What happens to the shape of the likelihood function as σ increases?
b. How does the estimate of b change as σ changes?
c. The spread of the distribution (in terms of it being more peaked or less peaked)is a measure of uncertainty of a parameter estimate. How does the resulting parameter uncertainty change as σ changes?

Exercise 9.6. Using Equation (9.9) with $\sigma = 1$:

a. Apply the natural logarithm to both sides of this expression.
b. Using properties of logarithms, show that the loglikelihood function $\ln(L(b)) = -2\ln(2) - 2\ln(\pi) - (3-b)^2 - (5-2b)^2 - (4-4b)^2 - (10-4b)^2$.
c. Make a plot of the log likelihood function (in desmos or R). Where is this function optimized? Is it a maximum or a minimum value?
d. Compare this likelihood estimate for b to what was found in Figure 9.9.

Exercise 9.7. Consider the linear model $y = a + bx$ for the following dataset:

x	y
1	3
2	5
4	4
4	10

a. With the function `compute_likelihood`, generate a contour plot of both the likelihood and log-likelihood functions.
b. Make a scatterplot of these data with the equation $y = a + bx$ with your maximum likelihood parameter estimates.
c. Earlier when we fit $y = bx$ we found $b = 1.86$. How does adding a as a model parameter affect your estimate of b?

Exercise 9.8. For the function $P(t) = \dfrac{K}{1+e^{a+bt}}$, with $P(0) = P_0$, determine an expression for the parameter a in terms of K, b, and P_0.

Exercise 9.9. The values returned by the maximum likelihood estimate for Equation (9.11) were a little different from those reported in Gause (1932):

Parameter	Maximum Likelihood Estimate	Gause (1932)
K	12.7	13.0
b	0.24242	0.21827

Make of plot of the function $y = \dfrac{K}{1+e^{a-bt}}$ with $a = \ln\left(\dfrac{K}{0.45} - 1\right)$ for both parameter values, along with the `yeast` data. to generate plots with the `yeast` data with the curves with parameters from both the Maximum Likelihood estimate and from Gause (1932). Which approach does a better job representing the data?

Exercise 9.10. An equation that relates a consumer's nutrient content (denoted as y) to the nutrient content of food (denoted as x) is given by: $y = cx^{1/\theta}$, where $\theta \geq 1$ and c are both constants.

a. Use the dataset `phosphrous` make a scatter plot with the variable `algae` on the horizontal axis, `daphnia` on the vertical axis.
b. Generate a contour plot for the likelihood function for these data. You may assume $1 \leq \theta \leq 20$ and $0 \leq c \leq 5$. What are the values of θ and c that optimize the likelihood? *Hint:* for the dataset `phosphorous` be sure to use the variables $x =$`algae` and $y =$`daphnia`.
c. With your values of c and θ add the function W to your scatterplot and compare the fitted curve to the data.

Exercise 9.11. A dog's weight W (pounds) changes over D days according to the following function:

$$W = f(D, p_1, p_2) = \frac{p_1}{1 + e^{2.462 - p_2 D}} \tag{9.12}$$

where we have the parameters p_1 and p_2. The dataset `wilson` shows how the weight of a dog named Wilson (adapted from here).

a. Make a scatterplot with the `wilson` data. What is the long term weight of the dog?
b. Generate a contour plot for the likelihood function for these data. What are the values of p_1 and p_2 that optimize the likelihood? *You may assume that p_1 and p_2 are both positive.*
c. With your values of p_1 and p_2 add the function W to your scatterplot and compare the fitted curve to the data.

Exercise 9.12. Consider the following data which represents the temperature over the course of a day:

Hour	0	1	2	3	4	5	6	7	8	9	10	11	12
Temperature (°F)	54	53	55	54	58	58	61	63	67	66	67	69	68
Hour	13	14	15	16	17	18	19	20	21	22	23	24	
Temperature (°F)	68	66	67	63	60	59	57	56	53	52	54	53	

A function that describes these data is $T = A + B\sin\left(\dfrac{\pi}{12}\cdot H\right) - C\cos\left(\dfrac{\pi}{12}\cdot H\right)$, where H is the hour and T is the temperature. Use the function `compute_likelihood` to determine maximum likelihood parameter estimates for A, B, and C. The values of A, B, and C are all positive.

Chapter 10

Cost Functions & Bayes' Rule

In Section 9 we examined likelihood functions which were needed when combining a model with data using probability density functions. In this section we will study this idea of parameter estimation using *cost functions*, which is another approach to the parameter estimation problem.

10.1 Cost functions: likelihood functions in disguise

Another approach that can be incorporated into parameter estimation is the idea of a *cost function*. Let's start again with the linear regression problem from Section 9:

> Assume we have the following (limited) data set of points that we wish to fit a function of the form $y = bx$ (note, we are forcing the intercept term to be zero).

x	y
1	3
2	5
4	4
4	10

Here, we will determine b through minimizing the square residual. We do this by computing the residual, or the expression $y - bx$. Let's extend out the above table a little more:

x	y	bx	$y - bx$
1	3	b	$3 - b$
2	5	$2b$	$5 - 2b$
4	4	$4b$	$4 - 4b$
4	10	$4b$	$10 - 4b$

You can see how this residual changes $y - bx$ for different values of b:

$y - bx$	$b = 1$	$b = 3$	$b = -1$
$3 - b$	2	0	4
$5 - 2b$	3	-1	7
$4 - 4b$	0	-8	8
$10 - 4b$	6	-2	14

Notice that the values of the residual at each (x, y) pair change as b changes - some of the residuals can be negative and some can be positive. If we were to assess the overall residuals as a function of the value of b, we need to take into account not just the value of the residual (positive or negative), but rather the *magnitude* of the square residual. So for example, the square

residual when $b = 1$ is:

$$\text{Square residual: } 2^2 + 3^2 + 0^2 + 6^2 = 49 \tag{10.1}$$

The other square residuals are 68 when $b = 3$ and 325 when $b = -1$. So of these choices for b, the one that minimizes the square residual is $b = 1$.

Let's generalize this to determine a function to compute the square residual for any value of b. We will call this function $S(b)$. How we do that is if we take the sum of the square difference (or the residual), we have:

$$S(b) = (3 - b)^2 + (5 - 2b)^2 + (4 - 4b)^2 + (10 - 4b)^2 \tag{10.2}$$

This looks like a function of one variable. Let's make a plot of $S(b)$ in Figure 10.1. Notice how the plot of $S(b)$ looks like a really nice quadratic function, with a minimum at $b = 1.86$. Did you notice that this value for b is the same value for the minimum that we found in the likelihood function from Section 9? In fact, if we multiplied out $S(b)$ and collected terms, this *would* be a quadratic function - which has a well defined optimum value that you can find using calculus.

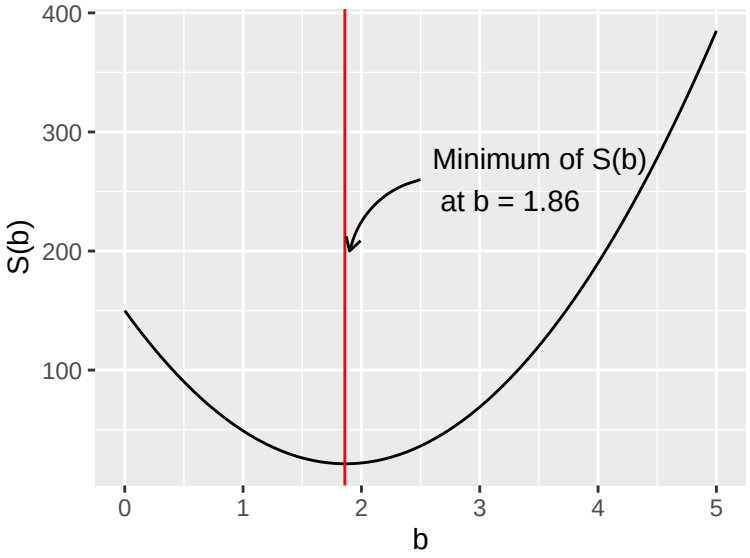


Figure 10.1: The square residual $S(b)$. The vertical line denotes the minimum value at $b = 1.561$.

Let's compare the value of b to the best fit line in Figure 10.2.

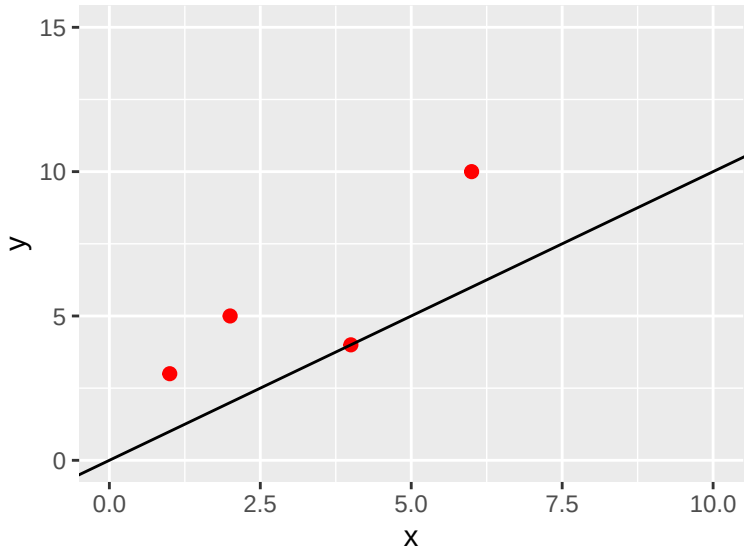


Figure 10.2: Data with the best fit line

The cost function approach described above seems to be working out well in that we have a value for b, but it also looks like a lot of the data lies above the best fit line. We can address that later, but one approach is to include the uncertainty on each of the measured values in the cost function. The uncertainty may be the same (σ) for all measurements or it could vary from measurement to measurement. In both cases we divide each of the components of the cost function by the given uncertainty. We can represent this cost function more generally using $\sum$ notation:

$$S(\vec{\alpha}) = \sum_{i=1}^{N} \frac{(y_i - f(x, \vec{\alpha}))^2}{\sigma^2} \tag{10.3}$$

So examining Equation (10.2) to Equation (10.3) we have $N = 4$, $\sigma = 1/\sqrt{2}$, and $f(x_i, \vec{\alpha}) = bx$.

10.2 Connection to likelihood functions

We call the function $S(b)$ the *cost function*. There is something big going on here. In Section 9 we also defined the log likelihood function (Equation (10.4), from Equation (9.10) with $N = 4$ and $\sigma = 1$):

$$\ln(L(\vec{\alpha}|\vec{x}, \vec{y})) = -2\ln(2) - 2\ln(\pi) - (3 - b)^2 - (5 - 2b)^2 - (4 - 4b)^2 - (10 - 4b)^2 \tag{10.4}$$

If we compare this log likelihood equation (Equation (10.4)) with Equation (10.2), then we $\ln(L(\vec{\alpha}|\vec{x}, \vec{y})) = -2\ln(2) - 2\ln(\pi) - S(b)$. **This is no coincidence**. Likelihood functions are similar in nature to cost functions. You may be thinking: but Equation (10.4) contains the extra factors of $- - 2\ln(2) - 2\ln(\pi)$ - but you need not worry. Here is why: our goal is to *optimize* a cost or log-likelihood function. What these extras terms do (for constant σ) is shift the graph of the log-likelihood function *vertically* but not *horizontally*. Vertically shifting a function doesn't not change the *location* of an optimum value (Why? Think back to derivatives from Calculus I).

Recognizing the connection between cost and likelihood functions and their goal of optimization leads to a key observation: **A quadratic cost function yields the same results as likelihood function assuming the residuals are normally distributed.**

10.3 Extending the cost function

The cost function $S(b)$ can be extended additionally to incorporate other types of data. For example, if we knew there was a given range of values that would make sense (say b is near 1.3 with a standard deviation of 0.1), we should be able to incorporate this information into the cost function. A naive approach would be just to add in some additional squared term ($\tilde{S}(b)$, Equation (10.5)), which is plotted in Figure 10.3.

$$\tilde{S}(b) = (3 - b)^2 + (5 - 2b)^2 + (4 - 4b)^2 + (10 - 4b)^2 + \frac{(b - 1.3)^2}{0.1^2} \tag{10.5}$$

```
## Warning: Removed 191 row(s) containing missing values (geom_path).
```

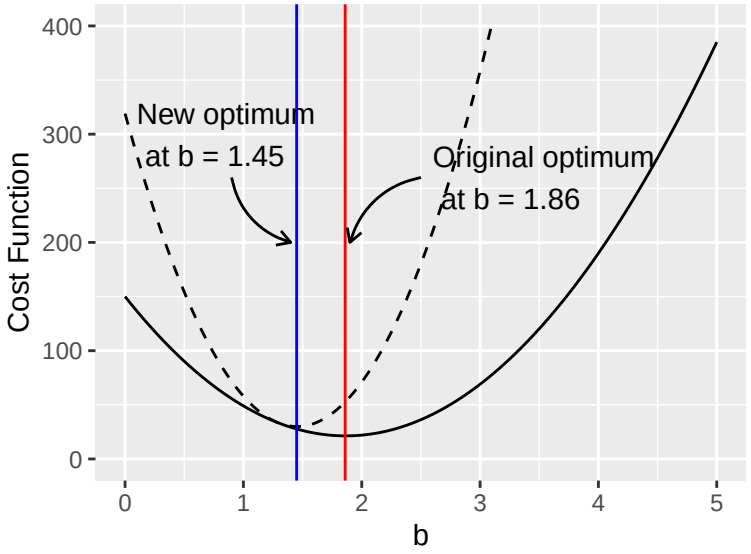


Figure 10.3: Comparing two cost functions $S(b)$ (black) and $\tilde{S}(b)$ (black dashed line)

Aha! Figure 10.3 shows how the revised cost function $\tilde{S}(b)$ changes the optimum value. Numerically this works out to be $\tilde{b} = 1.45$. In a homework problem you verify this new minimum value and compare the fitted value to the value of $b = 1.86$.

Adding this prior information seems like an effective approach - and perhaps is applicable for problems in the sciences. Many times a scientific study wants to build upon the existing body of literature and to take that into account. This approach of including prior information into the cost function borrows elements of Bayes' rule - so let's have a digression into what that means.

10.4 Conditional Probabilities and Bayes' Rule

In order to understand Bayesian statistics we first need to understand Bayes' rule and conditional probability. So let's look at an example.

Example 10.1. The following table shows results from a survey of people's views on the economy and whether or not they voted for the President in the last election. Percentages are reported as decimals. Probability tables are a clever way to organize this information.

Probability	Optimistic view on economy	Pessimistic view on economy	Total
Voted for the president	0.20	0.20	0.40
Did not vote for president	0.15	0.45	0.60
Total	0.35	0.65	1.00

Compute the probability of having an optimistic view on the economy.

Solution. Based on the probability table, we define the following probabilities:

- The probability you voted for the President *and* have an **optimistic** view on the economy is 0.20
- The probability you **did not** vote for the President *and* have an **optimistic** view on the economy is 0.15
- The probability you voted for the President *and* have an **pessimistic** view on the economy is 0.20
- The probability you **did not** vote for the President *and* have an **pessimistic** view on the economy is 0.45

We calculate the probability of having an **Optimistic view on the economy** by adding the probabilities with an optimistic view, whether or not they voted for the president. For this example, this probability sums to $0.20 + 0.15 = 0.35$. On the other hand, the probability you have a pessimistic view on the economy is $0.20 + 0.45 = 0.65$. Notice how the two of these together (probability of optimistic and pessimistic views of the economy is 1, or 100% of the outcomes.)

10.4.1 Conditional probabilities

A conditional probability is the probability of an outcome given some previous outcome, or $\Pr(A|B)$, where Pr means "probability of an outcome" and A and B are two different outcomes or events. In probability theory you might study the following law of conditional probability:

$$\begin{aligned}\Pr(A \text{ and } B) &= \Pr(A \text{ given } B) \cdot \Pr(B) \\ &= \Pr(A|B) \cdot \Pr(B) \\ &= \Pr(B|A) \cdot \Pr(A)\end{aligned} \tag{10.6}$$

Typically when expressing conditional probabilities we remove "and" and write $P(A \text{ and } B)$ as $P(AB)$ and "given" as $P(A \text{ given } B)$ as $P(A|B)$.

Example 10.2. Sometimes people believe that your views of the economy influence if you are going to vote for the President, so use the information from the table in Example 10.1 to compute the probability you voted for the president *given* you have an optimistic view of the economy.

Solution. To compute the probability you voted for the president *given* you have an optimistic view of the economy is a rearrangement of Equation (10.6):

$$\begin{aligned}&\Pr(\text{Voted for President} \mid \text{Optimistic View on Economy}) = \\ &\frac{\Pr(\text{Voted for President and Optimistic View on Economy})}{\Pr(\text{Optimistic View on Economy})} = \\ &\frac{0.20}{0.35} = 0.57\end{aligned} \tag{10.7}$$

So the probability you compute in Equation (10.7) seems telling. Contrast this percentage to that of the probability you voted for the President, which is 0.4. Perhaps your view of the economy does indeed influences whether or not you would vote to re-elect the President.

10.4.2 Bayes' Rule: Application of Conditional Probabilities

How could we systematically incorporate prior information into a parameter estimation problem? We are going to introduce *Bayes' Rule*, which is a rearrangment of the rule for conditional probability:

$$\Pr(A|B) = \frac{\Pr(B|A) \cdot \Pr(A)}{\Pr(B)} \tag{10.8}$$

It turns out Bayes' Rule is a really helpful way to understand how we can systematically incorporate this prior information into the likelihood function (and by association the cost function). For data assimilation problems our goal is to estimate parameters, given data. So we can think of Bayes' rule in terms of parameters and data:

$$\Pr(\text{ parameters} \mid \text{ data }) = \frac{\Pr(\text{ data} \mid \text{ parameters }) \cdot \Pr(\text{ parameters })}{\Pr(\text{ data })}. \tag{10.9}$$

Here are a few observations from that last equation:

- The term Pr(data | parameters) is similar to the model data residual, or the standard likelihood function.
- If we think of the term Pr(parameters), then prior information is a multiplicative effect on the likelihood function - this is good news! You will demonstrate in the homework that the log likelihood is related to the cost function - so when we added that additional term to form $\tilde{S}(b)$, we accounted for the prior information correctly.
- The expression Pr(parameters | data) is the start of a framework for a probability density function, which should integrate to unity. (You will explore this more if you study probability theory.) In many cases we select parameters that optimize a likelihood or cost function. So the expression in the denominator (Pr(data)) does not change the *location* of the optimum values. This denominator term is called a normalizing constant.

In the following sections we will explore Bayes' Rule in action and how to utilize it for different types of cost functions, but wow - we made some significant progress in our conceptual understanding of how to incorporate models and data.

10.5 Bayes' Rule and Linear Regression

Returning back to our linear regression problem ($y = bx$). We have the following assumptions:

- The data are independent, identically distributed. We can then write the likelihood function as the following:

$$\Pr(\vec{y}|b) = \left(\frac{1}{\sqrt{2\pi}}\right)^4 e^{-\frac{(3-b)^2}{\sigma}} \cdot e^{-\frac{(5-2b)^2}{\sigma}} \cdot e^{-\frac{(4-4b)^2}{\sigma}} \cdot e^{-\frac{(10-4b)^2}{\sigma}} \tag{10.10}$$

- Prior knowledge expects us to say that b is normally distributed with mean 1.3 and standard deviation 0.1. Incorporating this information allows us to write the following:

$$\Pr(b) = \frac{1}{\sqrt{2\pi} \cdot 0.1} e^{-\frac{(b-1.3)^2}{2 \cdot 0.1^2}} \tag{10.11}$$

So when we combine the two pieces of information, the probability of the b, given the data $\vec{y}$ is the following:

$$\Pr(b|\vec{y}) \approx e^{-\frac{(3-b)^2}{2}} \cdot e^{-\frac{(5-2b)^2}{2}} \cdot e^{-\frac{(4-4b)^2}{2}} \cdot e^{-\frac{(10-4b)^2}{2}} \cdot e^{-\frac{(b-1.3)^2}{2 \cdot 0.1^2}} \tag{10.12}$$

Notice we are ignoring the terms $\left(\frac{1}{\sqrt{2\pi}}\right)^4$ and $\frac{1}{\sqrt{2\pi} \cdot 0.1}$, because per our discussion above not including them the does not change the *location* of the optimum value, only the value of the likelihood function. The plot of $\Pr(b|\vec{y})$, assuming $\sigma = 1$ is shown in Figure 10.4:

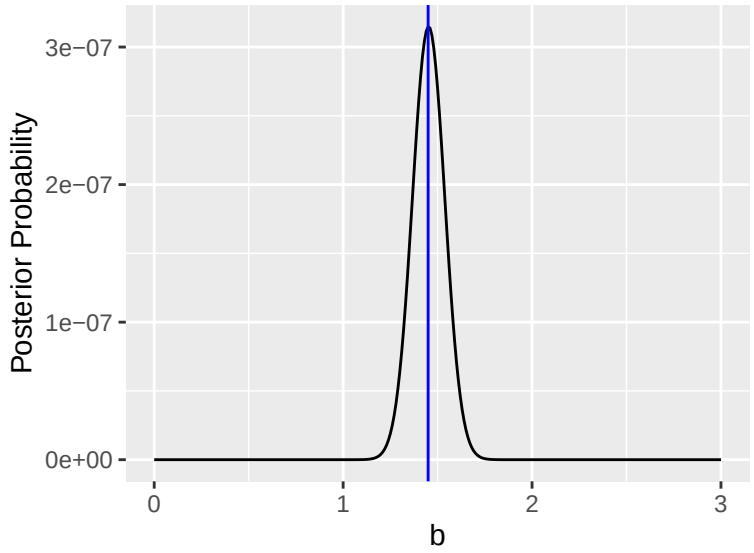


Figure 10.4: Equation (10.12) with optimum value denoted in blue.

It looks like the value that optimizes our posterior probability is $b = 1.45$. This is very close to the value of $\tilde{b}$ from the cost function approach. Again, *this is no coincidence*. Adding in prior information to the cost function or using Bayes' Rule are equivalent approaches. Now that we have seen the usefulness of cost functions and Bayes' Rule we can begin to apply this to larger problems involving more equations and data. In order to do that we need to explore some computational methods to scale this problem up - which we will do so in the next sections.

10.6 Exercises

Exercise 10.1. The following problem works with fitting $y = bx$ as in this section, with the following data:

x	y
1	3
2	5
4	4
4	10

a. Using calculus, show that the cost function $S(b) = (3-b)^2 + (5-2b)^2 + (4-4b)^2 + (10-4b)^2$ has a minimum value at $b = 1.86$.
b. Use a similar approach to determine the minimum of the revised cost function $\tilde{S}(b) = (3-b)^2 + (5-2b)^2 + (4-4b)^2 + (10-4b)^2 + (b-1.3)^2$. Call this value $\tilde{b}$.
c. Make a plot of the cost functions $S(b)$ and $\tilde{S}(b)$ to verify the optimum values.
d. Make a scatter plot with the data and the function $y = bx$ and $y = \tilde{b}x$. How do the two estimates compare with the data?

Exercise 10.2. (Inspired from Hugo van den Berg (2011)) Consider the nutrient equation $y = cx^{1/\theta}$ using the dataset `phosphorous`.

a. Write down a formula for the objective function $S(c, \theta)$ that characterizes this equation (that includes the dataset `phosphorous`).
b. Fix $c = 1.737$. Make a `ggplot` of $S(1.737, \theta)$ for $1 \leq \theta \leq 10$.
c. How many critical points does this function have over this interval? Which value of θ is the global minimum?

Exercise 10.3. Use the cost function $S(1.737, \theta)$ from Exercise 10.2 to answer the following questions:

a. Researchers believe that $\theta \approx 7$. Re-write $S(1.737, \theta)$ to account for this additional (prior) information.
b. How does the inclusion of this additional information change the shape of the cost function and the location of the global minimum?
c. Finally, reconsider the fact that $\theta \approx 7 \pm .5$ (as prior information). How does that modify $S(1.737, \theta)$ further and the location of the global minimum?

Exercise 10.4. Navigate to this desmos file, which you will use to answer the following questions:

a. By adjusting the sliders for a and b, determine the values of a and b that you think best minimizes the objective function.
b. Desmos can do linear regression! To do that, you need to start a new cell and enter in the regression formula: $y_1 \sim c + dx_1$. (We need to use different parameters c and d because a and b are defined above). How do the values of c and d compare to what you found with a and b?
c. Alternatively, you can also define an objective function with absolute value: $S_{mod}(a, b) = \sum_{i=1}^{n} |y_i - (a + bx_i)|$ Implement the absolute value objective function in Desmos and manipulate the slider values for a and b to determine where S_{mod} is minimized. How do those values compare to the least squares estimate?

Exercise 10.5. One way to generalize the notion of prior information using cost functions is to include a term that represents the degree of uncertainty in the prior information, such as σ. For the problem $y = bx$ this leads to the following cost function: $\tilde{S}_{revised}(b) = (3-b)^2 + (5-2b)^2 + (4-4b)^2 + (10-4b)^2 + \dfrac{(b-1.3)^2}{\sigma^2}$.

Use calculus to determine the optimum value for $\tilde{S}_{revised}(b)$, expressed in terms of $\tilde{b}_{revised} = f(\sigma)$ (your optimum value will be a function of σ). What happens to $\tilde{b}_{revised}$ as $\sigma \to \infty$?

Exercise 10.6. For this problem you will minimize some generic functions.

a. Using calculus, verify that the optimum value of $y = ax^2 + bx + c$ occurs at $x = -b/2a$. (You can assume $a > 0$.)
b. Using calculus, verify that the optimum value of $z = e^{-(ax^2+bx+c)^2}$ also occurs at $x = -b/2a$.
c. Algebraically show that $\ln(z) = -y$.
d. Explain why y is similar to a cost function $S(b)$ and z is similar to a likelihood function.

Exercise 10.7. This problem continues the re-election of the President and viewpoint on the economy. Determine the following conditional probabilities:

a. Determine the probability that you **voted for the president** given that you have a **pessimistic view on the economy**.
b. Determine the probability that you **did not vote for the president** given that you have an **pessimistic view on the economy**.
c. Determine the probability that you **did not vote for the president** given that you have an **optimistic view on the economy**.

Exercise 10.8. Incumbents have an advantage in re-election due to wider name recognition, which may boost their re-election chances. Complete the following table, estimating the following probabilities. Please report percentages as decimals.

Probability	Being elected for office	Not being elected for office	Total
Having name recognition	0.55	0.25	0.80
Not having name recognition	0.05	0.15	0.20
Total	0.60	0.40	1.00

Use Bayes' Rule to determine the probability of being elected, given that you have name recognition.

Exercise 10.9. Show how you can derive Bayes' Rule from the law of conditional probability.

Chapter 11

The Bootstrap Method

In Sections and 9 and 10 we saw how the parameter estimation problem is related to optimizing the likelihood or cost function. As an example, Figure 11.1 shows the cost function for the nutrient equation $y = cx^{1/\theta}$ using the dataset `phosphorous`:

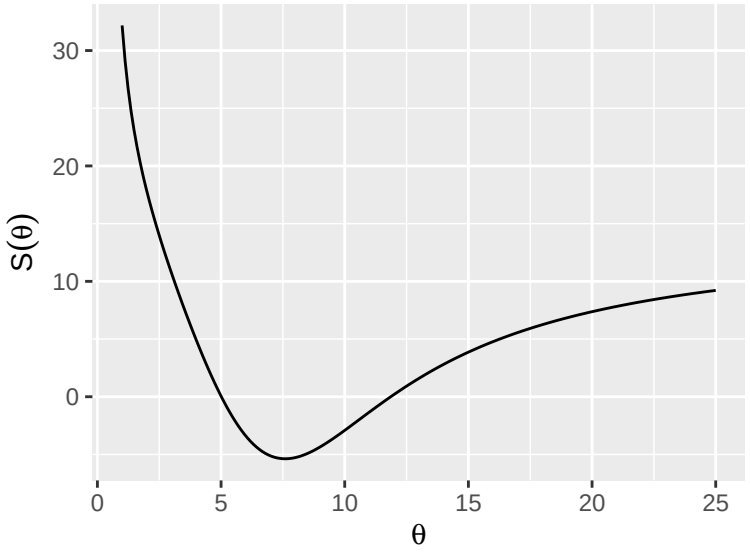


Figure 11.1: Nonlinear cost function plot for 'phosphorous' data set.

While Figure 11.1 shows a clearly defined minimum around $\theta \approx 6$, the shape of the cost function is not quadratic like Figure 10.3 in Section 10. For cases like these, direct optimization of the cost function using techniques learned in calculus may not be computationally feasible. Instead we need to efficiently examine different combinations of parameters, their model output, and to get a result of the sample.

In this section we are going to explore another way to approach this problem. An alternative approach uses the idea of random number generation and *sampling*, which can efficiently determine the the minimum through direct evaluation. To understand the idea of sampling we will study an approach called *bootstrapping*.

We will work you through this step by step, with example code that you can type along the way. As always it will be good to load up the libraries you will be using.

```
library(tidyverse)
library(demodelr)
```

11.1 Plotting histograms in R

In Section 9 we discussed probability distributions. Now we are going to discuss them a little more, but now we will first discuss plotting histograms in R. A quick recap of a histogram: this is a binned plot of data, where there are some predefined

Table 11.1: Weather station data from a Minnesota snowstorm.

date	time	station_id	station_name	snowfall
4/16/18	5:00 AM	MN-HN-78	Richfield 1.9 WNW	22.0
4/16/18	7:00 AM	MN-HN-9	Minneapolis 3.0 NNW	19.0
4/16/18	7:00 AM	MN-HN-14	Minnetrista 1.5 SSE	12.5
4/16/18	7:00 AM	MN-HN-30	Plymouth 2.4 ENE	18.5
4/16/18	7:00 AM	MN-HN-58	Champlin 1.5 ESE (118)	20.0
4/16/18	7:00 AM	MN-HN-89	Edina 1.7 N	11.0
4/16/18	7:00 AM	MN-HN-110	Edina 1.9 SSE	15.5
4/16/18	7:00 AM	MN-HN-134	Brooklyn Center 1.1 E	13.5
4/16/18	7:00 AM	MN-HN-150	Maple Grove 1.8 NE	22.0
4/16/18	8:00 AM	MN-HN-17	Eden Prairie 3.3 WSW	16.0
4/16/18	8:00 AM	MN-HN-72	Maple Grove 2.9 NE	13.0
4/16/18	8:00 AM	MN-HN-175	Bloomington 2.0 SE	13.1
4/16/18	8:30 AM	MN-HN-19	Edina 1.3 SW	11.0
4/16/18	8:30 AM	MN-HN-31	Maple Grove 1.0 NNE	19.5
4/16/18	6:00 PM	MN-HN-215	Richfield 1.4 W	18.0
4/16/18	10:30 PM	MN-HN-5	New Hope 1.9 S	13.0

bins and we count the number of observations in each bin.

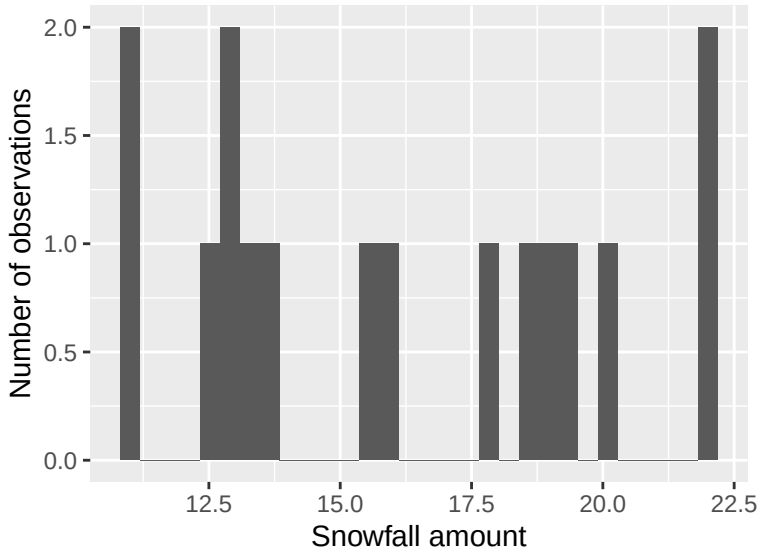
Consider the dataset of snowfall observations from weatherstations in Minnesota shown in Table 11.1, with the following table.

```
knitr::kable(snowfall, caption = "Weather station data from a Minnesota snowstorm.")
```

A histogram is an easy way to view the distribution of measurements. Doing a histogram in R is easy to do:

```
ggplot(data = snowfall) +
  geom_histogram(aes(x = snowfall), ) +
  labs(
    x = "Snowfall amount",
    y = "Number of observations"
  )
```

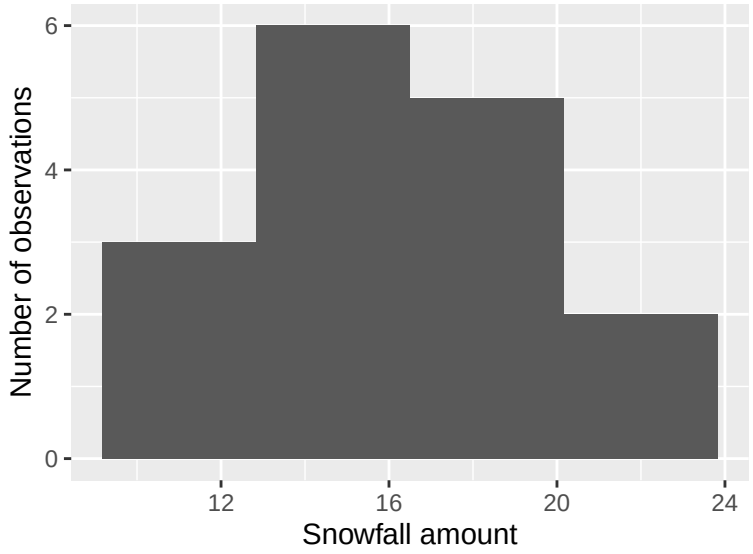
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



This code introduces `geom_histogram`. Notice , which has two key inputs:

- The code `aes(x = snowfall)` is computing the histogram for the `snowfall` column in the dataset `snowfall`. You may have received a warning about the bins `stat_bin()` using `bins = 30`. Pick better value with `binwidth`, so let's adjust the number of bins to 4:

```
ggplot() +
  geom_histogram(data = snowfall, aes(x = snowfall), bins = 4) +
  labs(
    x = "Snowfall amount",
    y = "Number of observations"
  )
```



The resulting histogram may look blockier, but that is ok.

11.2 Statistical theory: Sampling distributions

Now we are going to discuss the process of sampling a distribution. Figure 11.2 is a histogram of 50 random samples of the standard normal distribution (with $\mu = 0$ and $\sigma = 1$):

```
my_data <- tibble(samples = rnorm(50))

ggplot() +
  geom_histogram(data = my_data, aes(x = samples), bins = 10) +
  labs(
    x = "Samples",
    y = "N"
  )
```

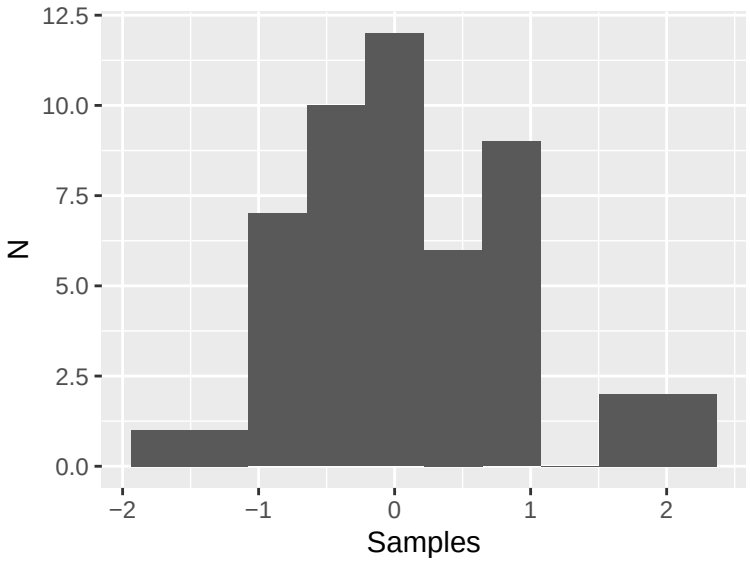


Figure 11.2: Fifty samples of the standard normal distribution.

This histogram *looks* like a normal distribution, but since we only have 50 samples that may not be enough data to adequately justify that conclusion, especially in a statistical sense. Now let's see what happens if we increase the number of samples by factors of 10:

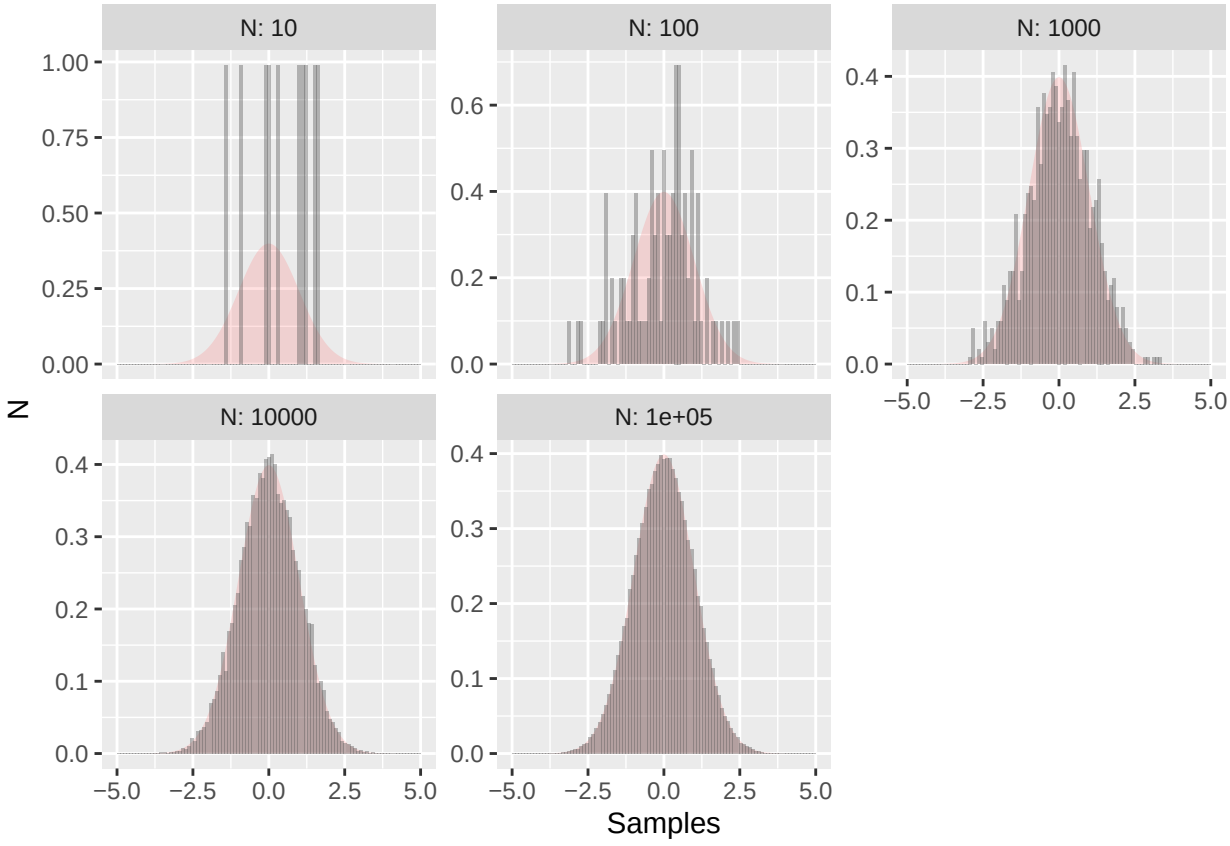


Figure 11.3: More random samples of the normal distirubtion with the true distribution outlined in red.

Figure 11.3 also shows the true distribution shaded in red - clearly as the number of data points increases the more the random sample approaches the true distribution. This is the concept underlying statistical inference: process of drawing conclusions about the entire population based on a sample. By analogy, there is a difference between the true distribution (red shading in Figure 11.3) and samples (grey histograms).

When we discuss likelihood and cost functions, we assume that the parameters follow some underlying probability distribution (such as normal, uniform, etc). However sometimes if the model is extremely nonlinear, the cost function ends up being non-linear. However by *sampling* the distribution we can characterize the distribution.

11.2.1 Sampling the empirical distirbution

We can apply this same principle to measurements. The distribution of measurements can be called an *empirical distribution*. To examine data, we will use a dataset of weather and precipitation, which is collected in a cooperative network of local observers. These stations come from the Twin Cities of Minneapolis and St. Paul and surrounding suburbs:

date	time	station_id	station_name	precip
9/21/2018	05:00:00	MN-HN-52	Champlin 1.8 ESE	2.73
9/21/2018	05:00:00	MN-HN-78	Richfield 1.9 WNW	4.27
9/21/2018	05:00:00	MN-HN-154	Dayton 3.2 SE	2.25
9/21/2018	05:00:00	MN-HN-156	Minnetonka 1.8 SW	3.41
9/21/2018	05:15:00	MN-HN-148	Maple Grove 4.0 SW	2.21
9/21/2018	06:00:00	MN-HN-37	Plymouth 0.8 SSE	2.70
9/21/2018	06:00:00	MN-HN-39	Rockford 0.5 NE	2.55
9/21/2018	06:00:00	MN-HN-49	Maple Grove 2.4 E	2.56
9/21/2018	06:00:00	MN-HN-185	Long Lake 1.7 NNW	2.05
9/21/2018	06:03:00	MN-HN-220	Minnetrista 3.9 SW	1.73

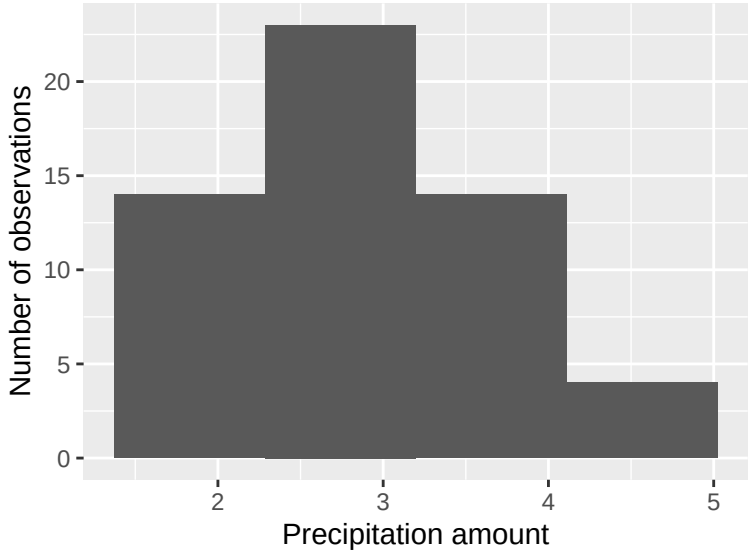
While it is great to have only one value is reported for precipitation amounts, so the question is: *What could be a representative number for the average amount of rainfall received for this storm?*

Let R be the distribution of rainfall in the Twin Cities. The measurements are *samples* of this distribution. One way to say is that the average of the precipitation data (2.8872727 inches) is good enough, but what we don't know is how well that average value approximates the expected value of the distribution for R.

As an exploratory data analysis approach, one way we can do this is by a histogram of the precipitation measurements:

```
ggplot() +
  geom_histogram(data = precipitation, aes(x = precip), bins = 4) +
  labs(
    x = "Precipitation amount",
    y = "Number of observations"
  )
```

```
## Warning: Removed 1 rows containing non-finite values (stat_bin).
```



The precipitation measurement illustrates the difference between a population (the true distribution of measurements) and a sample (what people observe). The histogram shown is called the *empirical distribution* of these data. For the purposes we get here, we are after what is called the *sample statistic*, which is usually the mean or standard deviation.

This distribution looks bimodal with a lot of variability. How could we account for the representative value of the distribution R? Each of the entries in the `precipitation` data frame represents a measurement made by a particular observer. To get the true distribution we would need to add more observers (such as when we sampled the normal distribution), but that isn't realistic as the event has already passed.

The way around this is a bootstrap sample, which is a sample of the original dataset with replacement. Sampling with replacement is the process of remaking a dataset, but you get to reselect *from the entire dataset at the same time*. This is easily done with the command `slice_sample`:

```
p_new <- slice_sample(precipitation, prop = 1, replace = TRUE)
```

Let's break this code down:

- We are sampling the `precipitation` data frame with replacement (`replace = TRUE`). Here is how replacement works: say you have each of the precipitation measurements written on a piece of paper in a hat. You draw one slip of paper, record the measurement, and then replace that slip of paper back in the hat to draw again, until you have as many measurements as the original data frame (in this case this is 56. -The command `prop=1` means that we are sampling 100% of the `precipitation` data frame.

One thing that we can do is compute the mean (average) and the standard deviation of the sample:

```
slice_sample(precipitation, prop = 1, replace = TRUE) %>%
  summarize(
    mean = mean(precip, na.rm = TRUE),
    sd = sd(precip, na.rm = TRUE)
  )
```

```
## # A tibble: 1 x 2
##    mean    sd
##   <dbl> <dbl>
## 1  2.90 0.738
```

How this code works is to first to do the sampling, and then the summary. The command `summarize` is collapsing the `precipitation` data frame and computes the mean and the standard deviation `sd` of the column `precip`. We have the command `na.rm=TRUE` to remove any `NA` values that may affect the computation. If we want to run this multiple times we need some more powerful functionality here. The `purrr` package has the wonderful command `map`, which allows you to quickly iterate through a list quickly.

```
purrr::map_df(
  1:10,
  ~ (
    slice_sample(precipitation, prop = 1, replace = TRUE) %>%
      summarize(
        mean = mean(precip, na.rm = TRUE),
        sd = sd(precip, na.rm = TRUE)
      )
  ) # Close off the tilde ~ ()
) # Close off the map_df
```

```
## # A tibble: 10 x 2
##     mean    sd
##    <dbl> <dbl>
##  1  2.86 0.657
##  2  2.84 0.796
##  3  2.89 0.687
##  4  2.99 0.852
##  5  2.86 0.737
##  6  2.87 0.734
##  7  2.74 0.707
##  8  2.86 0.673
##  9  2.90 0.701
## 10  2.95 0.742
```

What should be returned is a dataframe with columns `mean` and `sd` that represents the mean and standard deviation of each bootstrap sample. The process of randomly sampling a dataset is called *bootstraping*.

Let's review this code bit by bit. Notice that I've written this on multiple lines to aid in reading.

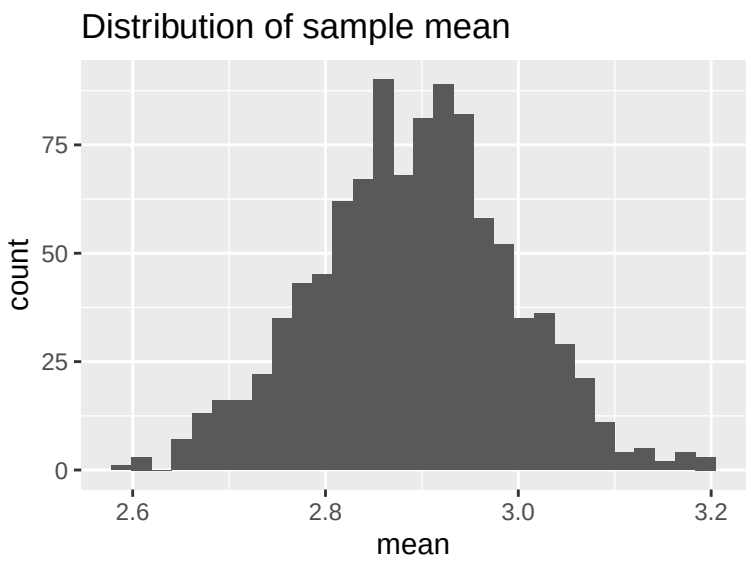
- The basic structure is `map_df(1:N,~(COMMANDS))`, where `N` is the number of times you want to run your code (in this case `N=10`).
- The second part `~(COMMANDS)` lists the different commands we want to re-run (in this case it is our mean and standard deviation of the data frame).

I can appreciate this programming might be a little tricky to understand and follow - don't worry - the goal is to give you a tool that you can easily adapt to a situation. What I would do in a use-case scenario is to first get a working example (where you compute the mean and standard deviation), and then use the `map_df` to return your result.

The final step would be to visualize the mean and the standard deviation. Let's re-run our example and then plot:

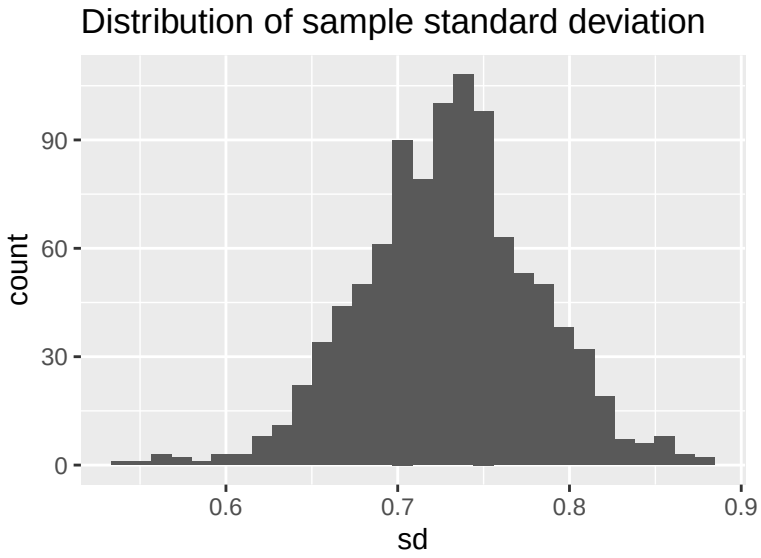
```
bootstrap_samples <- purrr::map_df(
  1:1000,
  ~ (
    slice_sample(precipitation, prop = 1, replace = TRUE) %>%
      summarize(
        mean = mean(precip, na.rm = TRUE),
        sd = sd(precip, na.rm = TRUE)
      )
  ) # Close off the tilde ~ ()
) # Close off the map_df

# Now make the histogram plots for the mean and standard deviation:
ggplot(bootstrap_samples) +
  geom_histogram(aes(x = mean)) +
  ggtitle('Distribution of sample mean')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(bootstrap_samples) +
  geom_histogram(aes(x = sd)) +
  ggtitle('Distribution of sample standard deviation')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Excellent! This is shaping up nicely. Once we have sampled as much we want, *then* investigate the distribution of the computed sample statistics (we call this the *sampling distribution*). It turns out that sample statistics (such as the mean or the standard deviation) will in the long run - approximate the true distribution statistic (such as the mean or standard deviation). This is an example of a non-parametric bootstrap - meaning we are not trying to force a priori a distribution onto the data.

One last task: summarize the distribution of bootstrap means and standard deviations. We will do this using a confidence interval, which comes from the percentiles of the distribution. Let's take a look at the distribution of bootstrap means. The 2.5 percentile is approximately 2.7 inches. This means 2.5% of the distribution is at 2.7 inches or less. The median (50th percentile) is 2.9, so half of the distribution for is 2.9 inches or less. The 97.5 percentile is approximately 3.1, so 97.5% of the distribution is 3.1 inches or less. If we take the difference between 2.5% and 97.5% that is 95%, so 95% of the distribution is contained between 2.7 and 3.1 inches. If we are using the bootstrap mean, we would report that the median rainfall is 2.9 with a 95% confidence interval of 2.7 to 3.1. The confidence interval to give some indication of the uncertainty in the measurements.

Here is how we would compute these different statistics using the `quantile` command, which we need to do separately for the mean and standard deviation:

```
quantile(bootstrap_samples$mean, probs = c(0.025, 0.5, .975))
```

```
##     2.5%      50%    97.5%
## 2.687598 2.894414 3.089496
```

```
quantile(bootstrap_samples$sd, probs = c(0.025, 0.5, .975))
```

```
##      2.5%       50%     97.5%
## 0.6306580 0.7309818 0.8290933
```

Notice how we using the `probs=c(0.025,0.5,.975)` command to compute the different quantiles - they need to be scaled between 0 and 1.

This code works through the bootstrap for 100 samples.

11.3 Bootstapping with linear regression

Hopefully you can see how much more useful the bootstrap is in terms calculating sample statistics. Another neat application of the bootstrap is determining and expands the provenance of the data. A key goal is to get at the population level parameters, rather than the data level parameters. Can we use this as a modeling tool? - YES!

In this case, the "population" represent the distribution on possibilities of estimated regression parameters. A set of measurements is a *sample* of these parameters.

We are going to return to our example of the global temperature dataset from Section 8, and do the following steps:

- Do a bootstrap sample of the data 100 times.
- With each sample, fit a quadratic function.
- Following each fit, examine the histogram of each fitting coefficient.

Here is how we can do this in with the command `bootstrap` from the `modelr` pacakge:

```
# Define the regression formula
regression_formula <- globalTemp ~ 1 + yearSince1880 + I(yearSince1880^2)

# Generate the bootstrap samples
boot <- modelr::bootstrap(global_temperature, n = 100)

# You might not need to modify this code as much
models <- purrr::map(boot$strap, ~ lm(regression_formula, data = .))
tidied <- purrr::map_df(models, broom::tidy, .id = "id")

# Make the histogram.  We will make a facetted plot
ggplot(data = tidied) +
  geom_histogram(aes(x = estimate)) +
  facet_wrap(~term, scales = "free")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

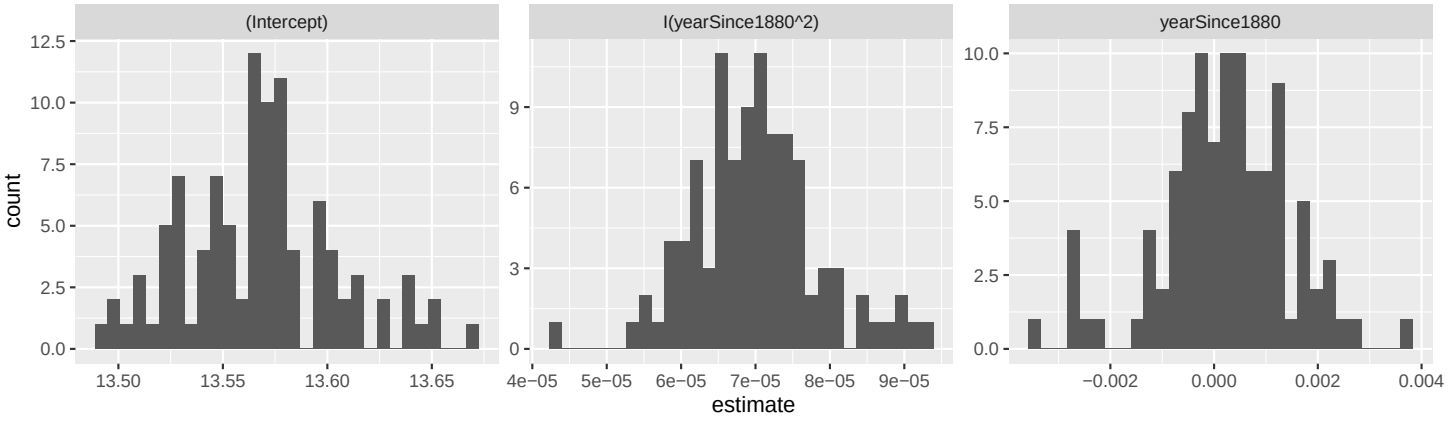


Figure 11.4: Distribution of bootstrap regression coefficients from the `global_temperature` dataset.

There several new elements of this code, so let's break this down bit by bit:

- The code `boot <- modelr::bootstrap(global_temperature, n=100)` creates 100 bootstrap samples of the `global_temperature` dataset. The list `boot` has two entries: (1) `.id` which is a reference (1 to 100) of a particular bootstrap sample and (2) `strap` which contains the information about the bootstrap sample.
- The next portion of the code applies the `map` command (similar to `map_df`) to first compute the linear regression for each bootstrap sample. The linear regression fits are stored in the data frame `models`.
- Finally we extract out the information about parameters using the command `tidy` from the `broom` package. The data frame `tidied` is organized by the bootstrap sample `.id` and has several columns, but we are going to focus on two of them: `estimate`, which tells you the numerical value of the coefficient in the column `term`. Other information about error estimates and statistical significance are included.
- In our histogram's horizontal axis we want to plot the value of the quantitative variable `estimate`.

- You should see an interesting plot here. We are facetting the histograms by the coefficients of the histogram (hence the term `facet_wrap(~term,scales="free")`, which says "plot a histogram for each variable in the column term." We use `scales="free"` because each coefficient has a different range of values. (You can see the difference if you remove `scales="free"` from the plotting command.)

Figure 11.4 is an example of a "small multiples" plot - the title of the plot is the value of the coefficient multiplying each term:

Small multiple title	**Coefficient of** $T = a + bY + cY^2$
`Intercept`	a
`yearSince1880`	b

Small multiple title	**Coefficient of** $T = a + bY + cY^2$
`I(yearSince1880^2)`	c

Notice that with the bootstrap we can get information about the distribution of the estimated parameters, which includes the median and the 95% confidence interval. This is super useful in reporting results from parameter estimates.

The idea of sampling with replacement, generating a parameter estimate, and then repeating over several iterations is at the heart of many computational parameter estimation methods. Such an approach helps make non-linear problems more tractable.

While we did a bootstrap parameter estimate on a quadratic function, this also works for a log transformation of a dataset, as with the `phosphorous` dataset of algae and daphnia we have studied previously. You will investigate this in one of the homework exercises.

This section extended your abilities in R by showing you how to generate histograms, sample a dataset, and compute statistics. The goal here is to give you examples that you can re-use in this section's exercises. Enjoy!

11.4 Exercises

Exercise 11.1. Histograms are an important visualization tool in descriptive statistics. Read the following essays on histograms, and then summarize 2-3 important points of what you learned reading these articles.

- http://tinlizzie.org/histograms/
- https://flowingdata.com/2017/06/07/how-histograms-work/
- https://flowingdata.com/2014/02/27/how-to-read-histograms-and-use-them-in-r/

Exercise 11.2. Average snow cover from 1970 - 1979 in October over Eurasia (in million km^2) were reported as the following:

$$\{6.5, 12.0, 14.9, 10.0, 10.7, 7.9, 21.9, 12.5, 14.5, 9.2\}$$

a. Create a histogram for these data.
b. Compute the sample mean and median of this dataset.
c. What would you report as a representative or typical value of snow cover for October? Why?
d. The 21.9 measurement looks like an outlier. What is the sample mean excluding that measurement?

Exercise 11.3. Consider the equation $S(\theta) = (3 - 1.5^{1/\theta})^2$. This function is an idealized example for the cost function in Figure 11.1.

a. What is $S'(\theta)$?
b. Make a plot of $S'(\theta)$. What are the locations of the critical points?
c. Algebraically solve $S'(\theta) = 0$. Does your compute critical point match up with the graph?

Exercise 11.4. Repeat the bootstrap sample for the precipitation dataset where the number of bootstrap samples is 1000 and 10000. Report the median and confidence intervals for the mean and the standard deviation of R. What do you notice as the number of bootstrap samples increases?

Exercise 11.5. Using the data in Exercise 11.2, do a bootstrap sample with $N = 1000$ to compute the a bootstrap estimate for the mean and the 95% confidence interval for October snowfall cover in Eurasia.

Exercise 11.6. We computed the 95% confidence interval using the `quantile` command. An alternative approach to summarize a distribution is with the `summary` command. Here is the output for the `summary` command for a dataframe:

knitr::include_graphics("figures/11-bootstrap/summary-output-11.png")

We call this command using `summary(data_frame)`, where `data_frame` is the particular dataframe you want to summarize. The output reports the minimum and maximum values of a dataset. The output `1st Qu.` and `3rd Qu.` are the 25th and 75th percentiles.

Do 1000 bootstrap samples using the data in Exercise 11.2 and report the output from the `summary` command.

Exercise 11.7. The dataset `snowfall` lists the snowfall data from a snowstorm that came through the Twin Cities on April 14, 2018.

a. Make an appropriately sized histogram for the snowfall observations.
b. What is the mean snowfall?
c. Do a bootstrap estimate with $N = 100$ and $N = 1000$ and plot their respective histograms.
d. For each of your bootstrap samples ($N = 100$ and $N = 1000$ and compute the mean and 95% confidence interval for the bootstrap distribution.
e. What would you report for the mean and 95% confidence interval for this snowstorm?

Exercise 11.8. This question tackles the dataset `global_temperature` to determine plausible models for a relationship between time and average global temperature. For this exercise we are going to look the variability in bootstrap estimates for models up to fourth degree.

Using the function `bootstrap_model`, generate a bootstrap sample of $n = 1000$ for each of the following functions.

- Linear: $T = a + bY$
- Quadratic: $T = a + bY + cY^2$
- Cubic: $T = a + bY + cY^2 + dY^3$
- Quartic: $T = a + bY + cY^2 + dY^3 + eY^4$

Your solution should include graphs of the data with the bootstrap predictions and the prediction from the linear regression model. How does the variability in the parameters change (a, b, c, d, e) as more terms in the model are added? How does the variability in the bootstrap predictions change as more terms in the model are added?

Exercise 11.9. Similar to the problems we have worked with before, the equation that relates a consumer's nutrient content (denoted as y) to the nutrient content of food (denoted as x) is given by: $y = cx^{1/\theta}$, where $\theta \geq 1$ and c are both constants is a constant. We will be using the dataset `phosphorous`.

a. Do 1000 bootstrap samples for this dataset.
b. To find c and θ we can apply logrithms to express this as a linear equation of this equation. (See Exercise @ref{exr:log-linear-08}). Do a linear model fit for this log-transformed equation.
c. Generate histograms for bootstrap-fitted parameters for your log-transformed equation.
d. What are the median and 95% confidence intervals for the bootstrap-fitted parameters?
e. Using the function `bootstrap_model`, generate a bootstrap sample of $n = 1000$ for the linear (log transformed) equation.
f. Translate these bootstrap confidence intervals of your fitted slope and intercept back into the values of c and θ.
g. These confidence intervals seem pretty large. What would be some strategies we could employ to narrow these confidence intervals?

Chapter 12

The Metropolis-Hastings Algorithm

In Sections 9 and 10 we discussed likelihood and cost functions, and Section 11 introduced the idea of sampling. For this section we will combine all these concepts to discover a powerful algorithm that can efficiently sample a probability distribution (think likelihood function) to estimate model parameters.

12.1 Estimating the growth of a dog

The problem we are considering is fitting the following data to a logistic model, adapted from here. We initially plotted these data in Section 2, but here is the code again.

```
wilson_data_plot <- ggplot(data = wilson) +
  geom_point(aes(x = days, y = mass)) +
  labs(
    x = "Days since birth",
    y = "Weight (pounds)"
  )

wilson_data_plot
```

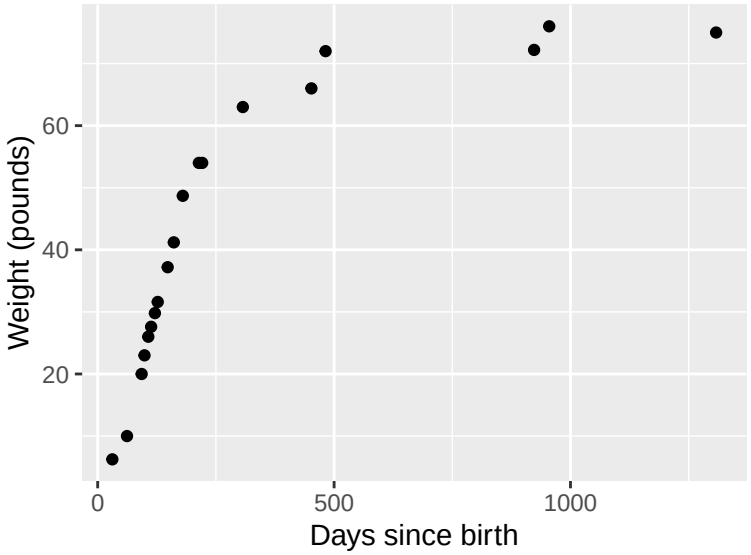


Figure 12.1: Weight of the dog Wilson over time.

Notice that for Figure 12.1 we stored the plot in the variable `wilson_data_plot`. We will be using this plot several times here, so storing it will help us save some typing.

The function we wish to fit from the data is the following

$$W = f(D, p_1) = \frac{p_1}{1 + e^{(p_2 - p_3 D)}}, \tag{12.1}$$

where we have the parameters p_1, p_2, and p_3. Notice how W is a function of D and p_1. For convenience we will set $p_2 = 2.461935$ and $p_3 = 0.017032$. Now we are only estimating the parameter p_1 which represents the maximum possible weight of the dog.

Let's take an initial guess for the parameter p_1. You may recognize that p_1 is the horizontal asymptote of this the function W. So at first glance let's set $p_1 = 78$. Let's plot that result along with the data:

```
days <- seq(0, 1500, by = 1)

p1 <- 78
p2 <- 2.461935
p3 <- 0.017032
mass <- p1 / (1 + exp(p2 - p3 * days))

my_guess <- tibble(days, mass)

my_guess_plot <- wilson_data_plot +
  geom_line(data = my_guess, color = "red", aes(x = days, y = mass))

my_guess_plot
```

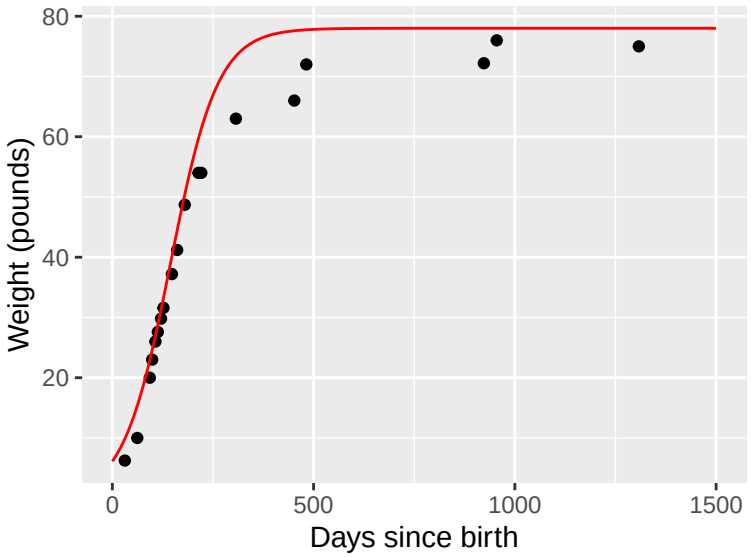


Figure 12.2: Weight of the dog Wilson with initial guess $p_1 = 78$.

As we did with Figure 12.1, we are going to store the updated plot as a variable. It seems that this value of p_1 does a good job capturing the initial rate of growth initially but perhaps predicts too high of a mass towards the end.

For comparison let's also examine the plot of the data when $p_1 = 65$:

```
days <- seq(0, 1500, by = 1)

p1 <- 65
p2 <- 2.461935
p3 <- 0.017032

mass <- p1 / (1 + exp(p2 - p3 * days))

my_guess_two <- tibble(days, mass)
```

```
my_guess_plot +
  geom_line(data = my_guess_two, color = "blue", aes(x = days, y = mass))
```

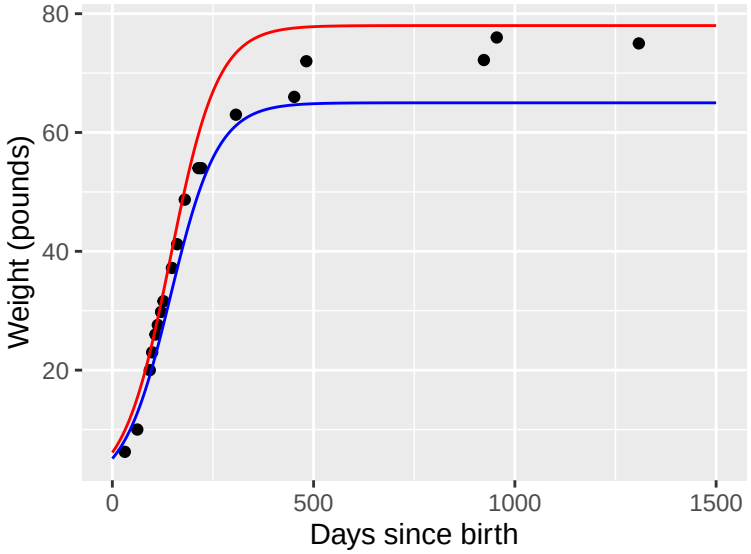


Figure 12.3: Weight of the dog Wilson with initial guess $p_1 = 78$ (red) and $p_1 = 65$ (blue).

12.2 Applying the likelihood to evaluate parameters

So with $p_1 = 65$ the modeled long-term trend seems to underestimate the data. Despite this, is the modeled growth with $p_1 = 65$ more plausible? We have nineteen measurements of the dog's weight over time. Assuming these measures are all independent and identically distributed, we have the following likelihood function:

$$L(p_1) = \prod_{i=1}^{19} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(W_i - f(D_i, p_1))^2}{2\sigma^2}} \tag{12.2}$$

We can easily compute the associated likelihood values with the function `compute_likelihood`:

```
# Define the model we are using
my_model <- mass ~ p1 / (1 + exp(p2 - p3 * days))


# This allows for all the possible combinations of parameters
parameters <- tibble(p1 = c(78, 65),
                     p2 = 2.461935,
                     p3 = 0.017032
                     )

# Compute the likelihood and return the likelihood from the list
out_values <- compute_likelihood(my_model, wilson, parameters)$likelihood

# Return the likelihood from the list:
out_values
```

```
## # A tibble: 2 x 5
##      p1    p2     p3   l_hood log_lik
##   <dbl> <dbl>  <dbl>    <dbl> <lgl>
## 1    78  2.46 0.0170 6.86e-29 FALSE
## 2    65  2.46 0.0170 1.20e-27 FALSE
```

Hopefully this code seems familiar to you from Section 9. We made some modifications to streamline things:

- We want to compare two values of `p1`, so when we defined `parameters` we included the two values of p_1 when defining `parameters`. The same values of `p2` and `p3` will apply to both. Don't believe me? Type `parameters` at the console line to see!
- Recall that when we apply `compute_likelihood` a list is returned (`likelihood` and `opt_value` . For this case we just want the `likelihood` data frame, hence the `$likelihood` at the end of `compute_likelihood`.

So we computed $L(78)$=$6.8560765 \times 10^{-29}$ and $L(65)$=$1.2038829 \times 10^{-27}$. As you can see the value of $L(65)$ is a larger number compared to $p_1 = 78$. One way we can get a sense for the magnitude of the scale is by examining the *ratio* of the likelihoods:

$$\text{Likelihood ratio: } \frac{L(p_1^{proposed})}{L(p_1^{current})}, \tag{12.3}$$

So let's compute this ratio: $\frac{L(65)}{L(78)}$ =rround(out_valuesl_hood[[2]]/out_valuesl_hood[[1]])‘. So with this comparison we would say that $p_1 = 65$ is more likely compared to the value of $p_1 = 78$.

Can we do better? What we can do is examine the likelihood ratio for another value. Since we might need to iterate through things let's define a function that we can reuse:

```
# Define a function that computes the likelihood ratio for Wilson's weight
likelihood_ratio_wilson <- function(proposed, current) {

  # Define the model we are using
  my_model <- mass ~ p1 / (1 + exp(p2 - p3 * days))


  # This allows for all the possible combinations of parameters
  parameters <- tibble(p1 = c(current, proposed), p2 = 2.461935, p3 = 0.017032)

  # Compute the likelihood and return the likelihood from the list
  out_values <- compute_likelihood(my_model, wilson, parameters)$likelihood

  # Return the likelihood from the list:
  ll_ratio <- out_values$l_hood[[2]] / out_values$l_hood[[1]]

  return(ll_ratio)
}

# Test the function out:
likelihood_ratio_wilson(65, 78)
```

```
## [1] 17.55936
```

You should notice that the reported likelihood ratio matches up with our computation - yes! So let's try to compute the likelihood ratio for $p_1 = 70$ compared to $p_1 = 65$. Try computing `likelihood_ratio_wilson(70,65)` - you should see that it is about 7.5 million times more likely!

I think we are onto something - Figure 12.4 compares the modeled values of Wilson's weight for the different parameters:

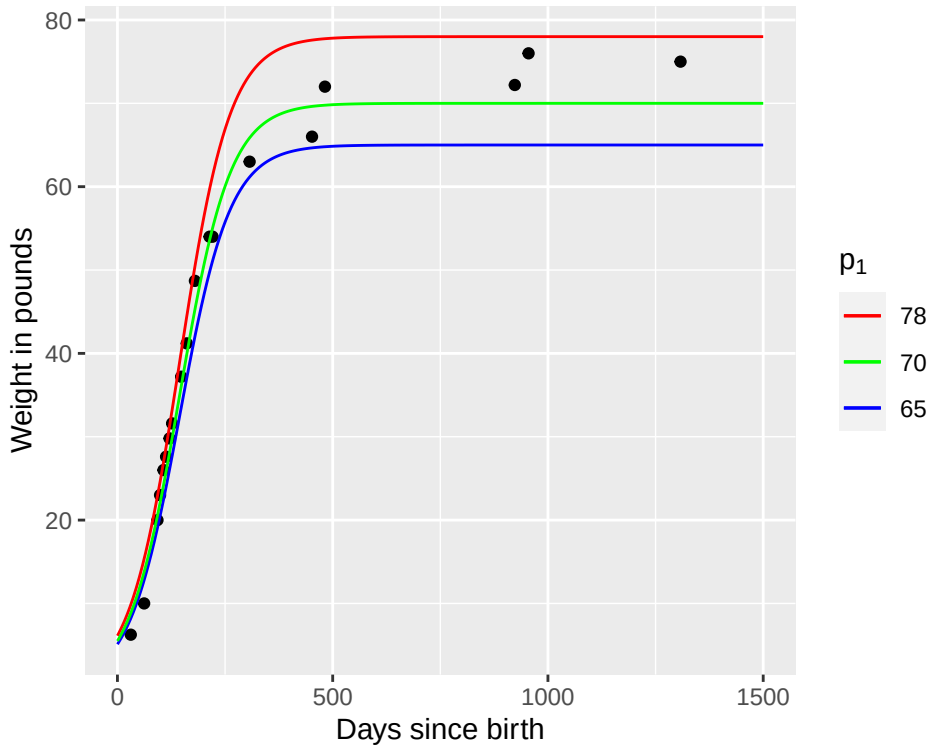


Figure 12.4: Comparison of our three estimates for Wilson's weight over time.

So now, let's try $p_1 = 74$ and compare the likelihoods: $\frac{L(74)}{L(70)}$=4.5897793 $\times 10^{-9}$. This seems to be *less* likely because the ratio was significantly less than one. If we are doing a hunt for the *best* optimum value, then we would reject $p_1 = 74$ and keep moving on, perhaps selecting another value closer to 70.

However, the reality is a little different. For non-linear problems we want to be extra careful that we do not accept a parameter value that leads us to a *local* (not global) optimum. A way to avoid this is compare the likelihood ratio to a uniform random number r between 0 and 1. We can do this very easily in `R` - remember the discussion of probability models in Section 9? At the `R` console type `runif(1)` - this creates one random number from the uniform distribution (remember the default range of the uniform distribution is $0 \leq p_1 \leq 1$). The `r` in `runif(1)` stands for *random*. When I tried `runif(1)` I received a value of 0.126. Since the likelihood ratio is smaller than the random number we generated, we will *reject* the value of p_1 and try again, keeping 70 as our value.

The process keep the proposed value based on some decision metric is called a *decision step*.

12.2.1 An iterative method to estimate parameters

The neat part of the previous discussion is that we can automate this process, such as in the following table:

Iteration	*Current value of* p_1	*Proposed value of* p_1	$\frac{L(p_1^{proposed})}{L(p_1^{current})}$	Value of `runif(1)`	*Accept proposed* p_1?
0	78	NA	NA	NA	NA
1	78	65	17.55936	NA	yes
2	65	70	7465075	NA	yes
3	70	74	0.09985308	0.126	no
4	70	...	...	...	...

This table is the essence of what is called the Metropolis-Hastings algorithm. Here are the following components:

1. A defined likelihood function.
2. A starting value for your parameter.

3. A proposal value for your parameter.
4. Comparison of the likelihood ratios for the proposed to the current value (Likelihood ratio: $\frac{L(p_1^{proposed})}{L(p_1^{current})}$). The goal of this algorithm method is to determine the parameter set that optimizes the likelihood function, or makes the likelihood ratio greater than unity. We will prefer values of p_1 that increase the likelihood.
5. A decision to accept the proposed parameter value. If the likelihood ratio is greater than 1, then we accept this value. However if the likelihood ratio is less than 1, we generate a random number r (using `runif(1)`) and use this following process:

- If r is less than the likelihood ratio we **accept** (keep) the proposed parameter value.
- If r is greater than the likelihood ratio we **reject** the proposed parameter value.

12.3 Concluding points

While we have done Metropolis-Hastings "by hand," you may realize that we can automate this process. We will explore that in the next section. However there are several modifications we can do to make this algorithm more sophisticated:

- While we have focused on implementation of the Metropolis-Hastings algorithm with one parameter, this is easily extended to sets of parameter values, but we just change one parameter at a time.
- We can select parameter values from multiple starting points to make sure we don't happen to start a chain near a local optimum. These starting points are called chains, and we can select the chain that has the highest likelihood value.
- We can use log likelihoods to make the likelihood ratio computation easier. By this way, instead of division we compute a difference of log likelihoods. This becomes numerically easier for R to handle, especially when the likelihood values are near zero. Here is some sample code that you can do this:

```
# Define a function that computes the LOG likelihood ratio for Wilson's weight
log_likelihood_ratio_wilson <- function(proposed, current) {

  # Define the model we are using
  my_model <- mass ~ p1 / (1 + exp(p2 - p3 * days))


  # This allows for all the possible combinations of parameters
  parameters <- tibble(p1 = c(current, proposed), p2 = 2.461935, p3 = 0.017032)

  # Compute the likelihood and return the likelihood from the list
  # Notice we've set logLikely = TRUE to compute the log likelihood
  out_values <- compute_likelihood(my_model, wilson, parameters, logLikely = TRUE)$likelihood

  # Return the likelihood from the list - here we compute the DIFFERENCE of likelihoods:
  ll_ratio <- out_values$l_hood[[2]] - out_values$l_hood[[1]]

  return(ll_ratio)
}
```

- As you would expect, the more times we iterate through this process, the better. Your initial guesses probably weren't that great (or close to the global optimum), so a common procedure is to throw out the first percentage of iterations and call that the "burn-in" period.
- We can systematically explore the parameter space, where the jump distance changes depending on if we are always accepting new parameters or not. This process has several different implementations, but one is called *simulated annealing*.

12.4 Exercises

Exercise 12.1. Using the dataset `wilson` as in this section, do 10 additional iterations of the Metropolis-Hastings Algorithm by continuing the table. See if you can get the value of p_1 to 2 decimal places of accuracy.

Exercise 12.2. An alternative model for the dog's mass is the following differential equation:

$$\frac{dW}{dt} = -k(W - p_1) \tag{12.4}$$

a. Apply separation of variables and $W(0) = 5$ and the value of p_1 from the previous problem to write down the solution to this equation. Your final answer will depend on k.
b. With the function `compute_likelihood` and the Metropolis-Hastings algorithm to estimate the value of k to three decimal places accuracy. The true value of k is between 0 and 1.

Exercise 12.3. Consider the linear model $y = 6.94 + bx$ for the following dataset:

x	y
-0.365	6.57
-0.14	6.78
-0.53	6.39
-0.035	6.96
0.272	7.20

With the function `compute_likelihood` apply 10 steps the Metropolis-Hastings algorithm to determine b.

Exercise 12.4. For the `wilson` dataset, repeat three steps of the parameter estimation to determine p_1 as in this section, but this time use the log-likelihood (so you will compare the *difference* of log likelihoods). Which method do you think is easier?

Chapter 13

Markov Chain Monte Carlo Parameter Estimation

This section applies elements of the Metropolis Algorithm to a sophisticated algorithm used for parameter estimation. This algorithm is called Markov Chain Monte Carlo (MCMC) parameter estimation. To learn more about the history of the MCMC algorithm see Richey (2010). contains While MCMC methods can be highly computational, you already have the skills in place already to understand how the MCMC method works - we will rely on functions from `demodelr` to do the heavy lifting.

The MCMC approach is a systematic exploration to determine the minimum value of the log likelihood function using the data and parameters for a model. In order to make this solver work method work, you will need four things:

- *Model*: A function that we have for our dynamics (this is $\frac{d\vec{y}}{dt} = f(\vec{y}, \vec{\alpha}, t)$), or an empirical equation $\vec{y} = f(\vec{x}, \vec{\alpha})$.
- *Data*: This can be a data frame or a spreadsheet file of the data you wish to use that is read into `R`.
- *Parameter bounds*: upper and lower bounds on your parameter values.
- *Initial conditions*: These may be optional and needed if you have a dynamic (differential equation) model.
- *Run diagnostics*: These are things that need to be specified in order to run the MCMC code, which may include how long you will run the code and aspects from the Metropolis Algorithm.

We will work you through this step by step, with example code that you can type along the way. As always it will be good to load up the libraries you will be using.

```
library(tidyverse)
library(demodelr)
```

Section 13 will examine two model the first example will be an empirical model (no differential equations), and the second will solve a differential equation.

13.1 MCMC Parameter Estimation with an Empirical Model

Here are going to return to the problem exploring the phosphorous content in algae (denoted by x) to the phosphorous content in daphnia (denoted by y).

The equation we are going to fit is:

$$y = c \cdot x^{1/\theta} \tag{13.1}$$

Equation (13.1) has two parameters c and θ, which range from $0 \leq c \leq 2$ and $1 \leq \theta \leq 20$, which we define below:

```
# Define the model
phos_model <- daphnia ~ c * algae^(1 / theta)

# Define the parameters that you will use with their bounds
phos_param <- tibble(
  name = c("c", "theta"),
```

```
  lower_bound = c(0, 1),
  upper_bound = c(2, 20)
)
```

The data that we use is the dataset `phosphorous`, which is already located in the `demodelr` package):

algae	daphnia
0.05376	1.08198
0.06506	1.13836
0.08217	1.26388
0.11031	1.43258
0.16061	1.45699
0.64414	1.51296

The final piece is to determine the number of iterations we run our the MCMC parameter estimate: That is it! All we need to do is to run our code:

```
# Define the number of iterations
phos_iter <- 1000

# Compute the mcmc estimate
phos_mcmc <- mcmc_estimate(
  model = phos_model,
  data = phosphorous,
  parameters = phos_param,
  iterations = phos_iter
)
```

The function `mcmc_estimate` may take some time (which is OK). This function has several inputs, which for convenience we write them on separate lines.

Let's take a look at the data frame `phosphorous_mcmc`:

```
glimpse(phos_mcmc)
```

```
## Rows: 1,000
## Columns: 4
## $ accept_flag <lgl> FALSE, FALSE, TRUE, FALSE, TRUE, TRUE, FALSE, FALSE, FALSE~
## $ l_hood      <dbl> 16.4535871, 16.4535871, -5.3947728, -5.3947728, -0.9328032~
## $ c           <dbl> 1.341311, 1.341311, 1.732444, 1.732444, 1.732444, 1.732444~
## $ theta       <dbl> 7.722542, 7.722542, 7.722542, 7.722542, 11.421999, 7.40589~
```

There are four columns here:

- `accept_flag` tells you if at that particular iteration the MCMC estimate was accepted or not.
- `l_hood` is the value of the likelihood for that given iteration.
- The values of the parameters follow on the next few lines. Notice that θ is written as `theta` in the resulting data frame.

Next we need to visualize our results. The function `mcmc_visualize` summarizes and visualizes the results filtering on whenever `accept_flag` is `TRUE` (which means the parameter was accepted).

```
# Visualize the results
mcmc_visualize(
  model = phos_model,
  data = phosphorous,
  mcmc_out = phos_mcmc
)
```

```
## [1] "The parameter values at the optimized log likelihood:"
## # A tibble: 1 x 3
##   l_hood     c theta
##    <dbl> <dbl> <dbl>
## 1  -5.56  1.65  8.98
## [1] "The 95% confidence intervals for the parameters:"
```

```
## # A tibble: 3 x 3
##   probs     c theta
##   <chr> <dbl> <dbl>
## 1 2.5%   1.51  7.06
## 2 50%    1.60 10.8
## 3 97.5%  1.75 18.1

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

`mcmc_visualize` will generate output to the console that contains parameter statistics and confidence intervals. Additionally `mcmc_visualize` will produce different types of graphs. Let's take a look at each one individually. The first plot is called a pairwise parameter plot. This contains a lot of different plots together, in a matrix pattern:

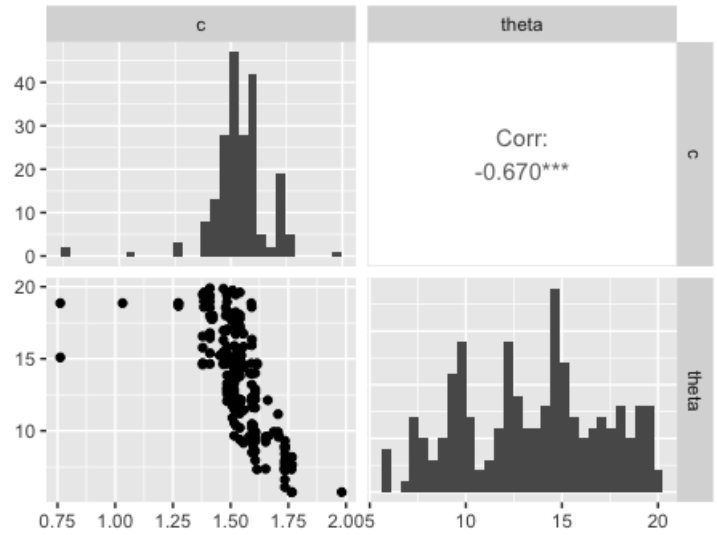


Figure 13.1: Pairwise parameter histogram from the MCMC parameter estimation for Equationeqrefeq:phos-13.

Along of the diagonal of Figure 13.1 is a histogram of the accepted parameter values from the Metropolis algorithm. Depending on the results that you obtain, you may have some interesting shaped histograms. Generally they are grouped in the following ways:

- *well-constrained:* the parameter takes on a definite, well-defined value. The parameter c seems to behave like this.
- *edge-hitting:* the parameter seems to cluster near the edges of its value.

- *non-informative:* the histogram looks like a uniform distribution. The parameter θ has some indications of being edge hitting, but we would need more iterations in order to determine this.

The off-diagonal terms in Figure 13.1 are interesting as well. The lower off-diagonal makes a scatter plot of the accepted values for the two parameters in the particular row and column, and the upper off-diagonal reports the correlation coefficient r of the variables in that particular row and column. The asterisks ($*$) denote the degree of significance of the linear correlation. Examining the pairwise parameter histogram helps ascertain the degree of *equifinality* in a particular set of variables (Beven and Freer 2001). In the Figure 13.1 it looks like c increases, θ decreases. This degree of linear coupling means that we may not be able to independent resolve each parameter separately. The presence of equifinality does not mean the parameter estimation is a failure - just that we need to be aware of these relationships. Perhaps we may be able to go out in the field and measure a parameter more (for example c) more carefully, narrowing the range of accepted values.

Figure 13.2 displays an *ensemble* estimate of the results with the data. The ensemble average plot provides a high-level model-data overview. The black line represents the median ensemble average, and the grey is the 95% confidence interval, giving you a perspective of the model spread with the data. For our results here it does look there is wide variation in the model, most likely due to the relative wide confidence intervals on our parameters.

Table 13.1: Scaled data on visitors and resources to a national park.

time	visitors	resources
0.0000000	0.0016667	0.9950
0.2833333	0.0046667	0.9860
0.3980000	0.0083333	0.9750
0.4513333	0.0150000	0.9550
0.4933333	0.0230000	0.9310
0.6553333	0.0238333	0.9285
0.6640000	0.0263330	0.9210
0.7280000	0.0321667	0.9035

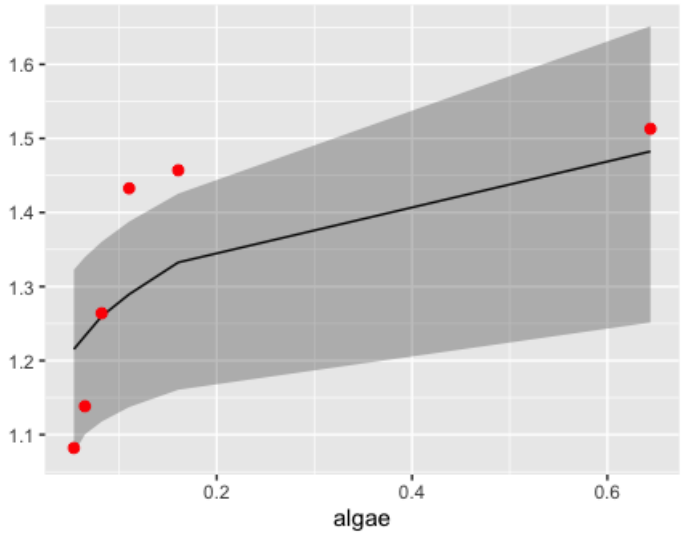


Figure 13.2: Ensemble output results from the MCMC parameter estimation for Equationeqrefeq:tourism-13.

13.2 MCMC Parameter Estimation with a Differential Equation Model

Next let's try parameter estimation with a differential equation model. What is different for this case is that we are given information about rate of change, so we need to first numerically solve the differential equation for a given parameter set and then compute the likelihood.

The example that we are going to use relates to land use management, in particular a coupled system between a resource (such as a national park) and the amount of visitors it receives (Sinay and Sinay 2006). The tourism model relies on two non-dimensonal scaled variables, R which is the amount of the resource (as a percentage) and V the percentage of visitors that could visit (also as a percentage):

$$\begin{aligned} \frac{dR}{dt} &= R \cdot (1 - R) - aV \\ \frac{dV}{dt} &= b \cdot V \cdot (R - V) \end{aligned} \tag{13.2}$$

Equation (13.2) has two parameters a and b, which relate to how the resource is used up as visitors come (a) and how as the visitors increase, word of mouth leads to a negative effect of it being too crowded (b).

For this case we are going to use a pre-defined dataset of the number of resources and visitors to a national park as reported in Sinay and Sinay (2006) (Table 13.1) and plotted in Figure 13.3

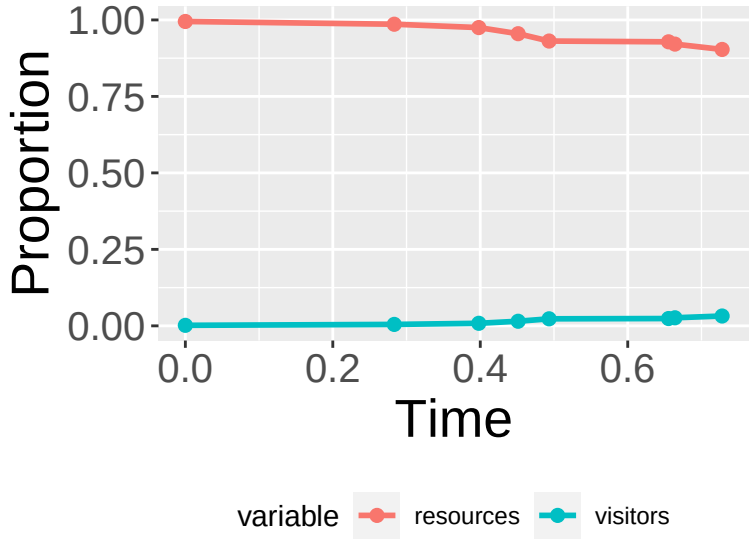


Figure 13.3: Scaled data on resources and visitors to a national park over time.

We can see that the data show as the visitors increase the percentage of the resources decrease. Perhaps from this limited dataset given we can estimate the parameters a and b. We are going to assume that $0 \leq a \leq 30$ and $0 \leq b \leq 5$. We will need to implement this model in our code, which combines our knowledge of how we numerically solved differential equations in Section 4.

```
# Define the tourism model
tourism_model <- c(
  dRdt ~ resources * (1 - resources) - a * visitors,
  dVdt ~ b * visitors * (resources - visitors)
)

# Define the parameters that you will use with their bounds
tourism_param <- tibble(
  name = c("a", "b"),
  lower_bound = c(10, 0),
  upper_bound = c(30, 5)
)

# Define the initial conditions
tourism_init <- c(resources = 0.995, visitors = 0.00167)

deltaT <- .1 # timestep length
n_steps <- 15 # must be a number greater than 1

# Define the number of iterations
tourism_iter <- 1000

tourism_out <- mcmc_estimate(
  model = tourism_model,
  data = parks,
  parameters = tourism_param,
  mode = "de",
  initial_condition = tourism_init,
  deltaT = deltaT,
  n_steps = n_steps,
  iterations = tourism_iter,
)
```

Notice how `mcmc_estimate` has some additional arguments. Most importantly is the option `mode`, which `de` stands for

differential equation. (The default mode is `emp`, or *empirical* model - like the `phosphorous` data set.) If the `de` mode is specified, then you also need to define defining the initial conditions (`tourism_init`), Δt (`deltaT`), and timesteps (`n_steps`).

Visualizing the data also is done with `mcmc_visualize`:

```
mcmc_visualize(
  model = tourism_model,
  data = parks,
  mcmc_out = tourism_out,
  mode = "de",
  initial_condition = tourism_init,
  deltaT = deltaT,
  n_steps = n_steps
)
```

Examining the parameter histograms (Figure 13.4) show b to be well-constrained. The histogram for a seems like it could be well-constrained - but we may need to run more iterations to confirm this.

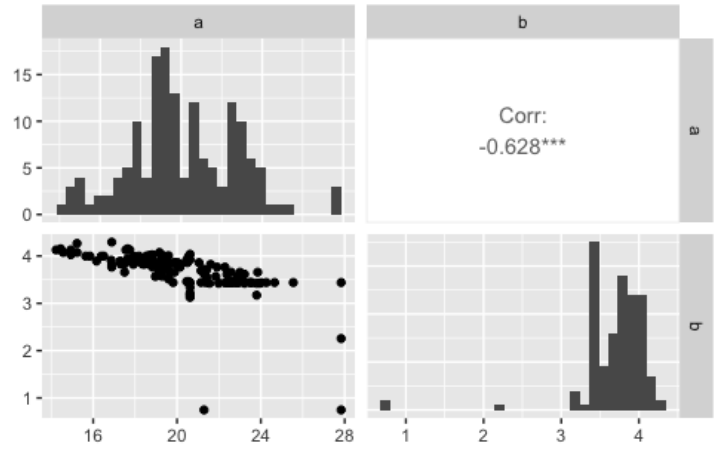


Figure 13.4: Pairwise parameter histogram of MCMC parameter estimation results for Equationeqrefeq:tourism-13

The model results and confidence intervals show good agreement to the data (Figure 13.5), also confirming that as visitors increase, the resources in the national park will decrease due to overuse.

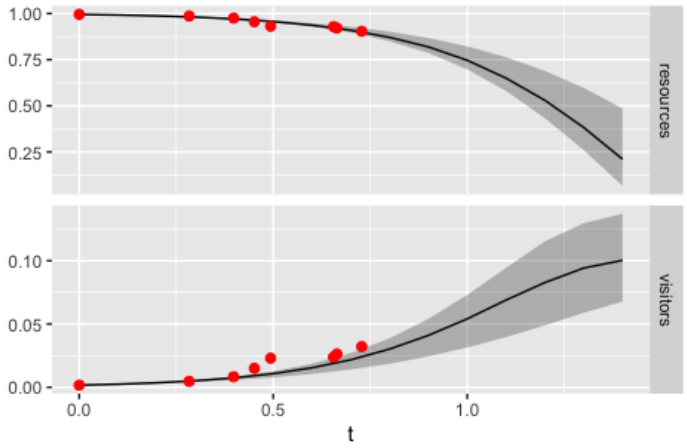


Figure 13.5: Ensemble output results from the MCMC parameter estimation for Equationeqrefeq:tourism-13.

13.3 Timing your code

As you can imagine the more iterations we have the better our parameter estimates will be. This comes with a tradeoff: it takes some time to run the full estimate. To get a sense of timing the code there is a helpful clock function in `R` that serves a stopwatch if you will. Let's check this out with one iteration of the phosphorous dataset:

```
# This "starts" the stopwatch
start_time <- Sys.time()

# Compute a single mcmc estimate
phosphorous1_mcmc <- mcmc_estimate(
  model = phos_model,
  data = phosphorous,
  parameters = phos_param,
  iterations = 1
)

# End the stopwatch
end_time <- Sys.time()

# Determine the difference between the start and end times
end_time - start_time
```

```
## Time difference of 0.07022405 secs
```

Timing the code for one iteration gives you a ballpark estimate for a full MCMC parameter estimate. If we were to run N MCMC iterations, a good benchmark would be to multiply the time difference (`end_time - start_time`) by N. Performance time varies by computer and the other programs / apps that are running at the same time. However, timing your code should give you an idea of what to expect.

13.4 Further extensions to MCMC

For the examples in this section we limited the number of iterations to a smaller number to make the results computationally feasible. However we can extend the MCMC approach a few different ways:

- One approach is to separate the data into two different sets - one for optimization and one for validation. In this approach the "optimization data'' consists a certain percentage of the original dataset, leaving the remaining to validate the forward forecasts. This is a type of cross-validation approach, and is generally preferred because you are demonstrating the strength of your model ability against non-optimized data.
- We also run multiple "chains" of optimization, starting from a different value in parameter space. What we do then after running each of these chains is to select the one with the best log-likelihood value, and run *another* MCMC iteration starting at that value. They idea is that we have sampled the parameter space and are hopefully starting near an optimum value.

As you can see, the MCMC algorithm is an extremely powerful technique for parameter estimation. While MCMC may take additional time and programming skill to analyze - it is definitely worth it! To learn more about the history of the MCMC algorithm and its other applications see Richey (2010).

13.5 Exercises

Exercise 13.1. For both of the MCMC examples in this section, increase the number of iterations to 10000. Analyze your results from both cases. How does increasing the number of iterations affect the posterior parameter estimates and their confidence intervals? Does the log likelihood value change?

Exercise 13.2. Time the MCMC parameter estimate for the `phosphorous` dataset for 1 iteration. Then time the MCMC parameter estimate for 10, 100, 1000, and 10000 iterations, recording the times for each one. Make a scatterplot with the number of iterations on the horizontal axis and time on the vertical axis. How would you characterize the relationship between the number of iterations and the time it takes to run the code?

Exercise 13.3. For the `parks` data (Equation (13.2)) studied in this section, compare the 1:1 and the posterior parameter plots. Summarize the following:

a. The posterior parameter estimates, with 95% confidence interval.
b. The posterior parameter histograms.

Apply your knowledge of equifinality and other observations to determine by how much you have estimated the parameters a and b fom the data.

Exercise 13.4. Run an MCMC parameter estimation on the dataset `yeast` from Gause (1932), where the equation for the volume of yeast V over time is given by the following equation for an yeast growing in isolation is:

$$V = \frac{K}{1 + e^{a-bt}}, \tag{13.3}$$

where K is the carrying capacity, a and b respective rate constants. Apply the data for Sacchromyces to do an MCMC estimate for this equation. You may assume the following prior values on your parameters:

- K : 0 to 20
- b: 0 to 1
- a: automatically equals to $a = \ln(K/0.45 - 1)$

Be sure to report all outputs from the MCMC estimation (this includes parameter estimates, confidence intervals, log likelihood values, and any graphs).

Exercise 13.5. Another model for this growth of yeast is the function $V = K + Ae^{-bt}$. Compute an MCMC estimate for the parameters K and b (use the same bounds as in the previous problem). You may assume that $V(0) = 0.45$, so $A = K - 0.45$. Be sure to report all outputs from the MCMC estimation (this includes parameter estimates, confidence intervals, log likelihood values, and any graphs). Compare your results to the previous exercise.

Exercise 13.6. Run an MCMC parameter estimation on the dataset `wilson` according to the following differential equation:

$$\frac{dP}{dt} = b(N - P) \tag{13.4}$$

- K : 60 to 90
- b: 0 to 1

Be sure to report all outputs from the MCMC estimation (this includes parameter estimates, confidence intervals, log likelihood values, and any graphs).

Chapter 14

Information Criteria

Sometimes we have not just one model but several different models to describe a context or situation. While having these different options are good, but we also like to know which is the *best* model. How would you decide that?

This first step is to develop some guidelines. Here would be a list of they types of things that would go into evaluating which model would be best:

- The model complexity - how many equations do we have?
- The number of parameters - a few or many?
- Do the model outputs match the data?
- (For timeseries data) are the trends accurately represented?
- Is the model easy to use?
- How will model outputs compare the newly collected measurements?

These questions are related to one another - and answering these questions (or ranking criteria for them) is at the heart of the topic of *model selection* which is the study of deciding the **best approximating model**. Let's start to dig into this topic

14.1 Why bother with more models?

Recall the key problem of *parameter estimation*, which can be generally stated as the following:

> Determine the set of parameters $\vec{\alpha}$ that minimize the difference between data $\vec{y}$ and the output of the function $f(\vec{x}, \vec{\alpha})$ and measured error $\vec{\sigma}$.

Let's say we have not just one model $f(\vec{x}, \vec{\alpha})$, but an alternative model for the data $\vec{y}$ that can be represented as $g(\vec{x}, \vec{\beta})$. Between these two models (f and g), how would we determine which one is "best?"

Let's talk about a specific example. Let's focus on the dataset `global_temperature` in the `demodelr` library. This dataset represents the average global temperature determined by NASA using local temperature data. When we did linear regression on the global temperature dataset the quadratic and cubic models were approximately the same (Figure 14.1):

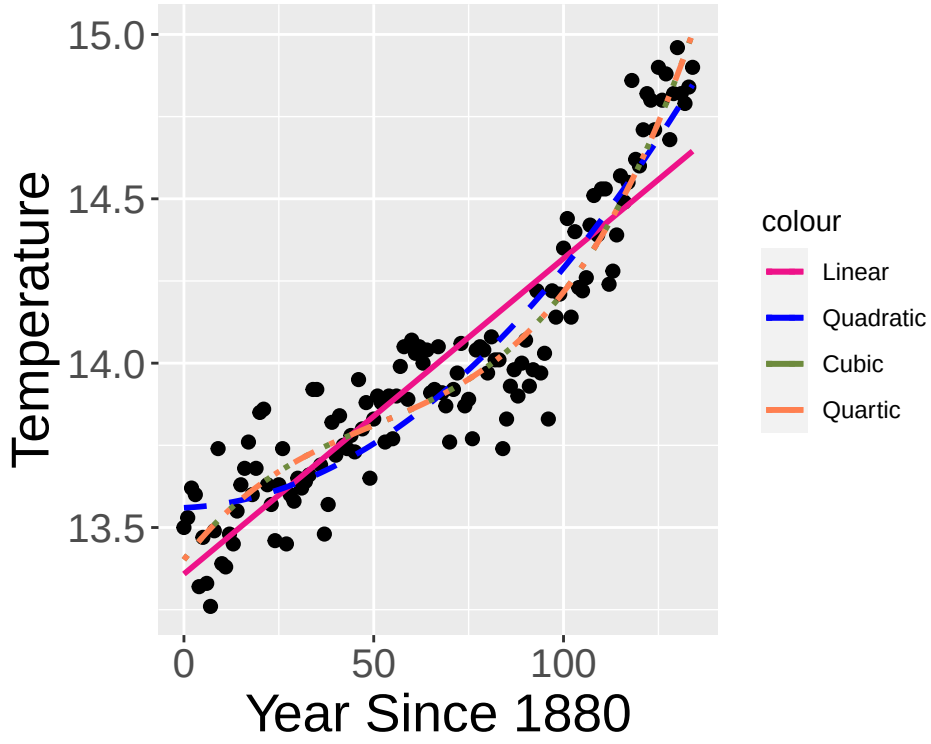


Figure 14.1: Comparison of global temperature data with various polynomial fitted models.

The variation in the different models fits for Figure 14.1 shows how different, but similar, the model results can be depending on the choice of regression function. In some cases, the Log likelihood decreases (indicating a more likely model), with a smaller root mean square error (RMSE, indicating the fitted model more closely matches the observations).

Model	Log Likelihood	RMSE
Linear	49.917	0.167
Quadratic	74.385	0.139
Cubic	89.49	0.125
Quartic	89.599	0.125

Further model evaluation can be examined by the following:

- Compare the measured values of $\vec{y}$ to the modeled values of $\vec{y}$ in a 1:1 plot. Does g does a better job predicting $\vec{y}$ than f?
- Related to that, compare the likelihood function values of f and g. We would favor the model that has the lower log likelihood.
- Compare the number of parameters in each model f and g. We would favor the model that has the fewest number of parameters.

14.2 The Information on Information Criterion

Building off the previous work, one approach to model assessment examines the number of parameters in the model f and g and evaluates the tradeoff between model complexity (i.e. the number of parameters used) and the overall likelihood. *Information criteria* are used to assess the tradeoffs between model complexity and the number of parameters. The goal of information criteria is to determine the *best approximating model*.

There are several types of information criteria, but we are going to focus on two::

- The *Akaike Information Criteria* (AIC, Akaike (1974)) is a commonly used information criteria:

$$AIC = -2LL_{max} + 2P \tag{14.1}$$

- An alternative to the AIC is the **Bayesian Information Criterion** (BIC, Schwartz (1978))

$$BIC = -2LL_{max} + P\ln(N), \tag{14.2}$$

In Equations (14.1) and (14.2) N is the number of data points, P is the number of estimated parameters, and LL_{max} is the log-likelihood for the parameter set that maximized the likelihood function. In both cases, a lower value of the information criteria indicates greater support for the model from the data. For both Equations (14.1) and (14.2) show the dependence on the log likelihood function and the number of parameters.

Let's evaluate how the AIC and BIC compare for the global temperature data. When we have a statistical model fit, computing these are fairly easy to compute.

For you `R` purists, you could also use the function `AIC` or `BIC`. To apply them you need to first do the model fit (with the function `lm`[1]:

```
regression_formula <- globalTemp ~ 1 + yearSince1880
fit <- lm(regression_formula,data=global_temperature)
AIC(fit)
```

```
## [1] -93.83421
```

```
BIC(fit)
```

```
## [1] -85.11838
```

We can then make a table comparing the different models and their AIC:

Model	AIC
Linear	-93.834
Quadratic	-140.769
Cubic	-168.98
Quartic	-167.198

These results show that the cubic model is the better approximating model.

14.3 A few cautionary notes

- Information criteria a relative measures. In a study it may be more helpful to report the change in the information criteria, or even a ratio (see Burnham and Anderson (2002) for a detailed analysis).
- Information criteria are not cross-comparable across studies. If you are pulling in a model from another study, it is helpful to re-calculate the information criteria.
- An advantage to the BIC is that it measures tradeoffs between favoring a model that has the fewer number of data needed to estimate parameters. Other information criteria examine the distribution of the likelihood function and parameters.

The upshot: Information criteria are *one* piece of evidence to help you to evaluate the best approximating model. You should do additional investigation (parameter evaluation, model-data fits, forecast values) in order to help determine the best model.

[1] You can compute the log likelihood with the function `logLik(fit)`, where `fit` is the result of your linear model fits.

14.4 Exercises

Exercise 14.1. Use the dataset `global_temperature` and for a linear, quadratic, cubic, and quartic polynomial and compute the BIC. What is the best approximating polynomial for the BIC? How does that compare to the best approximating model with the AIC?

&nsbp;

Exercise 14.2. You are investigating different models for the growth of a yeast species in population where V is the rate of reaction and s is the added substrate:

$$\text{Model 1: } V = \frac{V_{max}s}{s + K_m}$$
$$\text{Model 2: } V = \frac{K}{1 + e^{-a-bs}}$$
$$\text{Model 3: } V = K + Ae^{-bs}$$

With a dataset of 7 observations you found that the log-likelihood for Model 1 `26.426`, for Model 2 the log-likelihood is `15.587`, and for Model 3 the the log-likelihood is `21.537`. Apply the AIC and the BIC to evaluate which model is the best approximating model.

Exercise 14.3. An equation that relates a consumer's nutrient content (denoted as y) to the nutrient content of food (denoted as x) is given by: $y = cx^{1/\theta}$, where $\theta \geq 1$ and c are both constants is a constant. We can apply linear regression to the dataset $(x,\ \ln(y))$, so the intercept of the linear regression equals $\ln(c)$ and the slope equals $1/\theta$.

a. Show that you can write this equation as linear equation by applying a logarithm to both sides and simplifying.
b. With the dataset `phosphorous`, take the logarithim the `daphnia` variable and then to determine a linear regression fit for your new linear equation. What are the reported values of the slope and intercept from the linear regression, and by association, c and θ?
c. Use the function `logLik` to report the likelihood of the fit.
d. What are the reported values of the AIC and the BIC?
e. An alternative linear model is the equation $y = a+b\sqrt{x}$. Use the R command `model2 <- lm(daphnia~I(sqrt(algae),data = phosphorous))` to first obtain a linear fit. Then compute the log likelihood and the AIC and the BIC. Of the two models, which one is the better approximating model?

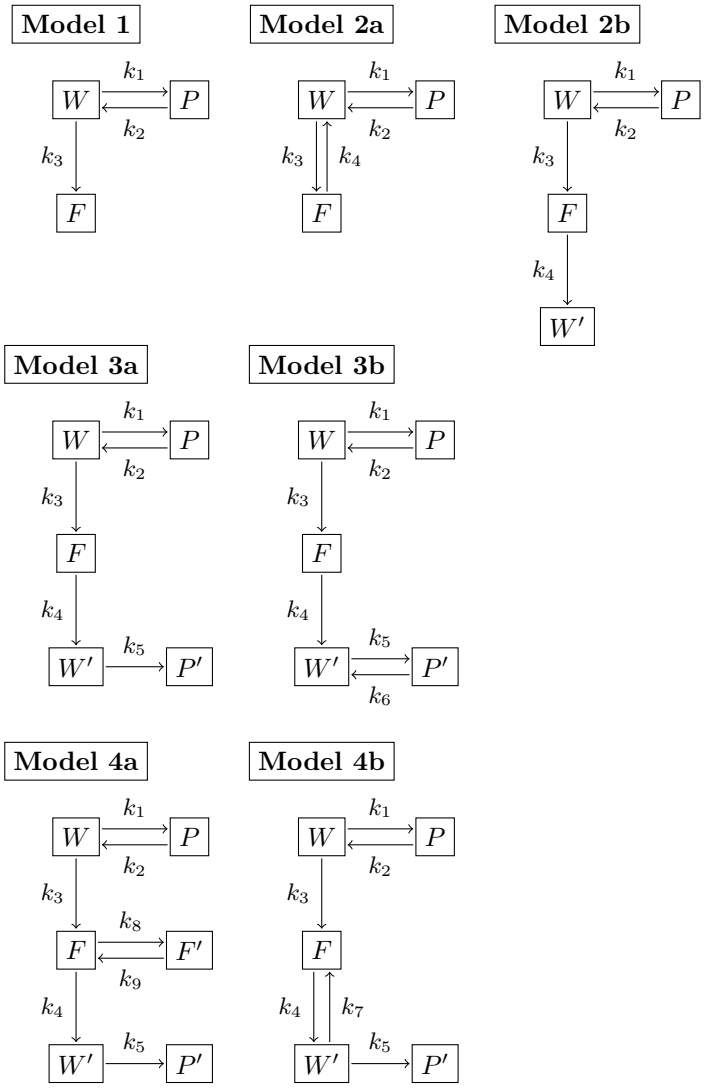


Figure 14.2: Reaction schemes.

Exercise 14.4. (Inspired by Burnham and Anderson (2002)) You are tasked with the job of investigating the effect of a pesticide on water quality, in terms of its effects on the health of the plants and fish in the ecosystem. Different models can be created that investigate the effect of the pesticide. Different types of reaction schemes for this system are shown in Figure 14.2, where F represents the amount of pesticide in the fish, W the amount of pesticide in the water, and S the amount of pesticide in the soil. The prime (e.g. F', W', and S' represent other bound forms of the respective state). In all seven different models can be derived.

These models were applied to a dataset with 36 measurements of the water, fish, and plants. The table for the log-likelihood for each model is shown below:

Model	Log Likelihood
1	-90.105
2a	-71.986
2b	-56.869
3a	-31.598
3b	-31.563
4a	-8.770
4b	-14.238

a. Use Figure 14.2 to identify the number of parameters for each model.
b. Apply the AIC and the BIC to the data in the above table to determine which is the best approximating model.

Part III

Stability Analysis for Differential Equations

Chapter 15

Systems of linear equations

So far we have looked at understanding differential equations qualitatively and estimating model parameters with data. Now we are going to delve into a deeper understanding of differential equations by examining long term stability of equilibrium solutions. As a first step this section focuses on *linear* systems of differential equations.

For example consider this following system of equations:

$$\begin{aligned} \frac{dx}{dt} &= 2x \\ \frac{dy}{dt} &= x + y \end{aligned} \tag{15.1}$$

This set of differential equations is linear because it does not have any nonlinear (squared, power terms) or transendental functions (sine, cosine, exponential) of the variables x and y. This system is an example of a *coupled* system of equations, mainly due to the expression $\frac{dy}{dt} = x + y$.

One way to write this is with matrix notation, where we use the prime notation (x' or y') to signify $\frac{dx}{dt}$ or $\frac{dy}{dt}$:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 2x \\ x + y \end{pmatrix} \tag{15.2}$$

$$= \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{15.3}$$

We can also represent the differential equation in a compact vector notation: $\frac{d\vec{x}}{dt} = A\vec{x}$, where for this example $\vec{A} = \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix}$.

Linear equations have some *very* interesting properties - which I hope you will explore more by taking linear algebra! But let's generalize a little bit now. If we have a system of linear equations

$$\begin{aligned} \frac{dx}{dt} &= ax + by \\ \frac{dy}{dt} &= cx + dy, \end{aligned} \tag{15.4}$$

you can write it in the following manner:

$$\begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{15.5}$$

15.1 Equilibrium solutions

Now that we have our systems of linear equations, let's understand the dynamics. A first question is to examine the equilibrium solutions, or places where both $\frac{dx}{dt} = 0$ and $\frac{dy}{dt} = 0$. It might be helpful to imagine what we should *expect* for an equilibrium solution. Think back to calculus - what types of functions have a zero derivative? (Hopefully constant functions comes to mind!)

If constant functions are a type of equilibrium solution, does that mean *any* constant function is an equilibrium solution? Let's try this out with our example:

$$\begin{aligned} \frac{dx}{dt} &= 2x \\ \frac{dy}{dt} &= x + y \end{aligned} \tag{15.6}$$

If we evaluate our differential equation at $x = 3$ and $y = 5$ (which are both constant solutions), we have the following:

$$\begin{aligned} \frac{dx}{dt} &= 2 \cdot 3 = 6 \\ \frac{dy}{dt} &= 3 + 5 = 8 \end{aligned} \tag{15.7}$$

So looking at our results the right hand sides of the differential equation do not evaluate to zero. Does that mean our intuition is wrong? Not necessarily - maybe we just didn't pick the correct solution. Note in the second equation we have $\frac{dy}{dt} = x + y$ - what if we picked $x = 0$ and $y = -5$? That makes $\frac{dx}{dt} = 0$, but $\frac{dy}{dt} = -5$. Oh no - that is not an equilibrium solution either!

However, a solution that could work is if both $x = 0$ and $y = 0$ (Try to do this on your own.) Here is an amazing fact: it turns out **any linear system has the origin as its only equilibrium solution.**. I won't discuss this fact more, but one way to explain this is by examining solutions for linear systems of equations in linear algebra.

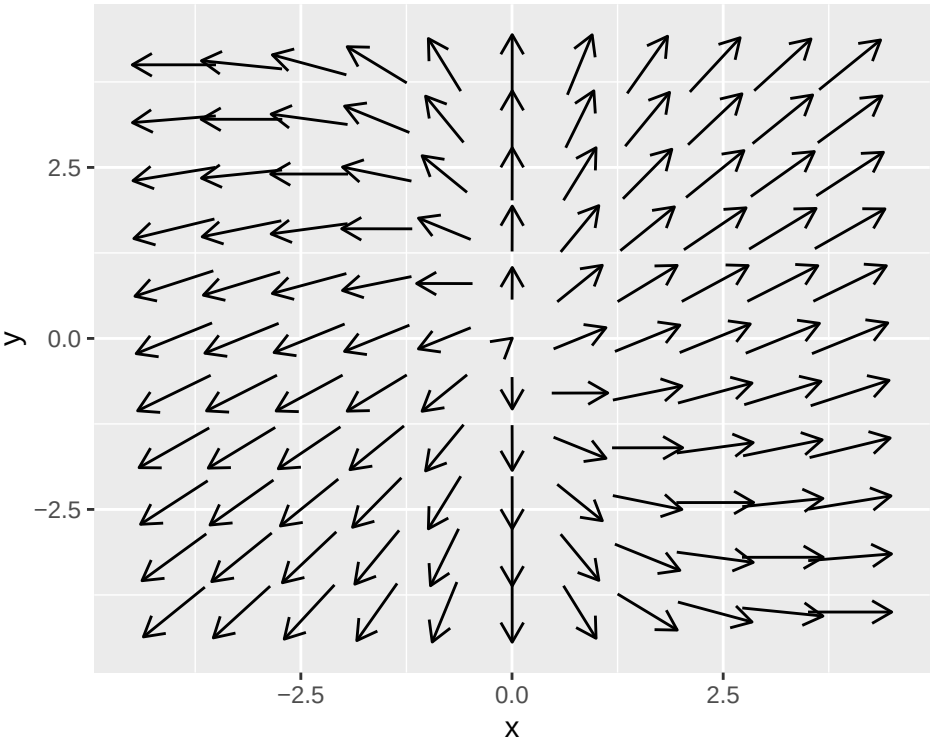
You might be wondering why all the fuss with equilibrium solutions - especially the origin ($x = 0$ and $y = 0$)[1]. So while equilibrium solutions are not terribly interesting question at the moment, the *stability* of solutions are. In order to understand what I mean by stability let's re-examine phase planes from Section 6

15.2 The phase plane

The phase plane is helpful here to understand the nature of the equilibrium solution. Remember that the phase plane shows the motion of solutions, visualized as a vector. For the system we examined earlier let's take a look at the phase plane. Here is some `R` code from the `demodelr` package to help you visualize a phaseplane:

```
# For a two variable systems of differential equations we need to define dx/dt and dy/dt separately:


systems_eq <- c(
  dxdt ~ 2 * x,
  dydt ~ x + y
)

# Now we plot the solution.
phaseplane(systems_eq, "x", "y")
```

[1] Another name for the origin equilibrium solution is the *trivial equilibrium*.



Let's talk a little bit about each of the commands first.

- You need to define both derivatives as separate functions (the lines that contain `dxdt <- ...` and `dydt <- ...`). We collect this in a vector called `systems_eq`, with the tilde (~) replacing the equals (=) sign.
- The command `phaseplane` has required inputs of $\frac{dx}{dt}$ and $\frac{dy}{dt}$ and the names of the variables. There are additional inputs defining (1) the x and y windows (the default is $-4 \leq x \leq 4$ and $-4 \leq y \leq 4$), and (2) the number of arrows in each row and column (default is 10). For example, the command `phaseplane(systems_eq,'x','y',x_window=c(-5,5),y_window=c(-5,5),plot_points=20)` will make a phaseplane diagram for the differential equation using 20 arrows with the axes limits to be between -5 and 5, and axis labels x and y.

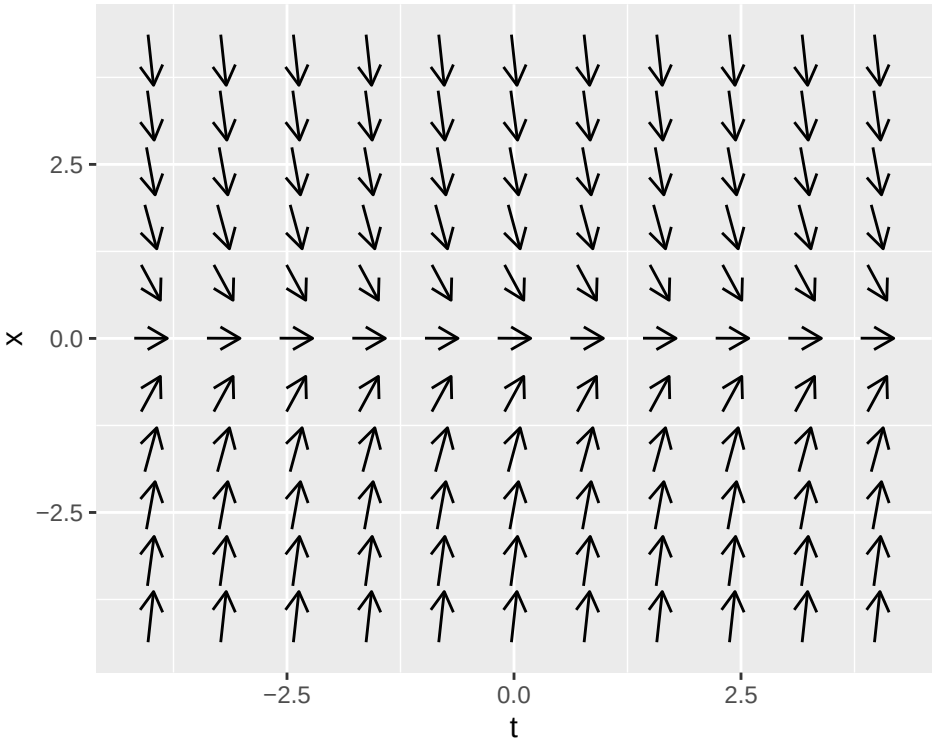
Notice how in the phaseline the arrows seem to spin out from the origin. We are going to discuss that below - but based on what we see we might expect the stability of the origin to be *unstable*.

15.2.1 What about one dimensional equations?

Remember how earlier on we had talked about equilibrium solutions with a *phase line*? It turns out we can use the command `phaseplane` to visualize a phase line. The subtlety is that we need to rewrite the one dimensional equation as a system of equations.

Let's take the example $\frac{dx}{dt} = -3x$

```
# For a two variable systems of differential equations we need to define dx/dt and dy/dt separately:


systems_eq <- c(
  dt ~ 1,
  dx ~ -3 * x
)

# Now we plot the solution.
phaseplane(systems_eq, "t", "x")
```



This codes looks similar to the previous example, with one small modification. For the variable `systems_eq` we need `dt ~ 1` - which basically uses the horizontal axis as the time dependent variable - this basically says that the rate of change on the horizontal axis is constant (which yields the solution t). You will work out the details for converting a one-dimensional equation into a system of equations in Exercise 15.5.

15.3 Stability of solutions

Because we have already identified the equilibrium solution for Equation (15.1), the focus is to understand if the equilibrium solution is stable or unstable. There are two other (non-equilibrium) solutions to this system of differential equations:

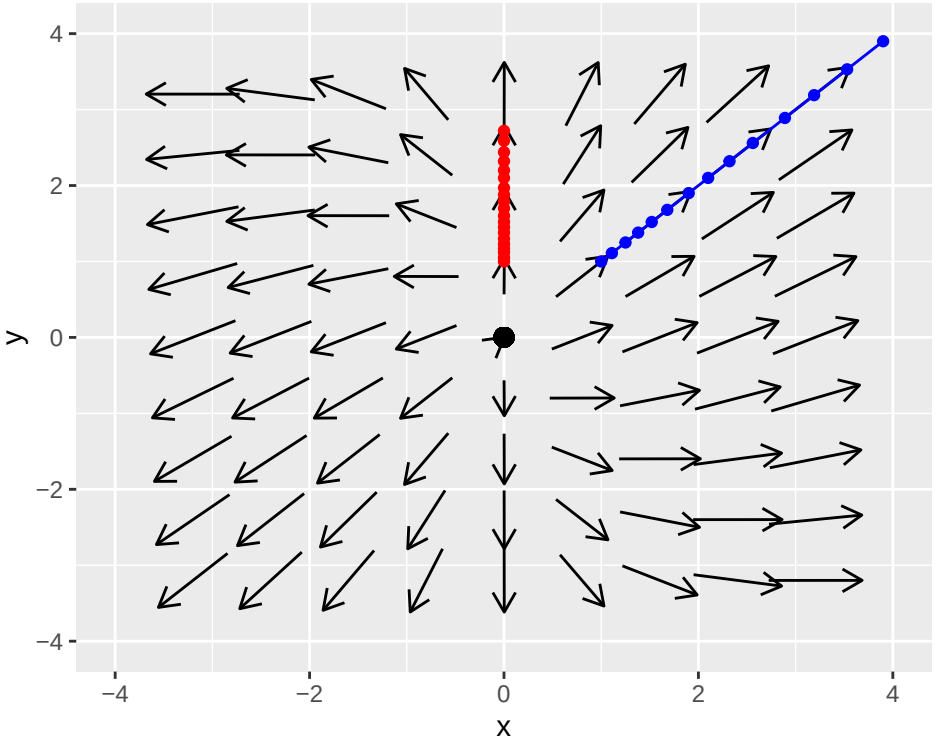
- **Solution 1:** $x = 0$ and $y = e^t$
- **Solution 2:** $x = e^{2t}$ and $y = e^{2t}$

We can make a table of the solution for different values of t:

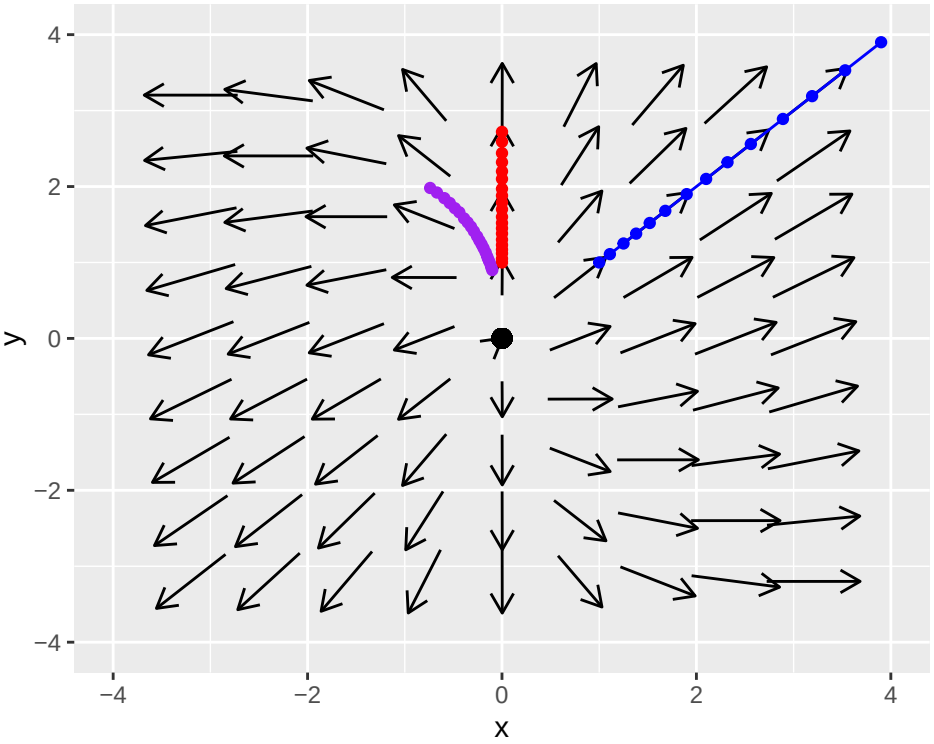
We can incorporate the information from the table with the phase plane:

Table 15.1: Solution values

	Solution 1		Solution 2	
t	x1	y1	x2	y2
0.00	0	1.00	1.00	1.00
0.05	0	1.05	1.11	1.11
0.11	0	1.12	1.25	1.25
0.16	0	1.17	1.38	1.38
0.21	0	1.23	1.52	1.52
0.26	0	1.30	1.68	1.68
0.32	0	1.38	1.90	1.90
0.37	0	1.45	2.10	2.10
0.42	0	1.52	2.32	2.32
0.47	0	1.60	2.56	2.56
0.53	0	1.70	2.89	2.89
0.58	0	1.79	3.19	3.19
0.63	0	1.88	3.53	3.53
0.68	0	1.97	3.90	3.90
0.74	0	2.10	4.39	4.39
0.79	0	2.20	4.85	4.85
0.84	0	2.32	5.37	5.37
0.89	0	2.44	5.93	5.93
0.95	0	2.59	6.69	6.69
1.00	0	2.72	7.39	7.39



Do you notice the motion in the xy plane of the two solutions? They are straight lines! It turns out that these straight lines solutions are quite useful - we will study them in a later section. But for the moment we will apply the idea of linear combinations to plot another solution, where $\vec{z}(t) = \vec{x}_1(t) - 0.1\vec{x}_2(t)$:



The phase plane suggests that the equilibrium solution at the origin is unstable because *both* the arrows and the solution seem to be pointing away from the origin. We can also investigate stability *algebraically* for each solution ($s_1(t)$ and $s_2(t)$). We will organize our solutions in vector format, factoring out the exponential functions in each of the expressions:

Let's look at this more systematically:

- Solution 1: $\vec{s}_1(t) = \begin{pmatrix} 0 \\ e^t \end{pmatrix} = \begin{pmatrix} 0 \\ e^t \end{pmatrix} = e^t \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
- Solution 2: $\vec{s}_2(t) = \begin{pmatrix} e^{2t} \\ e^{2t} \end{pmatrix} = \begin{pmatrix} e^{2t} \\ e^{2t} \end{pmatrix} = e^{2t} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

By factoring out the exponential functions we can see the straight line solutions! The vectors $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$ and $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$ are the lines $x = 0$ and $y = x$, as shown in our phase plane diagrams with the red and blue lines respectively.

To investigate stability we investigate the long term behavior of the exponential functions. Notice that $\lim_{t\to\infty} e^t$ and $\lim_{t\to\infty} e^{2t}$ *both* do not have a finite value, so we classify the equilibrium solution as "unstable."

So to recap the following about straight line solutions to two-dimensional linear systems:

- Straight line solutions have the form $\vec{s}(t) = e^{\lambda t}\vec{v}$. Methods to determine λ and $\vec{v}$ will be studied in later sections.
- For a 2 dimensional linear system, you have generally will two straight line solutions $\vec{s}_1$ and $\vec{s}_2$. This means you will have two different values of λ (λ_1 and λ_2). The general "solution" to the system of differential equations is the linear sum of the two: $\vec{x}(t) = c_1\vec{s}_1(t) + c_2\vec{s}_2(t)$.
- Geometrically these straight line solutions are lines that passing through the origin in the xy plane.
- If both values of λ are *greater* than 0, equilibrium solution is *unstable*.
- If both values of λ are *less* than 0, equilibrium solution is *stable*.

In the exercises you will look at additional examples to understand the behavior of linear systems.

15.4 Exercises

Exercise 15.1. Write the following systems of equations in matrix notation ($\frac{d\vec{x}}{dt} = A\vec{x}$):

a.

$$\begin{aligned} \frac{dx}{dt} &= 3x - 4y \\ \frac{dy}{dt} &= 2x - y, \end{aligned} \tag{15.8}$$

b.

$$\begin{aligned} \frac{dx}{dt} &= x + y \\ \frac{dy}{dt} &= y - x, \end{aligned} \tag{15.9}$$

c.

$$\begin{aligned} \frac{dx}{dt} &= 5x - 4y + z \\ \frac{dy}{dt} &= y - 9z, \\ \frac{dz}{dt} &= 7x - z, \end{aligned} \tag{15.10}$$

d.

$$\begin{aligned} \frac{dx}{dt} &= -cx \\ \frac{dy}{dt} &= rcx - y, \end{aligned} \tag{15.11}$$

Exercise 15.2. Verify that $x = 0$ and $y = e^t$ and $x = e^{2t}$ and $y = e^{2t}$ are solutions for the differential equation

$$\begin{aligned} \frac{dx}{dt} &= 2x \\ \frac{dy}{dt} &= x + y \end{aligned} \tag{15.12}$$

Exercise 15.3. Verify that $x = 0$ and $y = 0$ are solutions to the differential equation

$$\begin{aligned} \frac{dx}{dt} &= -3x \\ \frac{dy}{dt} &= .2x - y, \end{aligned} \tag{15.13}$$

Exercise 15.4. Verify that $x = 0$, $y = 0$, and $z = 0$ are solutions to the differential equation

$$\begin{aligned} \frac{dx}{dt} &= 5x - 4y + z \\ \frac{dy}{dt} &= y - 9z, \\ \frac{dz}{dt} &= 7x - z, \end{aligned} \tag{15.14}$$

Exercise 15.5. Consider the differential equation $\frac{dx}{dt} = -3x$. This exercise will help you work through the details of creating a two dimensional system of equations by re-parameterizing $s = t$.

a. Define the variable $t = s$. For this case, what is $\dfrac{dt}{ds}$?

b. Explain if $ x = f (t (s))$ (x is a composition between t and s), explain why the chain rule has $\dfrac{dx}{ds} = \dfrac{dx}{dt} \cdot \dfrac{dt}{ds}$.

c. Use the fact that $\dfrac{dx}{ds} = \dfrac{dx}{dt} \cdot \dfrac{dt}{ds}$ to explain why $\dfrac{dx}{ds} = -3x$.

d. Finally use your previous work to determine the system of equations for $\dfrac{dx}{ds}$ and $\dfrac{dt}{ds}$.

Exercise 15.6. This problem considers the differential equation

$$\begin{aligned} \frac{dx}{dt} &= x + y \\ \frac{dy}{dt} &= y - x, \end{aligned} \tag{15.15}$$

a. Use the command `phaseplane` to create a phaseplane of this differential equation.

b. Change the number of arrows shown to 5 and 20 (2 different plots). What do you notice about the updated phaseplane?

c. Change the viewing window from the default to -10 to 10 in both axes. Now change the number of arrows shown to 5 and 20 (2 different plots). What do you notice about the updated phaseplane?

Exercise 15.7. Explain why we call $x = 0$ and $y = 0$ *equilibrium* solutions to the general linear differential equation $\dfrac{d\vec{x}}{dt} = A\vec{x}$. In other words, why is the word *equilibrium* important? (Hint: Think about the graph of x and y for the solution in this case.)

Exercise 15.8. (Inspired by logan_mathematical_2009) Consider the following differential equation:

$$\begin{aligned} \frac{dx}{dt} &= -ax - y \\ \frac{dy}{dt} &= x - ay \end{aligned} \tag{15.16}$$

a. Write this system in the form $\dfrac{d\vec{x}}{dt} = A\vec{x}$.

b. Let $a = -2,\ -1,\ -0.5,\ 0,\ 0.5,\ 1,\ 2$. Generate a phase plane for each of these values of a and characterize the behavior of the equilibrium solution.

Exercise 15.9. Generate a phaseplane for the following differential equations and using your result, classify if the equilibrium solution is stable or unstable.

a. $\dfrac{dx}{dt} = -x,\ \dfrac{dy}{dt} = -2y$

b. $\dfrac{dx}{dt} = 3x + y,\ \dfrac{dy}{dt} = 2x + 4y$

c. $\dfrac{dx}{dt} = 8x - 11y,\ \dfrac{dy}{dt} = 6x - 9y$

d. $\dfrac{dx}{dt} = 3x - y,\ \dfrac{dy}{dt} = 3y$

e. $\dfrac{dx}{dt} = -2x - 3y,\ \dfrac{dy}{dt} = 3x - 2y$

Exercise 15.10. Consider the following differential equation:

$$\begin{aligned} \frac{dx}{dt} &= -y \\ \frac{dy}{dt} &= x \end{aligned} \tag{15.17}$$

a. Generate a phase plane diagram of this system. What do you notice?
b. Verify that $x(t) = A\cos(t)$ and $y(t) = A\sin(t)$ is a solution to this differential equation.
c. An equation of a circle of radius R is $x^2 + y^2 = R^2$. Use implicit differentiation to differentiate this equation. Remember you are differentiating with respect to t, and x and y are functions of time t.
d. Substitute the differential equation into your implicit derivative to verify $x^2 + y^2 = R^2$ is a solution to the differential equation.
e. Verify that $x(t) = A\cos(t) + B\sin(t)$ and $y(t) = A\sin(t) - B\cos(t)$ are also solutions.
f. Make a plot of $x(t) = A\cos(t) + B\sin(t)$ and $y(t) = -A\sin(t) - B\cos(t)$ for $A = 1$, $B = 1$.

Chapter 16

Systems of nonlinear equations

In Section 15 we discussed systems of linear equations. For this section we focus on *non*-linear systems of equations. Here is the good news: the processes and approaches that we will apply for nonlinear systems are similar to linear equations. While we previously discussed coupled (nonlinear) systems of equations in Section 6, it helps to have a refresher with an example.

Consider the following non-linear system of equations:

$$\begin{aligned} \frac{dx}{dt} &= y - 1 \\ \frac{dy}{dt} &= x^2 - 1 \end{aligned} \tag{16.1}$$

The nullclines for this system are $y = 1$ and $x = \pm 1$, leading to two equilibrium solutions when we create the phaseplane:

```
# Define the range we wish to evaluate this vector field

system_eq <- c(
  dx ~ y - 1,
  dy ~ x^2 - 1
)

phaseplane(system_eq, "x", "y", x_window = c(-2, 2), y_window = c(-2, 2))
```

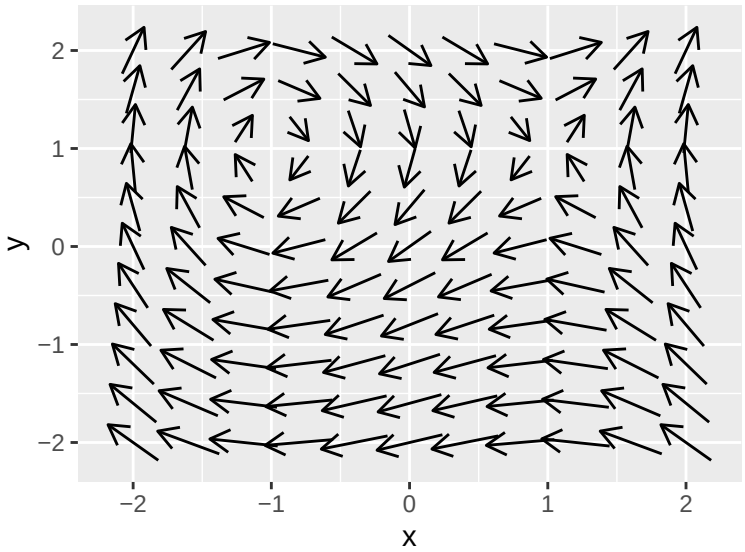


Figure 16.1: Phaseplane for $x' = y - 1$ and $y' = x^2 - 1$.

Wow! The phaseplane in Figure 16.1 looks really interesting. In the upper left corner there is some swirling action, so let's zoom in somewhat:

```
# Define the range we wish to evaluate this vector field

system_eq <- c(
  dx ~ y - 1,
  dy ~ x^2 - 1
)

phaseplane(system_eq, "x", "y", x_window = c(-1.5, -0.5), y_window = c(0.5, 1.5)) +
  geom_point(data = tibble(xint = c(-1), yint = c(1)), aes(x = xint, y = yint), color = "red", size = 4)
```

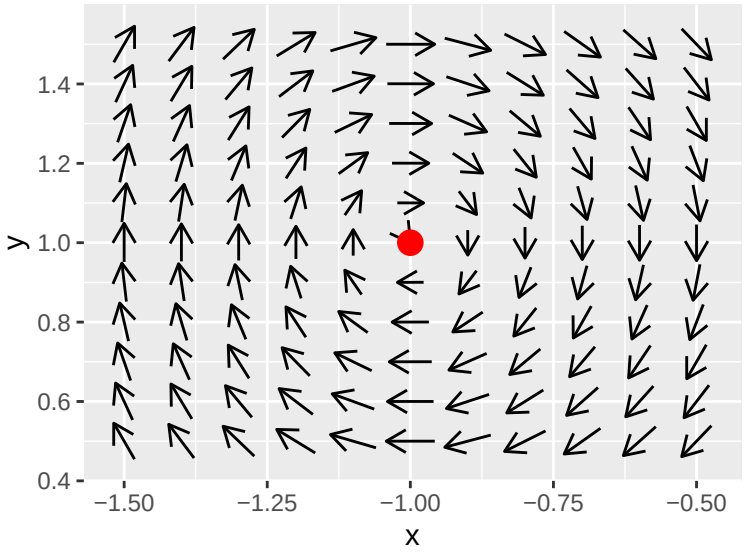


Figure 16.2: Zoomed in phaseplane for $x' = y - 1$ and $y' = x^2 - 1$.

Something interesting is happening at the point $(-1, 1)$ in Figure 16.2. Let's take a look if we evaluate this point in our differential equation:

$$\begin{aligned} \frac{dx}{dt} &= 1 - 1 = 0 \\ \frac{dy}{dt} &= (-1)^2 - 1 = 0 \end{aligned} \tag{16.2}$$

Aha! So the point $(-1, 1)$ is an equilibrium solution. In later sections we will discuss *why* we are observing the behavior with the swirling arrows. For now, the key point from Figure 16.2 is to recognize that *nonlinear systems* have equilibrium solutions as well.

Something else is interesting in the upper right corner of Figure 16.1. Let's zoom in near the point $(1, 1)$:

```
# Define the range we wish to evaluate this vector field
system_eq <- c(
  dx ~ y - 1,
  dy ~ x^2 - 1
)

phaseplane(system_eq, "x", "y", x_window = c(0.5, 1.5), y_window = c(0.5, 1.5)) +
  geom_point(data = tibble(xint = c(1), yint = c(1)), aes(x = xint, y = yint), color = "red", size = 4)
```

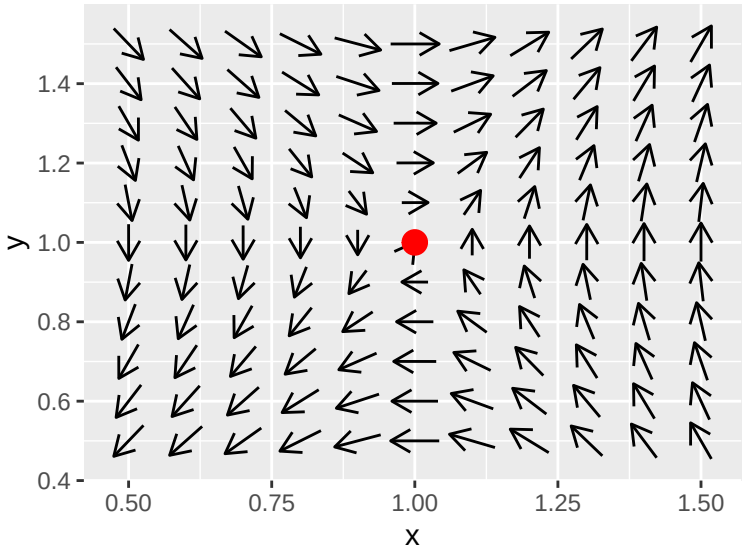


Figure 16.3: Another zoomed in phaseplane for $x' = y - 1$ and $y' = x^2 - 1$.

It seems like there is a *second* equilibrium solution at the point $(1, 1)$! Let's confirm this:

$$\begin{aligned} \frac{dx}{dt} &= 1 - 1 = 0 \\ \frac{dy}{dt} &= (1)^2 - 1 = 0 \end{aligned} \tag{16.3}$$

We learned something important about nonlinear systems compared to linear systems. In Section 15 we learned that the origin is the sole equilibrium solution for a linear system of differential equations. On the other hand, there may be *multiple* equilibrium solutions for nonlinear systems of equations.

16.1 Determining equilibrium solutions

To determine an equilbrium solutions for a system of differential equations we first need to find the intersection of different nullclines. We do this by setting each of the rate equations ($\frac{dx}{dt}$ or $\frac{dy}{dt}$) equal to zero. Equation (16.1) has two nullclines:

$$\begin{aligned} \frac{dx}{dt} &= 0 \to y - 1 = 0 \\ \frac{dy}{dt} &= 0 \to x^2 - 1 = 0 \end{aligned} \tag{16.4}$$

So solving for both nullclines in Equation (16.4) we have that $y = 1$ or $x = \pm 1$. You can visually see the phaseplane with the nullclines in Figure 16.4.

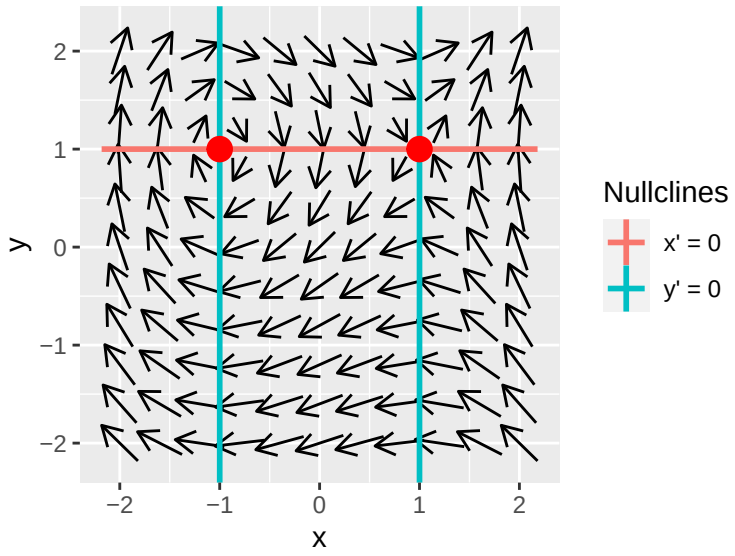


Figure 16.4: Phaseplane for $x' = y - 1$ and $y' = x^2 - 1$.

Remember: equilibrium solutions occur when two *different* nullclines intersect.

Let's try another example:

$$\begin{aligned} \frac{dx}{dt} &= x - 0.5yx \\ \frac{dy}{dt} &= yx - y^2 \end{aligned} \tag{16.5}$$

Figure 16.5 shows the phaseplane for this example. Can you guess where an equilibrium solution would be?

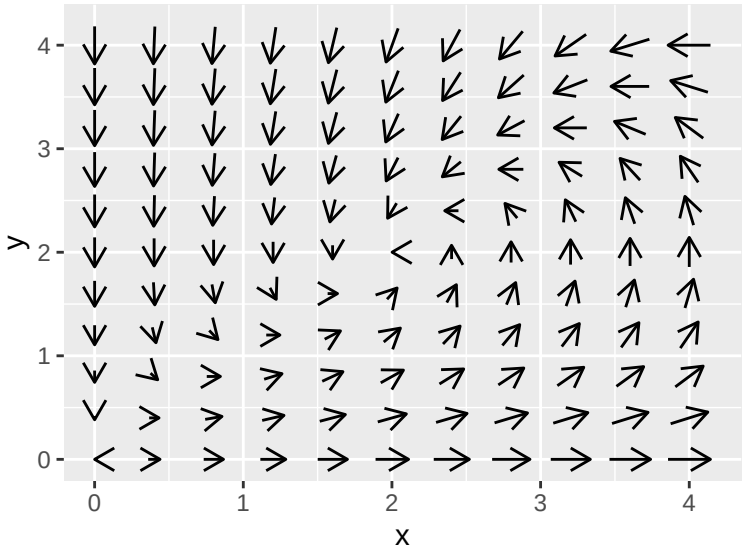


Figure 16.5: Phaseplane for $x' = x - 0.5xy$ and $y' = yx = y^2$.

Let's find the equations for the nullclines:

$$\begin{aligned} \frac{dx}{dt} = 0 &\rightarrow x - 0.5yx = 0 \\ \frac{dy}{dt} = 0 &\rightarrow yx - y^2 = 0 \end{aligned} \tag{16.6}$$

The algebra is becoming a little more involved. Factoring $x - 0.5yx = 0$ we have $x \cdot (1 - 0.5y) = 0$, so either $x = 0$ or $y = 2$. Factoring the second equation we have $y \cdot (x - y) = 0$, so either $y = 0$ or $x = y$. Notice how this second nullcline is a function of x and y.

For Equation (16.5) the equilibrium solutions are $(0, 0)$, $(2, 2)$. You may be tempted to think that $(0, 2)$ is also an equilibrium solution - however - $x = 0$ and $y = 2$ are equations for the x nullcline. It is easy to forget, but equilibrium solutions are determined from the intersection of *distinct* nullclines.

16.2 Stability of an equilibrium solution

The idea of stability of an equilibrium solution for a nonlinear system is intuitively similar to that of a linear system: the equilibrium is stable when all the phaseplane arrows point towards the equilibrium solution. For Equation (16.5), the equilibrium solution at $(1, 1)$ is *unstable* because some of the arrows point towards the equilibrium solution, whereas others point away from it.

For the moment, we won't go into more specifics of how to classify a stable versus unstable equilibrium solution. If you can recognize from the phaseplane equilibrium solutions that appear to be stable (versus those aren't), then you have established some good intuition that can be confirmed with additional analyses. We will study how to do these analyses in the next few sections.

16.3 Exercises

Exercise 16.1. Consider the following nonlinear system of equations:

$$\begin{aligned} \frac{dx}{dt} &= x - .5xy \\ \frac{dy}{dt} &= .5yx - y \end{aligned} \tag{16.7}$$

a. What are the equations for the nullclines for this differential equation?
b. What are the equilibrium solutions for this differential equation?
c. Generate a phaseplane that includes all equilibrium solutions.
d. Based on the phaseplane, evaluate the stability of the equilibrium solution.

Exercise 16.2. (Inspired by Logan and Wolesensky (2009)) A population of fish has natural predators. A model that describes this interaction is the following:

$$\begin{aligned} \frac{dN}{dt} &= N - .3NP \\ \frac{dP}{dt} &= .5NP - P, \end{aligned} \tag{16.8}$$

a. What are the equations for the nullclines for this differential equation?
b. What are the equilibrium solutions for this differential equation?
c. Generate a phaseplane that includes all the equilibrium solutions.
d. Based on the phaseplane, evaluate the stability of the equilibrium solution.

Exercise 16.3. Consider the following system:

$$\begin{aligned} \frac{dx}{dt} &= y^2 \\ \frac{dy}{dt} &= -\frac{2}{3}x, \end{aligned} \tag{16.9}$$

a. What are the nullclines for this system of equations?
b. What is the equilibrium solution for this system of equations?
c. Generate a phaseplane that includes the equilibrium solution.
d. Based on the phaseplane, evaluate the stability of the equilibrium solution.

Exercise 16.4. The *Van der Pol Equation* is a second-order differential equation used to study radio circuits: $x'' + \mu \cdot (x^2 - 1)x' + x = 0$, where μ is a parameter.

a. Let $y' = x$. Show that with this change of variables the Van de Pol equation can be written as a system:

$$\begin{aligned} \frac{dx}{dt} &= y \\ \frac{dy}{dt} &= -x - \mu(x^2 - 1)y \end{aligned} \tag{16.10}$$

b. Verify that the only equilibrium solution is $(0, 0)$.
c. Set $\mu = 1$. Generate a phaseplane that includes the equilibrium solution.
d. Based on the phaseplane, evaluate the stability of the equilibrium solution.

Exercise 16.5. Consider the following nonlinear system:

$$\begin{aligned} \frac{dx}{dt} &= x - y \\ \frac{dy}{dt} &= -y + \frac{5x^2}{4 + x^2}, \end{aligned} \tag{16.11}$$

a. What are the two equations for the nullclines?
b. Using desmos (or some other graphing utility), graph the two nullclines simultaneous. What are the intersection points?
c. Generate a phaseplane for this system that contains all the equilibrium solutions.
d. Let's say instead that $\frac{dx}{dt} = bx - y$, where b is a parameter such that $0 \leq b \leq 2$. How many equilibrium solutions do you have as b changes?

Exercise 16.6. (Inspired by Logan and Wolesensky (2009)) Let C be the amount of carbon in a forest ecosystem, with P be the rate of increase due to photosynthesis. Herbivores H consume carbon on the following predator-prey model:

$$\begin{aligned} \frac{dC}{dt} &= -aC - bHC \\ \frac{dH}{dt} &= ebHC - dC \end{aligned} \tag{16.12}$$

a. What are the equations for the nullclines?
b. Determine the equilibrium solutions.

Chapter 17

Local Linearization and the Jacobian

In Sections 15 and 16 we focused on systems of differential equations and the types of behaviors we should expect when analyzing the stability of the equilibrium solution. In this section we will analyze stability of a nonlinear system through the concept of *local linearization*, beginning to build a bridge between nonlinear and linear systems of equations.

17.1 A first example

Let's take a look at the following phase plane to this non-linear system:

$$\begin{aligned} \frac{dx}{dt} &= y - 1 \\ \frac{dy}{dt} &= x^2 - 1 \end{aligned} \tag{17.1}$$

The nullclines for this system are $y = 1$ and $x = \pm 1$, leading two two equilibrium solutions when we create the phaseplane:

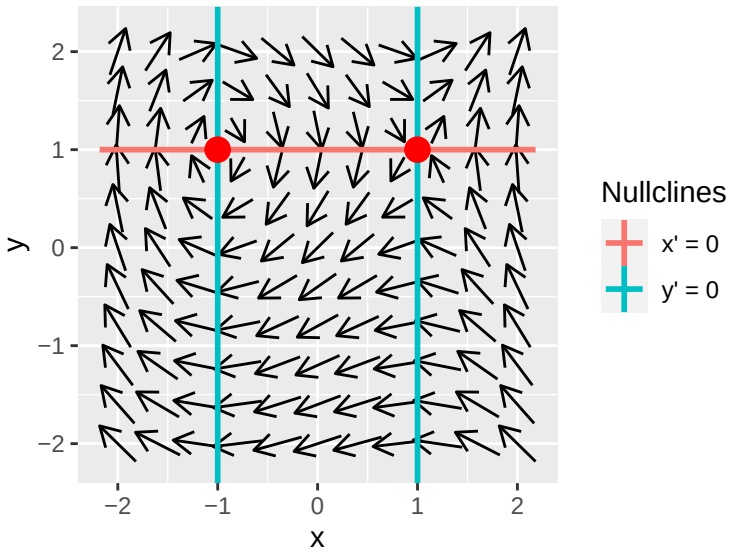


Figure 17.1: Phaseplane for $x' = y - 1$ and $y' = x^2 - 1$.

From Figure 17.1, arrows on the phaseplane by the equilibrium solution at $(x, y) = (-1, 1)$ seems to spiral inwards, whereas the second equilibrium solution at $(x, y) = (1, 1)$ has behavior that flows in and out depending on the direction you approach. Beyond this visualization we can define additional tools to help characterize the stability of this system near the equilibrium solutions.

This section is focused on understanding the behavior near the equilibrium solutions, and in particular applying concepts of *local linearization* along with our understanding of linear systems. Let's get started!

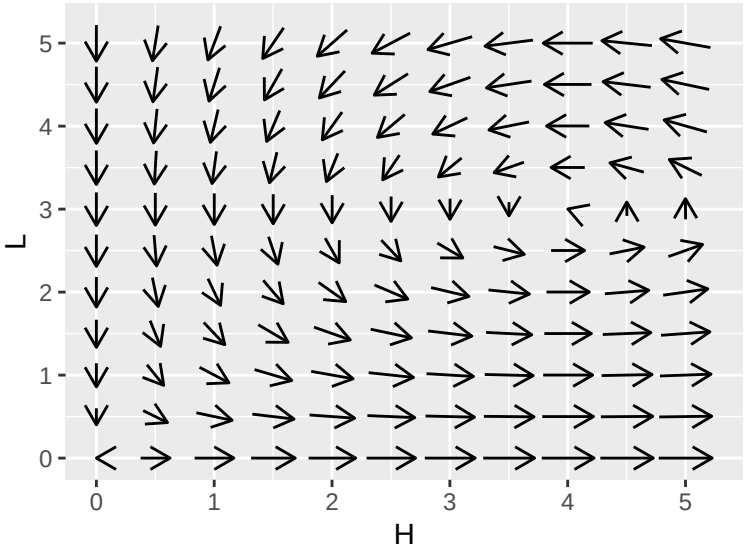
17.2 The lynx hare revisited

Let's take a look at another familiar example. Consider the following nonlinear system of equations from the Lynx-Hare model, with H and L measured in thousands of animals:

$$\begin{aligned} \frac{dH}{dt} &= .3H - .1HL \\ \frac{dL}{dt} &= .05HL - .2L \end{aligned} \tag{17.2}$$

You can show that the steady states for Equation (17.2) are $(H, L) = (0, 0)$ and $(4, 3)$. The phase plane diagram for this system is the following:

```
# Define the range we wish to evaluate this vector field
H_window <- c(0,5)
L_window <- c(0,5)

system_eq <- c(dH ~ .3*H - .1*H*L,
               dL ~ .05*H*L -.2*L)

# Reminder: The values in quotes are the labels for the axes
phaseplane(system_eq,'H','L',x_window = H_window, y_window = L_window)
```



Let's take a closer look at the phase plane near the first equilibrium solution:

```
# Define the range we wish to evaluate this vector field
H_window <- c(0,0.5)
L_window <- c(0,0.5)

system_eq <- c(dH ~ .3*H - .1*H*L,
               dL ~ .05*H*L -.2*L)

# Reminder: The values in quotes are the labels for the axes
phaseplane(system_eq,'H','L',x_window = H_window, y_window = L_window)
```

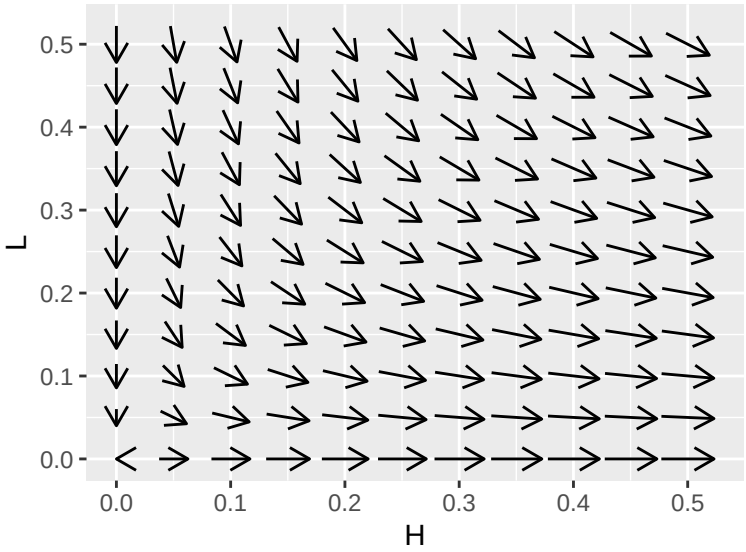


Figure 17.2: A zoomed in view of the lynx-hare system.

While the phaseplane in Figure 17.2 *looks* like this equilibrium solution is unstable, verifying this with another approach would be useful. To do so, we are going to do a **locally linear approximation** or **tangent plane approximation** around $H = 0$, $L = 0$, which we discuss next.

17.3 Tangent plane approximations

To extend the notion of a linear approximation, we apply the **tangent plane approximation** at the point $x = a$, $y = b$:

$$L(x, y) = f(a, b) + f_x(a, b) \cdot (x - a) + f_y(a, b) \cdot (y - b), \tag{17.3}$$

where f_x is the partial derivative of $f(x, y)$ with respect to x and f_y is the partial derivative of $f(x, y)$ with respect to y. For simplicity let's work with the equilibrium solution at $(0, 0)$. If we say that $f(H, L) = .3H - .1HL$, we know $f(0, 0) = 0$. Furthermore the locally linear approximation is:

$$\begin{aligned} f_H = .3 - .1L \to f_H(0, 0) = .3 \\ f_L = -.1H \to f_L(0, 0) = 0 \end{aligned} \tag{17.4}$$

With this information we are able to take this and then write down the locally linear approximation for $f(H, L)$:

$$f(H, L) \approx 0 + .3 \cdot (H - 0) - 0 \cdot (L - 0) = .3H \tag{17.5}$$

Likewise if we consider $g(H, L) = .05HL - .2L$, then we have:

$$\begin{aligned} g_H = .05L - .2L \to g_H(0, 0) = 0 \\ g_L = .05H - .2 \to f_L(0, 0) = -.2 \end{aligned} \tag{17.6}$$

With these results, our locally linear approximation for $g(H, L)$ is:

$$g(H, L) \approx 0 + 0 \cdot (H - 0) - .2 \cdot (L - 0) = 0 - .2L. \tag{17.7}$$

So, at the equilibrium solution $(0, 0)$, Equation (17.2) behaves like the following system of equations:

$$\frac{dH}{dt} = .3H \tag{17.8}$$
$$\frac{dL}{dt} = -.2L \tag{17.9}$$

Notice how Equation (17.9) is an entirely decoupled linear system of equations that can be solved easily by separation of variables. For an initial condition (H_0, L_0) Equation (17.9) has a solution $H(t) = H_0 e^{.3t}$ and $L(t) = L_0 e^{-.2t}$.

17.4 The Jacobian matrix

RevisitingEquation (17.9) we can write our system in matrix form:

$$\begin{pmatrix} H' \\ L' \end{pmatrix} = \begin{pmatrix} .3 & 0 \\ 0 & -.2 \end{pmatrix} \begin{pmatrix} H \\ L \end{pmatrix} \tag{17.10}$$

We define the matrix $J = \begin{pmatrix} .3 & 0 \\ 0 & -.2 \end{pmatrix}$ as the *Jacobian matrix*. This matrix is key matrix to investigate stability of nonlinear systems.

17.5 Predator prey with logistic growth

Let's take a look at another model developed from the lynx-hare system. We assumed that the hare grow exponentially (notice the term rH in their equation.) However we can modify their growth rate to be a logistic growth function with carrying capacity K:

$$\begin{aligned} \frac{dH}{dt} &= rH\left(1 - \frac{H}{K}\right) - bHL \\ \frac{dL}{dt} &= ebHL - dL \end{aligned} \tag{17.11}$$

Through a rescaling of Equation (17.11) with the variables $x = \frac{H}{K}$, $y = \frac{L}{r/b}$ and $T = rt$ we can rewrite Equation (17.11) as:

$$\begin{aligned} \frac{dx}{dT} &= x(1-x) - xy \\ \frac{dy}{dT} &= \frac{ebK}{r}xy - \frac{d}{r}y \end{aligned} \tag{17.12}$$

In order to analyze the Jacobian matrix for Equation (17.12) we will need to compute several partial derivatives:

$$\begin{aligned} \frac{\partial}{\partial x}\left(f(x,y)\right) &= \frac{\partial}{\partial x}\left(x(1-x) - xy\right) = 1 - 2x - y \\ \frac{\partial}{\partial y}\left(f(x,y)\right) &= \frac{\partial}{\partial y}\left(x(1-x) - xy\right) = -x \\ \frac{\partial}{\partial x}\left(g(x,y)\right) &= \frac{\partial}{\partial x}\left(\frac{ebK}{r}xy - \frac{d}{r}y\right) = \frac{ebK}{r}y \\ \frac{\partial}{\partial y}\left(g(x,y)\right) &= \frac{\partial}{\partial y}\left(\frac{ebK}{r}xy - \frac{d}{r}y\right) = \frac{ebK}{r}x - \frac{d}{r} \end{aligned} \tag{17.13}$$

So now we can construct the Jacobian matrix:

$$J_{(x,y)} = \begin{pmatrix} 1-2x-y & -x \\ \frac{ebK}{r}y & \frac{ebK}{r}x - \frac{d}{r} \end{pmatrix} \tag{17.14}$$

The notation $J_{(x,y)}$ signifies the Jacobian matrix evaluated at the equilibrium solution (x, y). Sometimes computing the Jacobian matrix is a good first step so then you are ready to compute the equilibrium solutions. In the exercises you will determine equilibrium solutions and visualize the Jacobian matrix.

17.6 Concluding thoughts

To summarize, let's say we have the following system of equations

$$\begin{aligned} \frac{dx}{dt} &= f(x,y) \\ \frac{dy}{dt} &= g(x,y) \end{aligned} \tag{17.15}$$

Assuming we have an equibrium solution at $(x, y) = (a, b)$, the Jacobian matrix at that solution is:

$$J_{(a,b)} = \begin{pmatrix} f_x(a,b) & f_y(a,b) \\ g_x(a,b) & g_y(a,b) \end{pmatrix} \tag{17.16}$$

While we don't discuss it here, the Jacobian matrix also extends to higher order systems as well.

17.7 Exercises

Exercise 17.1. Consider the following nonlinear system:

$$\begin{aligned} \frac{dx}{dt} &= y - 1 \\ \frac{dy}{dt} &= x^2 - 1 \end{aligned} \tag{17.17}$$

a. Verify that this system has equilibrium solutions at $(-1, 1)$ and $(1, 1)$.
b. Construct the Jacobian matrix at the equilibrium solutions at $(-1, 1)$ and $(1, 1)$.
c. With the Jacobian matrix, visualize a phaseplane at these equilbrium solutions to determine stability.

Exercise 17.2. By solving directly, show that $(H, L) = (0, 0)$ and $(4, 3)$ are equilibrium solutions to the following system of equations:

$$\begin{aligned} \frac{dH}{dt} &= .3H - .1HL \\ \frac{dL}{dt} &= .05HL - .2L \end{aligned} \tag{17.18}$$

Exercise 17.3. Consider the following nonlinear system:

$$\begin{aligned} \frac{dx}{dt} &= x - y \\ \frac{dy}{dt} &= -y + \frac{5x^2}{4 + x^2}, \end{aligned} \tag{17.19}$$

a. Verify that the point $(x, y) = (1, 1)$ is an equilibrium solution.
b. Construct the Jacobian matrix at this equilibrium solution.
c. With the Jacobian matrix, visualize a phaseplane at that equilbrium solution to determine stability.

Exercise 17.4. Consider the nonlinear system

$$\begin{aligned} \frac{dx}{dt} &= 2x + 3y + xy \\ \frac{dy}{dt} &= -x + y - 2xy^3, \end{aligned} \tag{17.20}$$

a. Verify that the point $(0, 0)$ is an equilibrium solution.
b. Determine the Jacobian matrix at this equilbrium solution.
c. With the Jacobian matrix, visualize a phaseplane at that equilbrium solution to determine stability.

Exercise 17.5. Consider the following system:

$$\begin{aligned} \frac{dx}{dt} &= y^2 \\ \frac{dy}{dt} &= -\frac{2}{3}x, \end{aligned} \tag{17.21}$$

a. Determine the equilibrium solution.

b. Construct the Jacobian at the equilibrium solution.
c. With the Jacobian matrix, visualize a phaseplane at that equilbrium solution to determine stability.
d. Use the fact that that $\frac{dy}{dx}/\frac{dy}{dt} = \frac{dy}{dx}$, which should yield a separable differential equation that will allow you to solve $y(x)$. How does that solved differential equation compare to what you found by analyzing the Jacobian matrix?

Exercise 17.6. Consider the lynx-hare system with parameters r, b, e, and d:

$$\begin{aligned} \frac{dH}{dt} &= rH - bHL \\ \frac{dL}{dt} &= ebHL - dL \end{aligned} \tag{17.22}$$

a. Verify that the point $\left(\frac{d}{eb}, \frac{r}{b}\right)$ is an equilibrium solution.
b. Construct the Jacobian at the equilibrium solution. (Your final answer will include the parameters.)

Exercise 17.7. This problem revisits the modified lynx-hare system:

$$\begin{aligned} \frac{dH}{dt} &= rH\left(1 - \frac{H}{K}\right) - bHL \\ \frac{dL}{dt} &= ebHL - dL \end{aligned} \tag{17.23}$$

a. Verify that by rescaling of this system $x = \frac{H}{K}$, $y = \frac{L}{r/b}$ and $T = rt$ we can rewrite this system as:

$$\begin{aligned} \frac{dx}{dT} &= x(1-x) - xy \\ \frac{dy}{dT} &= \frac{ebK}{r}xy - \frac{d}{r}y \end{aligned} \tag{17.24}$$

b. Determine the three equilibrium solutions of this rescaled system.
c. Evaluate the Jacobian at each of the equilibrium solutions.
d. With your Jacobian, select reasonable values of the parameters to generate a phaseplane diagram and classify the stability of each equilibrium solution.
e. How does this revised system compare to the stability of the regular lynx-hare model?

Exercise 17.8. (Inspired by Logan and Wolesensky (2009)) A model for the spread of a disease where people recover is given by the following differential equation:

$$\begin{aligned} \frac{dS}{dt} &= -\alpha SI \\ \frac{dI}{dt} &= \alpha SI - \gamma I \\ \frac{dR}{dt} &= \gamma I, \end{aligned} \tag{17.25}$$

a. Determine the equilibrium solutions for this system of equations.
b. Construct the Jacobian for each of the equilibrium solutions.
c. Let $\alpha = 0.001$ and $\gamma = 0.2$. With the Jacobian matrix, generate the phase plane (using the equations for $\frac{dS}{dt}$ and $\frac{dI}{dt}$ only) for all of the equilibrium solutions and classify their stability.

Exercise 17.9. (Inspired by Logan and Wolesensky (2009)) A population of fish used as food is an example of a renewable resource. This population decreases due to harvesting the fish. As long as the rate of harvest is smaller than the replacement rate, theoretically the population would be considered a renewable resource. However fish do have natural predators, which may also be harvested at the same time. A model that describes this interaction is

$$\begin{aligned} \frac{dN}{dt} &= rN - cNP - \rho EN \\ \frac{dP}{dt} &= bcNP - mP - \sigma EP, \end{aligned} \tag{17.26}$$

where E is the fishing effort and ρ and σ are the catchability coefficients for the prey and predator.

a. Determine all the equilibrium solutions for this model.
b. Construct the Jacobian matrix for each of the the equilibrium solutions.
c. Select different parameter values. With the Jacobian, construct a phase plane diagram for each of the equilibrium solutions.
d. In the absence of fishing $E = 0$. How do these equilibrium solutions compare to the ones from the previous part?

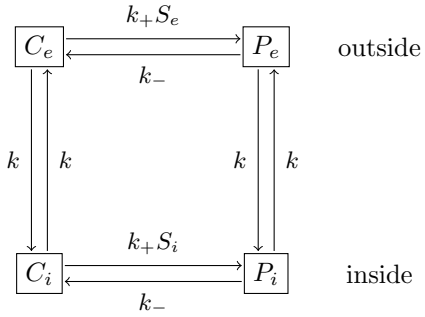


Figure 17.3: Glucose transporter reaction schemes.

Exercise 17.10. (Inspired by Keener et al. (2009)) The chemical glucose is transported across the cell membrane using a carrier proteins. These proteins can have different states (open or closed) that can be bound to a glucose substrate. The schematic for this reaction is shown in Figure 17.3. The system of differential equations describing this reaction is:

$$\begin{aligned} \frac{dp_i}{dt} &= kp_e - kp_i + k_+ s_i c_i - k_i p_i \\ \frac{dp_e}{dt} &= kp_i - kp_e + k_+ s_e c_e - k_- p_e \\ \frac{dc_i}{dt} &= kc_e - kc_i + k_- p_i - k_+ s_i c_i \\ \frac{dc_e}{dt} &= kc_i - kc_e + k_- p_e - k_+ s_e c_e \end{aligned} \tag{17.27}$$

a. We can reduce this to a system of three equations. First show that $\frac{dp_i}{dt} + \frac{dp_e}{dt} + \frac{dc_i}{dt} + \frac{dc_e}{dt} = 0$. Given that $p_i + p_e + c_i + c_e = C_0$, where C_0 is constant, use this equation to eliminate p_i and write down a system of three equations.
b. Determine the equilibrium solutions for this new system of three equations.
c. Construct the Jacobian matrix for each of these equilibrium solutions.

Chapter 18

What are eigenvalues?

To understand a system of differential equations it is key to analyze the locally linear behavior of the system near any equilibrium solution. Consider the following representative phaseplane from Section 17:

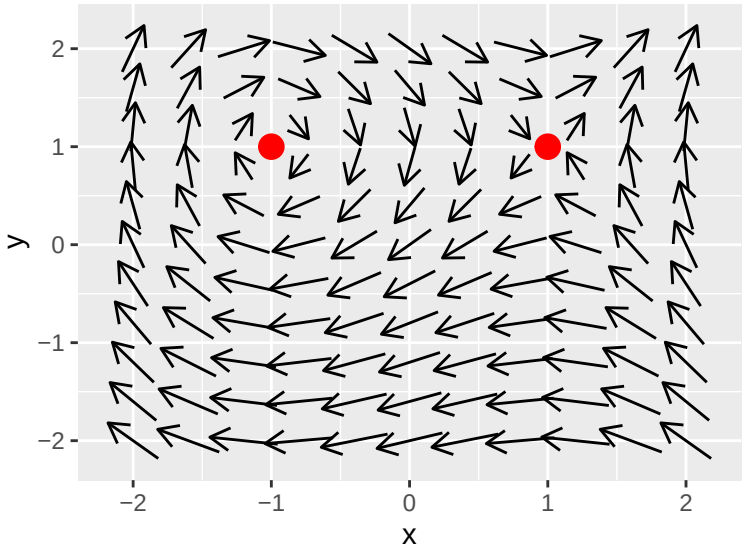


Figure 18.1: A representative phase plane.

Are the equilibrium solutions (red dots in Figure 18.1) stable? In this section we will develop an algebraic process to analyze locally linear behavior, connecting what we learned to straight line solutions (Section 15) and the Jacobian (Section 17).

18.1 Straight line solutions

Consider this following linear system of equations:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{18.1}$$

In Section 15 we found the two straight line solutions:

- Solution 1: $\vec{s}_1(t) = \begin{pmatrix} 0 \\ e^t \end{pmatrix} = \begin{pmatrix} 0 \\ e^t \end{pmatrix} = e^t \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.
- Solution 2: $\vec{s}_2(t) = \begin{pmatrix} e^{2t} \\ e^{2t} \end{pmatrix} = \begin{pmatrix} e^{2t} \\ e^{2t} \end{pmatrix} = e^{2t} \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

Let's verify that Solution 2 is indeed a solution to this linear system. First we will take the derivative of Solution 2:

$$\frac{d}{dt}\left(\vec{s}_2(t)\right) = \frac{d}{dt}\left(e^{2t}\begin{pmatrix}1\\1\end{pmatrix}\right) = 2e^{2t}\begin{pmatrix}1\\1\end{pmatrix} \tag{18.2}$$

Let's compare this solution to the right hand side of the differential equation:

$$\begin{pmatrix}2 & 0\\1 & 1\end{pmatrix}\left(e^{2t}\begin{pmatrix}1\\1\end{pmatrix}\right) = \left(e^{2t}\begin{pmatrix}2\\2\end{pmatrix}\right) = 2e^{2t}\begin{pmatrix}1\\1\end{pmatrix} \tag{18.3}$$

So, indeed Solution 2 *is* a solution to the differential equation. However something interesting is occurring. Notice how $\frac{d}{dt}\left(\vec{s}_2(t)\right)$ equals $2\vec{s}_2(t)$, so if $\frac{d}{dt}\left(\vec{s}_2(t)\right) = \begin{pmatrix}2 & 0\\1 & 1\end{pmatrix}\vec{s}_2(t)$, then the following equality has to hold:

$$\begin{pmatrix}2 & 0\\1 & 1\end{pmatrix}\vec{s}_2(t) = 2\vec{s}_2(t) \tag{18.4}$$

In fact, for any linear system $\vec{y'} = A\vec{y}$, straight line solutions have the property that $A\vec{v} = \lambda\vec{v}$. In the example we just computed $\vec{v} = \begin{pmatrix}1\\1\end{pmatrix}$, and $\lambda = 2$.

Some apply some terminology here. For these special straight line solutions, we give a particular name to $\vec{v}$ - we call it the *eigenvector*. The name we give to λ is the *eigenvalue*. (Eigen means *own* in German - get it?)

Notice how $s_2(t)$ was expressed as $e^{2t}\begin{pmatrix}1\\1\end{pmatrix}$. In fact, any straight line solution has the form $\vec{s} = e^{\lambda t}\vec{v}$, where λ and $\vec{v}$ are the eigenvalue and eigvenvector.

So how do we *determine* an eigenvalue or eigenvector? A straight line solution satisfies the equation $A\vec{v} = \lambda v$. Re-arranging this equation we can express this as $(A - \lambda I)\vec{v} = \vec{0}$, where $\vec{0}$ is a vector of all zeros and I is called the *identity matrix*, or a square matrix with ones along the diagonal and zero everywhere else. The goal is to find a λ and $\vec{v}$ consistent with this equation.

The eigenvalues of λ that solve $(A\vec{v} - \lambda\vec{v}) = 0$ can be found by solving $\det(A - \lambda I) = 0$, where $\det(M)$ is the determinant. Once the eigenvalues are found, we then determine the eigenvectors.

Time out. I recognize that we are starting to get deeper into linear algebra. We will just borrow some key results that we will need - so hopefully that will give you a leg up when you study linear algebra - it is a great topic! Let's get to work.

18.2 Computing eigenvalues and eigenvectors

Let's understand a little on solving $\det(A - \lambda I) = 0$ to determine the eigenvalues.

First consider the 2 by 2 matrix:

$$A = \begin{pmatrix}a & b\\c & d\end{pmatrix},$$

So then $A - \lambda I$ is the matrix:

$$A - \lambda I = \begin{pmatrix}a - \lambda & b\\c & d - \lambda\end{pmatrix}$$

The determinant of a 2 by 2 matrix is the product of the diagonal entries (for $A - \lambda I$ they are $(a - \lambda) \cdot (d - \lambda)$) less the product of the off-diagonal entries (in this case bc). So $\det(A - \lambda I) = 0$ is the equation $(a - \lambda)(d - \lambda) - bc = 0$. When we multiply this equation out we obtain a quadratic equation to solve (for λ - see Exercise 18.7). The equation $\det(A - \lambda I) = 0$ even has a special name - it is called the *characteristic equation*. (We will find out why later.)

Example 18.1. Compute the eigenvalues for the matrix $A = \begin{pmatrix}-1 & 1\\0 & 3\end{pmatrix}$.

Solution. The matrix $A - \lambda I$ is: $A - \lambda I = \begin{pmatrix} -1-\lambda & 1 \\ 0 & 3-\lambda \end{pmatrix}$. So we have:

$$\det(A - \lambda I) = (-1-\lambda)(3-\lambda) - 0 = 0 \tag{18.5}$$

Solving the equation $(-1-\lambda)(3-\lambda) = 0$ yields two eigenvalues: $\lambda = -1$ or $\lambda = 3$.

Once we have determined the eigenvalues we next compute the eigenvectors associated with each eigenvalue. Remember that an eigenvector is a vector $\vec{v}$ consistent with $A\vec{v} = \lambda\vec{v}$ or $A\vec{v} - \lambda\vec{v} = \vec{0}$. Note that the eigenvectors have the form $\vec{v} = \begin{pmatrix} x \\ y \end{pmatrix}$.

Example 18.2. Compute the eigenvectors for the matrix $A = \begin{pmatrix} -1 & 1 \\ 0 & 3 \end{pmatrix}$.

Solution. So if we consider the right hand side of $A\vec{v} - \lambda\vec{v} = \vec{0}$ we have:

$$(A\vec{v} - \lambda\vec{v}) = \begin{pmatrix} -x + y - \lambda x \\ 3y - \lambda y \end{pmatrix}.$$

.

Examining this setup gives us two equations to work with: $-x + y - \lambda x = 0$ and $3y - \lambda y = 0$. We will need to analyze them for each of our eigenvalues.

- Case 1: $\lambda = -1$. In the first equation we have $-x + y + x = 0$, which just yields $y = 0$. For the second equation we also have $3y + y = 0$, so that tells us again that $y = 0$. Notice how we solved for y, but we didn't uniquely determine x. While this seems a little unsatisfying, that is ok. The form of this particular straight line solution is $s_1(t) = e^{-t}\begin{pmatrix} x \\ 0 \end{pmatrix}$, where x is a free variable.
- Case 2: $\lambda = 3$. For the second equation we have $3y - 3y = 0$, which is always true. However in the first equation we have $-x + y - 3x = 0$, or $y = 4x$. In this case, x can be anything as well, but this condition means that y will have to be 4 times that value. Hence, this particular straight line solution is $s_2(t) = e^{3t}\begin{pmatrix} x \\ 4x \end{pmatrix}$.

Notice that in both of our cases we had a free variable x. Given an initial condition to the differential equation we would specify this free variable (a good choice would be $x = 1$), or just leave this as a constant.

Once we have computed the eigenvalues and eigenvectors, we are now ready to express the most general solution for a system of differential equations. For a two-dimensional system of linear differential equations ($\frac{d}{dt}\vec{x} = A\vec{x}$, the most general solution is $\vec{x}(t) = c_1 e^{\lambda_1 t}\vec{v}_1 + c_2 e^{\lambda_2 t}\vec{v}_2$

Example 18.3. What is the solution to the differential equation $\frac{d}{dt}\vec{x} = \begin{pmatrix} -1 & 1 \\ 0 & 3 \end{pmatrix}\vec{x}$?.

Solution. Since we have already computed the eigenvalues and eigenvectors, our most general solution is:

$$\vec{x} = c_1 e^{-t}\begin{pmatrix} 1 \\ 0 \end{pmatrix} + c_2 e^{3t}\begin{pmatrix} 1 \\ 4 \end{pmatrix}$$

18.2.1 Eigenvalues computed with demodelr

While computing eigenvalues and eigenvectors is a good algebraic exercise, we can also program this in `R` using the function `eigenvalues` from the `demodelr` package. The syntax works where $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ is entered in as `eigenvalues(a,b,c,d,matrix_rows)` where `matrix_rows` is the number of rows[1]. What gets returned from the function will be the eigenvalues and eigenvectors for any square matrix.

```
# For a two dimensional equation the code assumes the default is a 2 by 2 matrix, .
eigenvalues(matrix_entries = c(-1, 1, 0, 3),
            matrix_rows = 2)
```

[1]If you have a 2 by 2 matrix, you can leave out `matrix_rows` (so just `eigenvalues(a,b,c,d)`) as the default is a 2 by 2 matrix.

```
## eigen() decomposition
## $values
## [1]  3 -1
##
## $vectors
##           X1 X2
## 1 0.2425356  1
## 2 0.9701425  0
```

```
# This is equivalent because we have a two dimensional equation:
#eigenvalues(c(-1, 1, 0, 3))
```

Notice that the eigenvalues and the eigenvectors get returned. How you read the output for the eigenvector is that `X1` is the eigenvector associated with the first eigenvalue ($\lambda = 3$) and `X2` is the eigenvector associated with the second eigenvalue ($\lambda = -1$). The eigenvector associated with $\lambda = 3$ is a little different from what we computed - `R` will normalize the vector, which means that its total length will be one.[2] However upon closer inspection in our solution we found that the second component was 4 times the first for the eigenvector, which is indeed the case.

18.3 What do eigenvalues tell us?

Here the focus of the section changes a little bit from finding the solution of the differential equation (e.g. the formulas $s_1(t)$ and $s_2(t)$) to understanding what the solutions would *look* like in the phaseplane. This is intentional: once we have found the eigenvalues, finding eigenvectors can seem rather mundane at times (perhaps heavy on the algebra), yet we know they will be a straight line through the origin. So what else is there?

Studying the eigenvalues helps us understand the qualitative nature of the solution to a differential equation. Let's think about the characteristic equation for a 2 by 2 matrix, simplifying things out:

$$\begin{aligned} \det(A - \lambda I) &= 0 \\ (a - \lambda)(d - \lambda) - bc &= 0 \\ \lambda^2 - (a + d)\lambda + ad - bc &= 0 \end{aligned}$$

The last line of the above equation is a quadratic equation in λ. Think about what you know about quadratic equations: there may be 2 distinct solutions, no solution (the solutions are imaginary) or 1 solution (repeated solutions). Also the solutions may be positive or negative. There are so many different combinations! What types of phaseplanes do all those different types of eigenvalues produce? Let's take a look at each of these cases, focusing our analysis on a two-dimensional system of equations (the results can be generalized to higher-dimensional systems).

18.3.1 All eigenvalues positive (unstable node)

Let's consider the phase plane for the differential equation $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

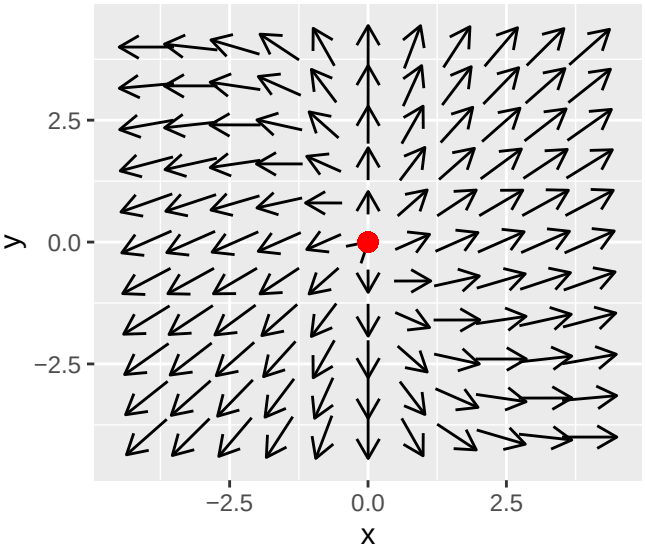
The eigenvalues are both positive:

```
eigenvalues(c(2, 0, 1, 1))
```

```
## eigen() decomposition
## $values
## [1] 2 1
##
## $vectors
##           X1 X2
## 1 0.7071068  0
## 2 0.7071068  1
```

[2] The length of a vector $\vec{v}$ is denoted as $||\vec{v}||$ and is computed the following way: $||\vec{v}|| = \sqrt{v_1^2 + v_2^2 + ... + v_n^2}$. We normalize a vector to a length of 1 by dividing each component by its length.

The phase plane for this diagram is shown below:



Notice how the phase plane diagram has all arrows pointing from the origin. In the phaseplane a solution would move *away*, so the origin is an unstable node.

Plotting both solutions versus time shows woudl show x and y increase in time.

18.3.2 All eigenvalues negative (stable node)

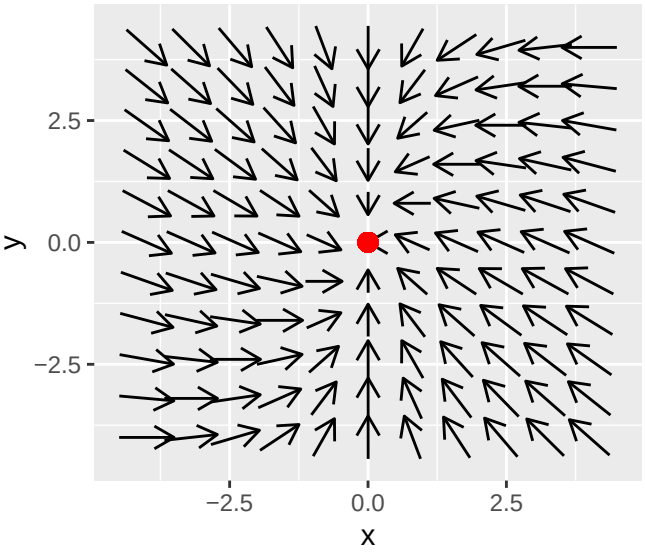
Now consider the phase plane for the differential equation $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -2 & 0 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

(This is a *slight* modification from the previous example.) The eigenvalues are both positive:

```
eigenvalues(c(-2, 0, 1, -1))
```

```
## eigen() decomposition
## $values
## [1] -2 -1
##
## $vectors
##           X1 X2
## 1  0.7071068  0
## 2 -0.7071068  1
```

In this case the situation is reversed, with the phase plane having all arrows pointing to the origin.



We say the equilibrium solution is a *stable node*. In the phaseplane a solution would eventually move towards the origin (asymptotically)

18.3.3 One positive one negative eigenvalue

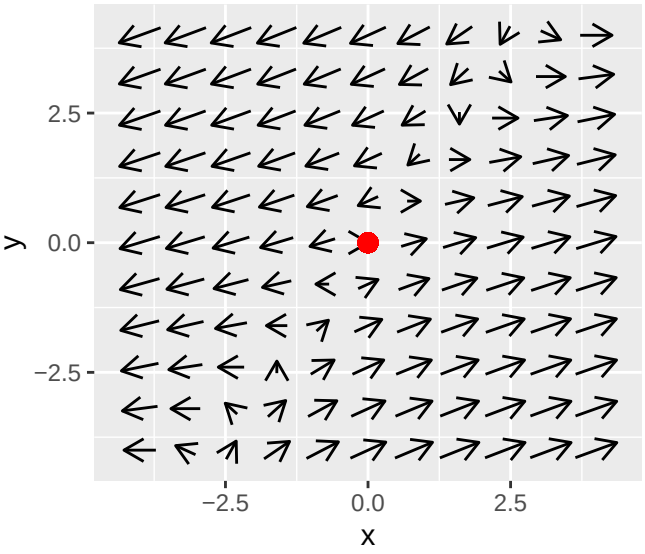
Let's consider the phase plane for the differential equation $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 3 & -2 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

The eigenvalues are both positive:

```
eigenvalues(c(3, -2, 1, -1))
```

```
## eigen() decomposition
## $values
## [1]  2.4142136 -0.4142136
##
## $vectors
##           X1        X2
## 1 0.9596830 0.5054495
## 2 0.2810846 0.8628562
```

This situation is called a saddle node, best explained with a phase diagram:



This equilibrium solution is called a *saddle node*. From one direction (the horizontal) the arrows point away from the origin, but in the vertical direction the arrows point towards the origin. We will see that this behavior is due to the contradictory nature of the solution - one part of the solution (the one associated with the negative eigenvalue) decays asymptotically to zero. The other positive eigenvalue is associated with the asymptotically unstable, giving the solution trajectory the saddle shape.

18.3.4 Imaginary eigenvalues

Let's consider the phase plane for the differential equation $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} -3 & -8 \\ 4 & -6 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

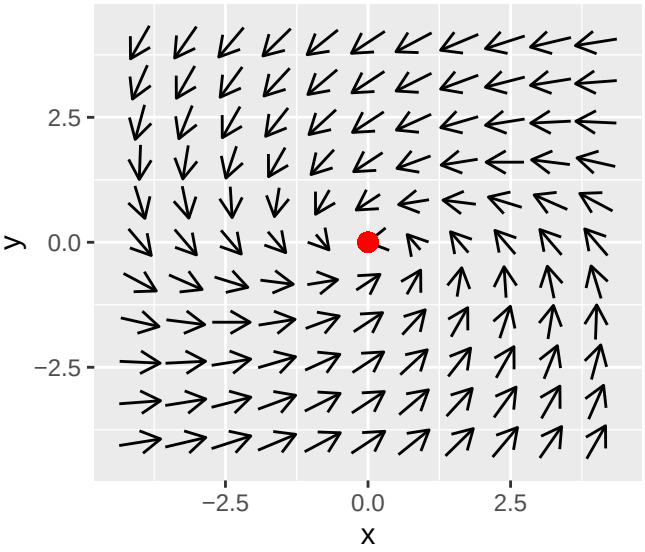
The eigenvalues are both positive:

```
eigenvalues(c(-3, -8, 4, -6))
```

```
## eigen() decomposition
## $values
## [1] -4.5+5.454356i -4.5-5.454356i
##
## $vectors
##                       X1                    X2
## 1 0.8164966+0.0000000i 0.8164966+0.0000000i
## 2 0.1530931-0.5566829i 0.1530931+0.5566829i
```

There are two eigenvalues to this system: $\lambda = -4.5 + 5.45i$ and $\lambda = -4.5 - 5.45i$. In this case the i means the eigenvalues are imaginary. Notice how the eigenvalues are similar, but the signs on the second term differs. We say the eigenvalues are *complex conjugates* of each other, and write them in the form $\lambda = \alpha \pm \beta i$.

Let's take a look at the phase plane for this system. When $\alpha < 0$ the equilibrium solution is a spiral sink:

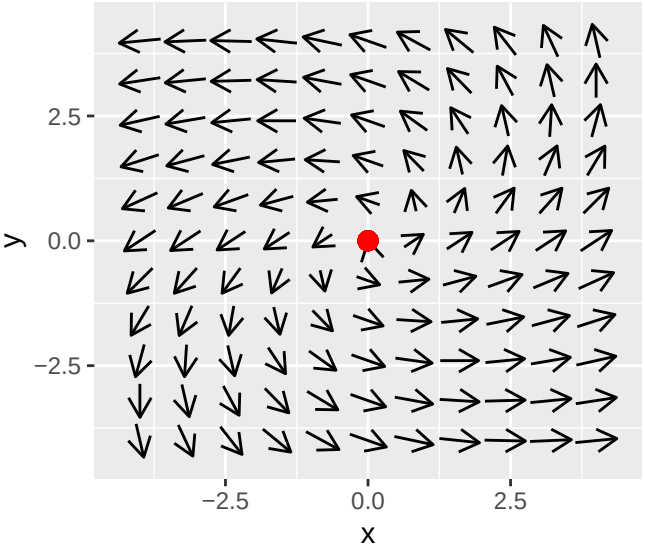


This phaseplane has some spiraling motion to it. Why does that occur? Imaginary eigenvalues can occur when the solution to $\det(A - \lambda I) = 0$ has imaginary solutions. More generally, we say $\lambda = \alpha \pm \beta i$. Because the eigenvalues are complex, we would also expect the eigenvectors to be complex as well (i.e. $\vec{v} \pm i\vec{w}$). Don't let the term *imaginary* fool you: by using properties from complex analysis it can be shown that the complete solution is:

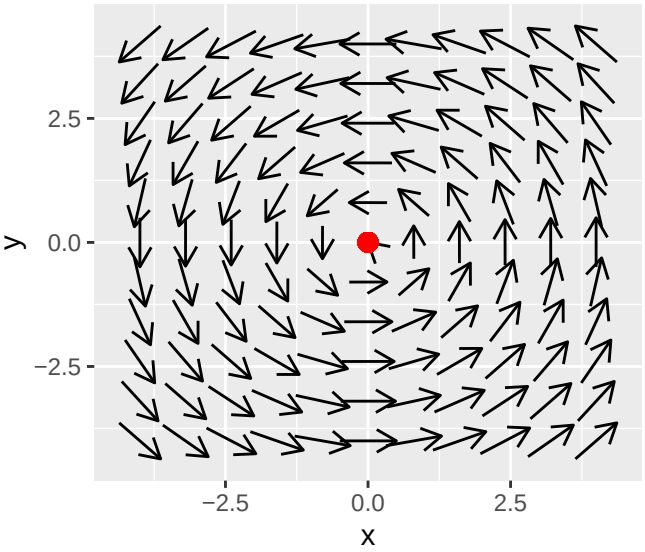
$$\vec{x}(t) = c_1 e^{\alpha t}(\vec{w}\cos(\beta t) - \vec{v}\sin(\beta t)) + c_2 e^{\alpha t}(\vec{w}\cos(\beta t) + \vec{v}\sin(\beta t)) \tag{18.6}$$

Notice the trigonometric terms in Equation (18.6) - we should expect the solution to be periodic and to have some periodic behavior to it. In fact, When $\alpha < 0$ we say the equilibrium solution is a *spiral sink* because the exponential terms in the solution decay asymptotically to zero.

As you would expect, when $\alpha > 0$ we classify a phaseplane as a spiral source, as we can see in the phase plane for $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 4 & -5 \\ 3 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$



The final case for imaginary eigenvalues is when $\alpha = 0$, which is termed a *center* equilibrium. As an example, let's examine the phaseplane for the system $\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$



Notice how the arrows don't spin in (or out here) - but seem to point in a circle.

18.3.5 Repeated eigenvalues

For repeated eigenvalues the solution is still stable or unstable depending on the sign of the eigenvalue, but rather the form of the solution changes:

$$\vec{x}(t) = (c_1 v_1 + c_2 \vec{v}_2)\, e^{\lambda t} + c_2 v_1 t e^{\lambda t} \tag{18.7}$$

18.4 Concluding thoughts

As you can see there is a lot of interesting behavior with eigenvalues and eigenvectors! But in all cases, stability really focuses on the eigenvalues and their relative (positive or negative) sign. How the straight line solutions *approach* the equilibrium solution is a function of the eigenvectors.

18.5 Exercises

Exercise 18.1. Verify that $\vec{s}_1(t) = \begin{pmatrix} 0 \\ e^t \end{pmatrix}$ is a solution to the following system of equations:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{18.8}$$

Exercise 18.2. (Inspired by Logan and Wolesensky (2009)) Compute the eigenvalues and eigenvectors for the following linear systems. Based on the eigenvalues, classify if the equilibrium solution is stable or unstable. Finally write down the most general solution for the system of equations.

a. $\frac{dx}{dt} = -x,\ \frac{dy}{dt} = -y$
b. $\frac{dx}{dt} = 3x - 2y,\ \frac{dy}{dt} = 2x - 2y$
c. $\frac{dx}{dt} = -4x + 2y,\ \frac{dy}{dt} = x - 3y$
d. $\frac{dx}{dt} = 4y,\ \frac{dy}{dt} = -9x$
e. $\frac{dx}{dt} = y,\ \frac{dy}{dt} = -x$

Exercise 18.3. Consider the following nonlinear system:

$$\begin{aligned} \frac{dx}{dt} &= y - x \\ \frac{dy}{dt} &= -y + \frac{5x^2}{4 + x^2}, \end{aligned} \tag{18.9}$$

a. Previously you verified that $(x, y) = (1, 1)$ is an equilibrium solution for this system. What is the Jacobian matrix at that equilibrium solution?
b. Generate a phaseplane for the Jacobian matrix.
c. What are the eigenvalues for the Jacobian matrix at the equilbrium solution?
d. Based on the eigenvalues, how would you classify the stability of the equilibrium solution?

Exercise 18.4. Consider the following nonlinear system:

$$\begin{aligned} \frac{dx}{dt} &= 2x + 3y + xy \\ \frac{dy}{dt} &= -x + y - 2xy^3, \end{aligned} \tag{18.10}$$

a. Previously you verified that $(x, y) = (0, 0)$ is an equilibrium solution for this system. What is the Jacobian matrix at that equilibrium solution?
b. Generate a phaseplane for the Jacobian matrix.
c. What are the eigenvalues for the Jacobian matrix at the equilbrium solution?
d. Based on the eigenvalues, how would you classify the stability of the equilibrium solution?

Exercise 18.5. Consider the following system:

$$\begin{aligned} \frac{dx}{dt} &= y^2 \\ \frac{dy}{dt} &= -\frac{2}{3}x, \end{aligned} \tag{18.11}$$

a. There is one equilibrium solution to this system of equations. What is it?
b. What is the Jacobian matrix for this equilibrium solution?
c. Generate a phaseplane for the Jacobian matrix.
d. What are the eigenvalues for the Jacobian matrix at the equilbrium solution?
e. Based on the eigenvalues, how would you classify the stability of the equilibrium solution?

Exercise 18.6. Consider the system $\frac{d}{dt}\vec{x} = A\vec{x}$.

a. Given the function $\vec{s}(t) = e^{\lambda t}\vec{v}$, where $\vec{v}$ is a constant vector, what is an expression for $\frac{d}{dt}\vec{s}(t)$?
b. Given the function $\vec{s}(t) = e^{\lambda t}\vec{v}$, where $\vec{v}$ is a constant vector, what is an expression for $A\vec{s}(t)$?
c. Now use the previous results to compare $\frac{d}{dt}\vec{s}(t) = A\vec{s}(t)$. Explain why it must be the case that $\lambda\vec{v} = A\vec{v}$

Exercise 18.7. In this section we learned that for a two dimensional matrix $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$, eigenvalues can be found by solving the characteristic equation $\det(A - \lambda I) = 0$, or $\lambda^2 - (a+d)\lambda + ad - bc = 0$. Use the quadratic formula to get an expression for the eigenvalues λ in terms of a,b, c, and d.

Chapter 19

Qualitative Stability Analysis

In the previous sections we studied the connections between linear systems, the phaseplane, and how eigenvalues help determine the stability of an equilibrium solution. You also learned with a non-linear system of differential equations you can apply a locally linear approximation via the Jacobian to determine the stability of an equilibrium solution.

Computing eigenvalues can be tricky - but for a two-dimensional system of DEs some interesting results occur because the characteristic equation ends up being a quadratic function. Because of this we can analyze stability for a two-dimensional system of equations without computing eigenvalues, but first we will need to understand some key relationships.

19.1 Two dimensional linear systems: the general case

Consider the following two dimensional linear system, where a, b, c, and d can be any number:

$$\begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{19.1}$$

Recall that eigenvalues are found by solving $\det(A - \lambda I) = 0$, which $\det(A - \lambda I)$ is computed in Equation (19.2).

$$\det \begin{pmatrix} a - \lambda & b \\ c & d - \lambda \end{pmatrix} = (a - \lambda)(d - \lambda) - bc \tag{19.2}$$

If we multiply out Equation (19.2) we obtain the characteristic equation:

$$\lambda^2 - (a + d)\lambda + ad - bc = 0 \tag{19.3}$$

What is cool about Equation (19.3) is that the roots can be expressed as functions of the entries of the matrix A (a, b, c, d). In fact, in linear algebra the term $a + d$ is the sum of the diagonal entries, also known as the **trace** of a matrix. The trace of a matrix can be expressed as $\text{tr}(A)$. And you may recognize that $ad - bc$ is the same as $\det(A)$. So our characteristic equation is can be rewritten as solving Equation (19.4):

$$\lambda^2 - \text{tr}(A)\lambda + \det(A) = 0 \tag{19.4}$$

Example 19.1. Write down the characteristic equation for the system $\vec{x}' = Ax$ where $A = \begin{pmatrix} -1 & 1 \\ 0 & 3 \end{pmatrix}$ and classify the stability of the equilibrium solution.

Solution. We can see that $\det(A) = -1(3) - 0(1) = -3$ and $\text{tr}(A) = 2$, so our characteristic equation is $\lambda^2 - 2\lambda - 3$. If we solve $\lambda^2 - 2\lambda - 3 = 0$ we have $(\lambda - 3)(\lambda + 1) = 0$, so our eigenvalues are $\lambda = 3$ and $\lambda = -1$. Since one eigenvalue is positive and the other one is negative, the equilibrium solution is a saddle node.

Solving Equation (19.4) might be a computationally easier way to compute the eigenvalues, especially if you have a known parameter. We can also exploit this relationship even more.

Let's say we have two eigenvalues λ_1 and λ_2. We make no assumptions on if they are real or imaginary or equal. But if they are eigenvalues, then they are roots of the characteristic polynomial. This means that $(\lambda - \lambda_1)(\lambda - \lambda_2) = 0$. If we multiply out this equation we have $\lambda^2 - (\lambda_1 + \lambda_2)\lambda + \lambda_1\lambda_2 = 0$. Hmmm. If we compare this equation with Equation (19.4) we have:

$$\lambda^2 - (\lambda_1 + \lambda_2)\lambda + \lambda_1\lambda_2 = \lambda^2 - \text{tr}(A)\lambda + \det(A) \tag{19.5}$$

This uncovers some neat relationships - in particular $\text{tr}(A) = (\lambda_1 + \lambda_2)$ and $\det(A) = \lambda_1 + \lambda_2$. Why should we bother with this? Well this provides an alternative pathway to understand stability through the trace and determinant, in particular we have the following correspondence between the signs of the eigenvalues and the trace and determinant:

Sign of λ_1	Sign of λ_2	Tendency of solution	Sign of $\text{tr}(A)$	Sign of $\det(A)$
Positive	Positive	Source	Positive	Positive
Negative	Negative	Sink	Negative	Positive
Positive	Negative	Saddle	?	Negative
Negative	Positive	Saddle	?	Negative

For the moment we will only consider real non-zero values of the eigenvalues - more specialized cases will occur later. But carefully at the table:

- If the determinant is *negative*, then the equilibrium solution is a *saddle*.
- If the determinant is *positive* and the trace is *negative*, then the equilibrium solution is a *sink*
- If the determinant and trace are both *positive*, then the equilibrium solution is a *source*.

Example 19.2. Use the trace and determinant relationships to classify the stability of the equilibrium solution for the linear system $\vec{x}' = A\vec{x}$ where $A = \begin{pmatrix} -1 & 1 \\ 0 & 3 \end{pmatrix}$.

Solution. We can see that $\det(A) = -1(3) - 0(1) = -3$ and $\text{tr}(A) = 2$. Since the determinant is negative, the equilibrium solution must be a saddle node.

Knowing the relationships between the trace and determinant for a two-dimensional system of equations is a pretty quick and easy way to investigate stability of equilibrium solutions!

Another way to graphically represent the stability of solutions is with the *trace-determinant plane* (shown in Figure 19.1), with $\text{tr}(A)$ on the horizontal axis and $\det(A)$ on the vertical axis:

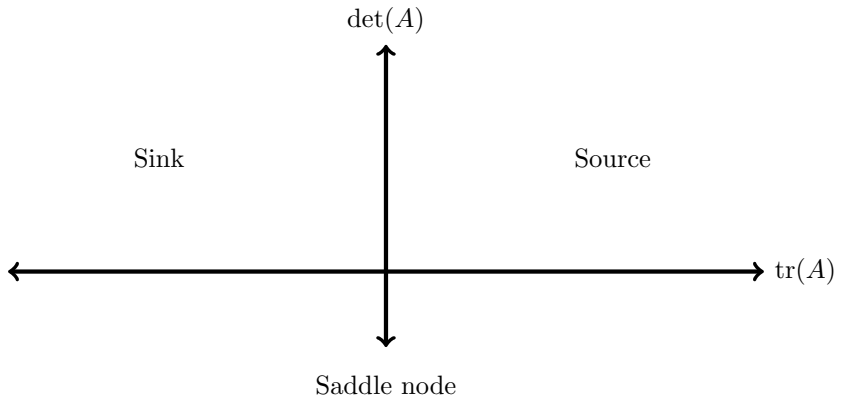


Figure 19.1: The trace-determinant plane

It also turns out that we can extend Figure 19.1 even more to include the imaginary and nonzero cases for the roots of the characteristic equation.

Firt we will apply the quadratic formula to Equation (19.4) to solve directly for the eigenvalues as a function of the trace and determinant:

$$\lambda_{1,2} = \frac{\text{tr}(A)}{2} \pm \frac{\sqrt{(\text{tr}(A))^2 - 4\det(A)}}{2} \tag{19.6}$$

While Equation (19.6) seems like a more complicated expression, it can be shown to be consistent with our above work. Imaginary eigenvalues can be a stable or unstable spiral depending on its location in the trace-determinant plane (Figure 19.2).

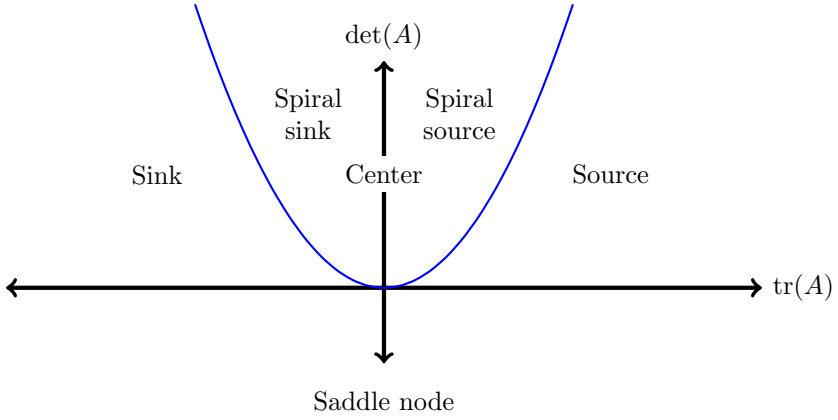


Figure 19.2: Revised trace-determinant plane.

If the discriminant (the part inside the square root for Equation (19.6), which is $(\text{tr}(A))^2 - 4\det(A)$) of the eigenvalue expression is negative then we have a spiral source or spiral sink depending on the positivity of the $\text{tr}(A)$.

Finally the *center* equilibrium occurs when the trace is exactly zero and the determinant is positive. This graphic of the trace-determinant plane is a quick way to analyze stability of a solution without a lot of algebraic analysis.

19.2 Sensitivity to parameters with the trace-determinant

Figure 19.2 is even more useful if your system consists of parameters that aren't specified. Consider the following lynx - hare system:

$$\begin{aligned} \frac{dH}{dt} &= rH - bHL \\ \frac{dL}{dt} &= ebHL - dL \end{aligned} \tag{19.7}$$

We have studied this system several times. We know that the steady states of this system are $(H, L) = (0, 0)$ and $\left(\frac{d}{eb}, \frac{r}{b}\right)$ in this model. Let's compute the Jacobian at each of these systems:

$$J_{(0,0)} = \begin{pmatrix} r & 0 \\ 0 & -d \end{pmatrix} \tag{19.8}$$

$$J_{(\frac{d}{eb}, \frac{r}{b})} = \begin{pmatrix} r - \frac{d}{e} & -\frac{d}{e} \\ er & 0 \end{pmatrix} \tag{19.9}$$

Let's analyze the Jacobian at each of the the equilibrium solutions.

- $(0, 0)$ equilibrium solution: We have $\text{tr}(J) = r - d$ and $\det(J) = -rd$. Since r and d are both positive parameters the determinant will always be negative, so no matter what the origin will be a saddle.

- $(\frac{d}{eb}, \frac{r}{b})$ equilibrium solution: We have $\text{tr}(J) = r - \frac{d}{e}$ and $\det(J) = d$. The determinant is positive, however we can see that the stability of this equilibrium solution will be dependent on the trace. If the trace is positive, or $r > \frac{d}{e}$ than the equilibrium solution will be unstable. If $r < \frac{d}{e}$ it will be stable. You can derive even stronger boundaries between a spiral source or spiral sink as well.

Notice how these relationships give you a sense for what you could expect in terms of a solution even before computing eigenvalues!

19.3 Higher dimensional stability

The trace-determinant plane is a really use approach to analyze stability without computing the eigenvalues directly. It is also useful when you have a parameter in your Jacobian - the trace-determinant will allow you to characterize the stability of solutions as a function of a parameter. This leads to what is known as a bifurcation diagram, which we will study in the next section.

The conditions between the trace and determinant really describe the notion of eigenvalues as a function of the entries of a matrix. It may be natural to ask if similar conditions exist for larger dimensioned matrices. It turns out yes ... to a point. The Routh-Hurwitz stability criterion is one such approach, but it gets tricky for larger matrices.

19.4 Exercises

Exercise 19.1. (Inspired by Logan and Wolesensky (2009)) Compute the trace and determinant for each of these linear systems. Use the trace-determinant condition to classify the stability of the equilibrium solutions. Verify your stability results are consistent when analyzing stability from calculating the eigenvalues.

a. $\frac{dx}{dt} = -x,\ \frac{dy}{dt} = -2y$
b. $\frac{dx}{dt} = 3x + y,\ \frac{dy}{dt} = 2x + 4y$
c. $\frac{dx}{dt} = 8x - 11y,\ \frac{dy}{dt} = 6x - 9y$
d. $\frac{dx}{dt} = 3x - y,\ \frac{dy}{dt} = 3y$
e. $\frac{dx}{dt} = -2x - 3y,\ \frac{dy}{dt} = 3x - 2y$

Exercise 19.2. (Inspired by Logan and Wolesensky (2009)) The following equation can be applied to study cell differentiation:

$$\begin{aligned} \frac{dx}{dt} &= y - x \\ \frac{dy}{dt} &= -y + \frac{5x^2}{4 + x^2} \end{aligned} \tag{19.10}$$

a. Previously you verified that $(x, y) = (1, 1)$ is an equilibrium solution for this system. What is the Jacobian matrix at that equilibrium solution?
b. What is $\text{tr}(J)$ and $\det(J)$ at that equilibrium soltion?
c. Evaluate the stability of the equilibrium solution solution using relationships between the trace and determinant.

Exercise 19.3. (Inspired by Logan and Wolesensky (2009)) Consider the following predator-prey model, where the carrying capacity of the predator (y) depends on the prey population (x):

$$\begin{aligned} x' &= \frac{2}{3}x \cdot \left(1 - \frac{x}{4}\right) - \frac{1}{6}xy \\ y' &= 0.5y \cdot \left(1 - \frac{y}{x}\right), \end{aligned} \tag{19.11}$$

a. There are two equilibrium solutions for this differential equation. What are they? *Hint:* first determine where $y' = 0$ and then substitute your solutions into $x' = 0$.
b. Use the command *phaseplane* to visualize this system of equations.
c. Compute the Jacobian matrix for both equilibrium solutions.
d. Use the trace-determinant relationships to evaluate the stability of the equilibrium solutions. Is that analysis consistent with your phaseplane?

Exercise 19.4. (Inspired by Logan and Wolesensky (2009)) Consider the linear system of equations:

$$\begin{aligned} \frac{dx}{dt} &= -ax - y \\ \frac{dy}{dt} &= -x - ay \end{aligned} \tag{19.12}$$

Apply the relationships between the trace and determinant to classify the stability of the equilibrium solution for different values of a. Be sure to include cases where the system will be a spiral source or sink.

Exercise 19.5. All of the following systems have an equilibrium solution at the origin (0,0). Compute the Jacobian of these solutions and apply the trace and determinant conditions to analyze the stability. The stability will be a function of the parameter μ. In your stability analysis you only need to classify differences between a source, sink, or saddle.

a.

$$\begin{aligned} x' &= x + \mu y \\ y' &= \mu x - y, \end{aligned} \tag{19.13}$$

b.

$$\begin{aligned} x' &= x + y \\ y' &= \mu x + y, \end{aligned} \tag{19.14}$$

c.

$$\begin{aligned} x' &= y \\ y' &= x^2 - x + \mu y, \end{aligned} \tag{19.15}$$

Exercise 19.6. For the lynx-hare system equilibrium $\left(\frac{d}{eb}, \frac{r}{b}\right)$, determine conditions on the parameters where the system will be a spiral source and sprial sink.

Exercise 19.7. (Inspired by Logan and Wolesensky (2009)) Let C be the amount of carbon in a forest ecosystem, with P be the rate of increase due to photosynthesis. Herbivores H consume carbon on the following predator-prey model:

$$\begin{aligned} \frac{dC}{dt} &= P - aC - bHC \\ \frac{dH}{dt} &= ebHC - dC \end{aligned} \tag{19.16}$$

a. Equation (19.16) has two equilibrium solutions. What are they?
b. Evaluate the Jacobian at each of the equilibrium solutions.
c. Evaluate the stability of each equilibrium solution using relationships between the trace and determinant. Be sure to include cases where the system will be a spiral source or sink.

Exercise 19.8. Apply the quadratic formula to $\lambda^2 - \text{tr}(A)\lambda + \det(A) = 0$ obtain Equation (19.6).

Exercise 19.9. Assume that you have two complex conjugate eigenvalues: $\lambda_1 = a \pm bi$ and $\lambda_2 = a - bi$.

a. What is an expression for $\lambda_1 + \lambda_2$?
b. What is an expression for $\lambda_1 \cdot \lambda_2$?
c. Use your answers from the previous two results to show that $\text{tr}(A) = 2a$ and $\det(A) = a^2 + b^2$.
d. If you were to have a system that is a spiral sink, what conditions on a and b need to hold?
e. Create a linear two-dimensional system where the equilibrium solution is a spiral sink. Show your system and the corresponding phaseplane.

Exercise 19.10. Consider a two-dimensional system where $\text{tr}(A) = 0$ and $\det(A) > 0$.

a. Given those conditions, explain why $\lambda_1 + \lambda_2 = 0$ and $\lambda_1 \cdot \lambda_2 > 0$.
b. What does $\lambda_1 + \lambda_2 = 0$ tell you about the relationship between λ_1 and λ_2?
c. What does $\lambda_1 \cdot \lambda_2 > 0$ tell you about the relationship between λ_1 and λ_2?
d. Look back to your previous two responses. First explain why λ_1 and λ_2 must be imaginary eigenvalues (in other words, not real values). Then explain why $\lambda_{1,2} = \pm bi$.
e. Given these constraints, what would the phaseplane for this system be?
f. Create a linear two-dimensional system where $\text{tr}(A) = 0$ and $\det(A) > 0$. Show your system and the phaseplane.

Chapter 20

Bifurcation

In this section we will study a key topic that applies qualititative stability analysis from Section 19 to examine how stability of an equilibrium solution is sensitivity to parameters. This general topic is called *bifurcation*, and to illustrate what bifurcation is we will examine several examples.

20.1 A series of equations

First let's consider the differential equation $\frac{dx}{dt} = 1 - x^2$. This equation has an equibrium solution at $x = \pm 1$. To classify the stability of the equilibrium solutions we apply the following test (developed in Section 5):

- If $f'(y_*)' > 0$ at an equilibrium solution, the equilibrium solution $y = y_*$ will be unstable.
- If $f'(y_*) < 0$ at an equilibrium solution, the equilibrium solution $y = y_*$ will be stable.
- If $f'(y_*) = 0$, we cannot conclude anything about the stability of $y = y_*$.

Applying this test, we know $f(x) = 1 - x^2$ and $f'(x) = -2x$. Since $f'(1) = -2$ and $f'(-1) = 2$, then the equilibrium solution $x = 1$ is stable and $x = -1$ unstable.

Figure 20.1 shows a plot of the phase plane along with some solutions. Notice how the solutions seem to flow to $x = 1$ and away from $x = -1$.

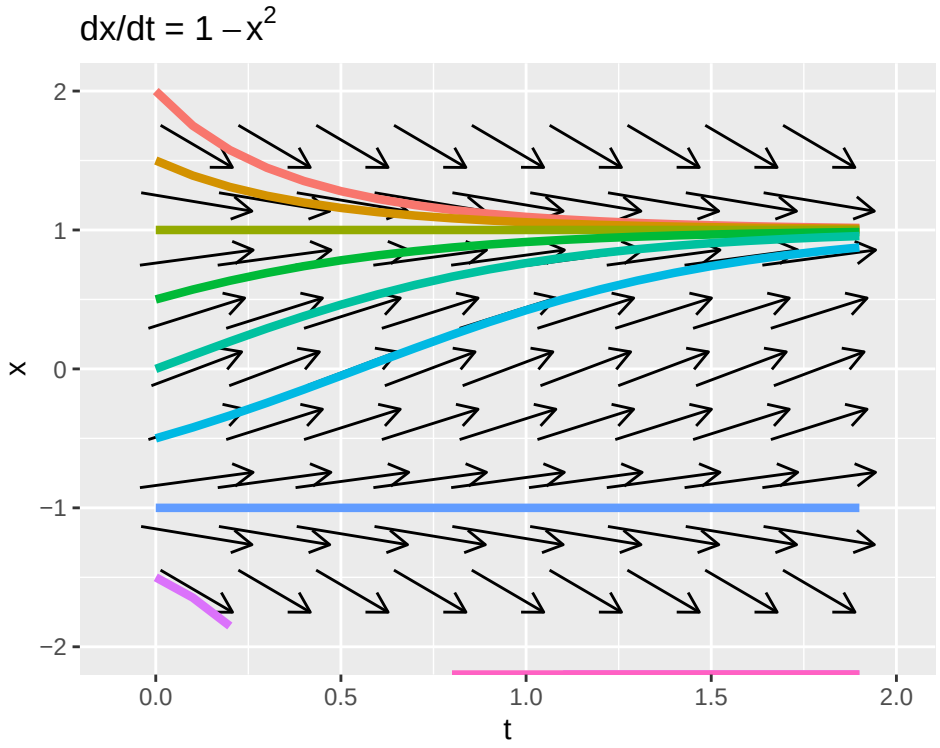
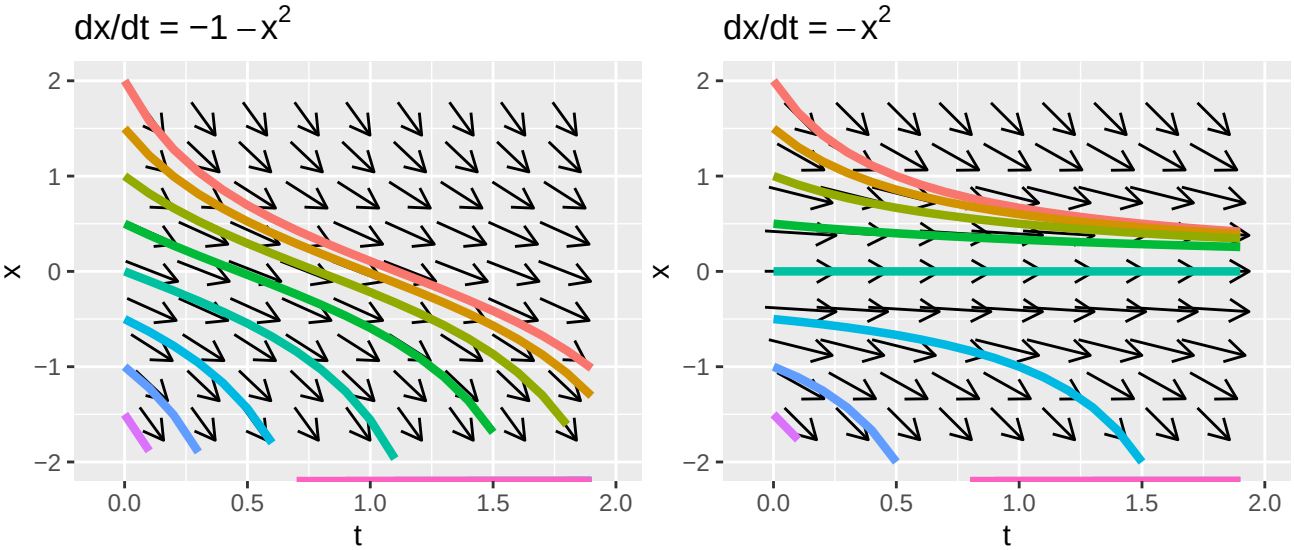


Figure 20.1: Phase plane of $x' = 1 - x^2$

Let's modify and extend this example further. Consider two more differential equations:

- $\frac{dx}{dt} = -1 - x^2$: This differential equation does not have any equilibrium solutions, so we do not need to apply the stability test.
- $\frac{dx}{dt} = -x^2$: The stability of the second equation is a little more tricky. While it has an equilibrium solution at $x = 0$, the stability test cannot apply because $f' = -2x$ and $f'(0) = 0$. However investigating the sign of $f(x)$ to the left and right of $x = 0$, we see that this solution is constantly decreasing, so the equilibrium is unstable.

So let's also take a look at the phaseplanes for $\frac{dx}{dt} = -1 - x^2$ and $\frac{dx}{dt} = -x^2$ along with the some solution curves (see also the accompanying figure).



There is an interesting pattern going on here. Let's build on these three examples in a more general context. Consider the differential equation $\frac{dx}{dt} = c - x^2$. In our first example, $c = 1$. In the second example, $c = -1$, and the third example $c = 0$.

The phaseplanes from these examples illustrates how the value of c influences the phase line and the resulting solution.

Steady states to $\frac{dx}{dt} = c - x^2$ are at $x^* = \pm\sqrt{c}$. We can also test out the stability of our steady states using the stability test, with $f(x) = c - x^2$ and $f'(x) = -2x$:

- If $c > 0$ we have two steady states, summarized in the following table:

Equilibrium solution	$f'(x^*)$	Tendency of solution
$x^* = \sqrt{c}$	$-2\sqrt{c}$	Stable
$x^* = -\sqrt{c}$	$2\sqrt{c}$	Unstable

(You should verify that the stability result we initially found when $c = 1$ matches the table.)

- If $c = 0$ there is only one steady state, summarized, in the following table:

Equilibrium solution	$f'(x^*)$	Tendency of solution
$x^* = 0$	0	Inconclusive

Based on the phaseplane for $x' = -x^2$, the equilibrium solution $x^* = 0$ is unstable. If the initial condition $x(0)$ is greater than 0, the solution flows towards $x = 0$, but if the initial condition is less than zero, the solution flows away from $x = 0$. This is similar to a one-dimensional analogue to a saddle node from Section 19.

- Finally, if $c < 0$, then there are no steady states because $\sqrt{c}$ is does not have a real solution when $c < 0$. (Note: eigenvalues can be imaginary, but equilibrium solutions for our contexts can only be real). We can also test out the stability of our steady states using the stability test, with $f(x) = c - x^2$ and $f'(x) = -2x$:

Notice how different values of c influence both the *value* ($x^* = \pm\sqrt{c}$) and the *stability* of the equilibrium solution. Rather than making a series of tables, we can represent the dependence of the equilibrium solution and its stability in what is called a *bifurcation diagram* (Figure 20.2).

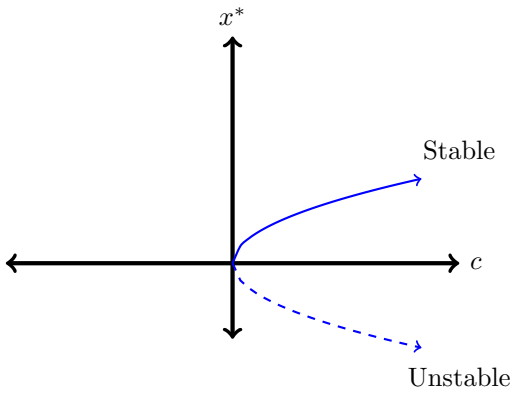


Figure 20.2: A saddle-node bifurcation

Let's talk about Figure 20.2. The graph represents the value of the equilibrium solution (x^*, vertical axis) as a function of the parameter c (horizontal axis). Since equilibrium solutions are characterized by $x^* = \pm\sqrt{c}$ we have the "sideways parabola," traced in blue in Figure 20.2. When $c < 0$, there is no equilibrium solution, (so nothing is plotted in the second and third quadrants of Figure 20.2). The difference between the solid and dashed lines in Figure 20.2 is used to distinguish between a stable equilibrium solution ($x^* = \sqrt{c}$ when $c > 0$) and unstable equilibrium solution ($x^* = -\sqrt{c}$ when $c > 0$). That is so cool that *all* the information about the equilibrium solution and its stability is contained in this diagram!

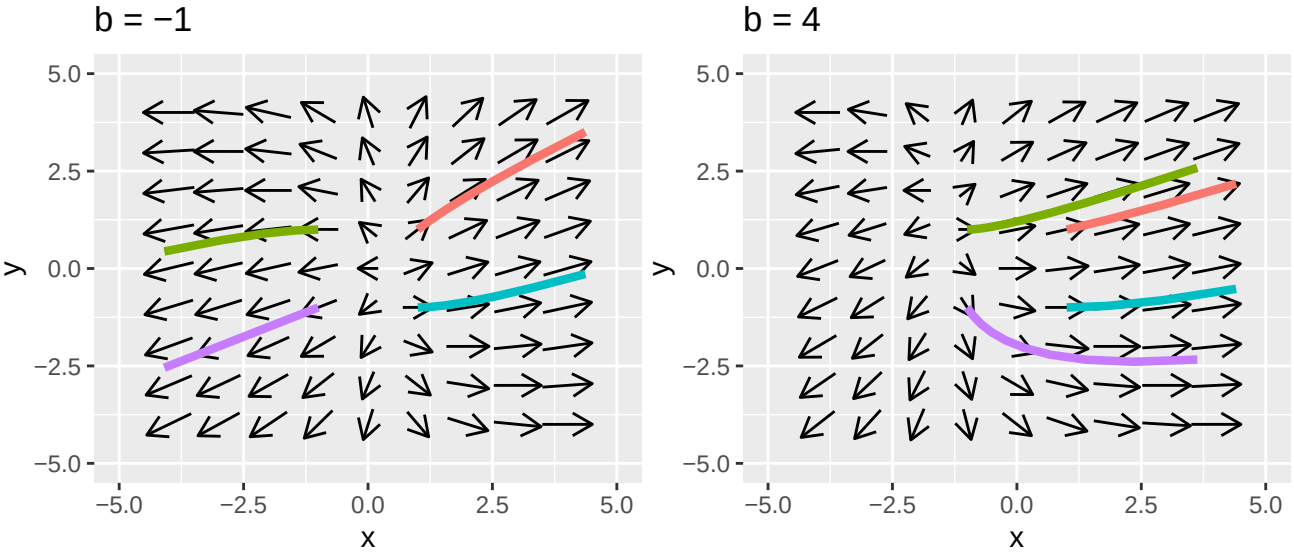
This particular example is called a *saddle-node* bifurcation. It might be helpful to think of this c like a tuning knob. As $c > 0$ we will always have two different equilibrium solutions that are symmetrical based on the value of c. The positive one will be stable, the other unstable. As c gets smaller these equilbrium solutions will collapse into one, which will be unstable. If c is negative, the equilibrium solution disappears.

20.2 Bifurcations with systems of equations

As another example, let's determine the behavior of solutions near the origin for the system of equations:

$$\frac{d\vec{x}}{dt} = \begin{pmatrix} 3 & b \\ 1 & 1 \end{pmatrix} \vec{x}. \tag{20.1}$$

Let's take a look at the phase plane with solution curves.



This equation has one free parameter b that we will analyze using the trace determinant conditions developed in Section 19. Let's call the matrix A, so the $\text{tr}(A) = 4$ and $\det(A) = 3 - b$. Since the trace is always positive either it will be a saddle if the $\det(A) < 0$, or when $3 < b$. We have a unstable spiral when $3 > b$ and $(\text{tr}(A))^2 - 4\det(A) < 0$, or when $4^2 - 4 \cdot (3 - b) = 16 - 12 + 4b = 4 + 4b < 0$, which leads to $b < -1$. Notice this is a contradictory condition - we have already assumed $b > 3$, so we will not have any unstable spirals.

To summarize, we have the following dynamics:

- When $b < 3$ the equilibrium solution will be a saddle.
- When $b = 3$ we will have an unstable solution.
- When $b > 3$ we will have a unstable node.

The benefit of a bifurcation diagram is that is provides a complete understanding of the dynamics of the system *as a function of the parameters*. In this section we examined *one-parameter* bifurcations (for example we looked the stability of the equilibrium solution as it depends on c or b), but bifurcations can also be extended further to two parameter bifurcation families, applying similar methods. In general the methods are similar to what we have done.

20.3 Limit Cycles and Bifurcations with systems of equations

The previous examples we have studied examine the stability of an equilibrium solution as a parameter changes. An extension of an equilibrium solution (as a point in space) is an equilibrium solution that is a *function*.

Consider the following highly nonlinear system:

$$\begin{aligned} \frac{dx}{dt} &= -y - x(x^2 + y^2 - 1) \\ \frac{dy}{dt} &= x - y(x^2 + y^2 - 1) \end{aligned} \tag{20.2}$$

The phase plane for Equation (20.2) is shown in Figure 20.3. You can verify that Equation (20.2) system has an equilibrium solution at the point $x = 0$, $y = 0$.

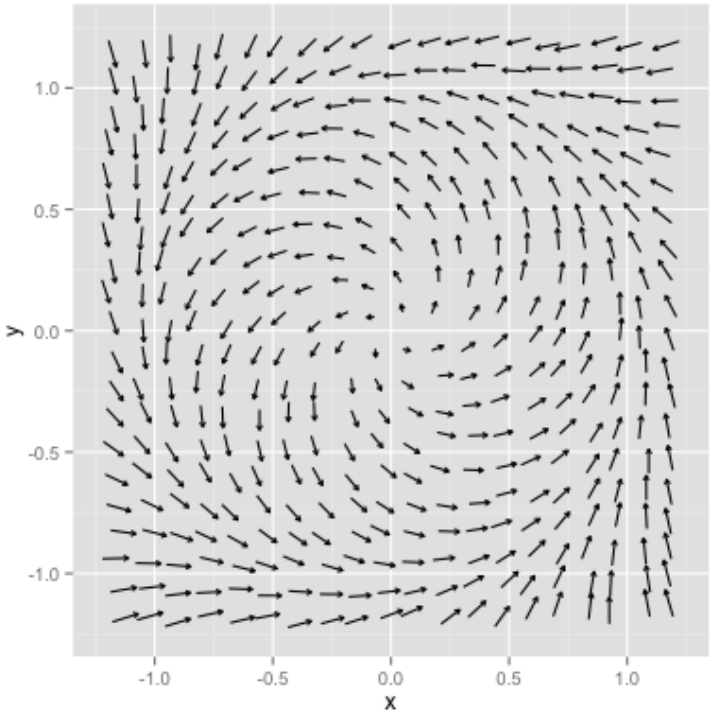


Figure 20.3: Phase plane for Equation ??eq:limit-cycle-20)

However the phase plane suggests there might be other equilibrium solutions. To further explore this, let's look at the phase plane with some solution curves in Figure 20.4:

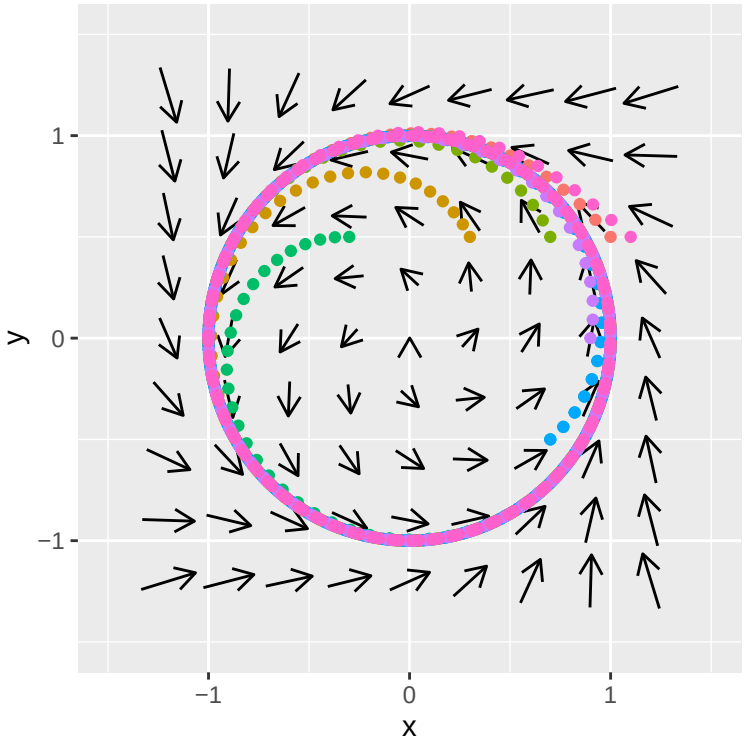


Figure 20.4: Phase plane with solutions curves for Equation (20.2)

What is interesting in Figure 20.4 that the solution is tending towards a circle of radius 1 (or the equation $x^2 + y^2 = 1$). This

is an example of an equilibrium solution that is a *curve* rather than a specific point. We can transform this system from x and y to a single new variables X (see Exercise 20.6).

$$\frac{dX}{dt} = -X(X-1), \text{ where } X = r^2. \tag{20.3}$$

How Equation (20.2) transforms to Equation (20.3) is by applying a polar coordinate transformation to this system. By applying stability analysis for Equation (20.3) we can show that the equilibrium solution $X = 0$ is unstable (meaning the origin $x = 0$ and $y = 0$ is an unstable solution) and the circle of radius 1 is stable (which is the equation $x^2 + y^2 = 1$). In this case we would say $r = 1$ is a *stable limit cycle*. You will study a similar system in Exercises 20.6 and 20.7.

Equation (20.3) is an example of next steps with studying the qualitative analysis of systems. While we have focused on bifurcation of equilibrium solutions, hopefully this brief introduction will pique your interest in further study of bifurcations.

The most important part in studying bifurcations is analyzing examples. I've provided a lot of exercises where you will construct bifurcation diagrams for one and two-dimensional equations. Given an equation with a parameter, it is helpful to construct some sample phaselines / phaseplanes before analyzing stability in general using the stability test or the trace-determinant plane.

20.4 Exercises

Exercise 20.1. Apply local linearization to classify stability of the following differential equations:

a. $\frac{dx}{dt} = x - x^2$
b. $\frac{dx}{dt} = -x^2$
c. $\frac{dx}{dt} = -x - x^2$

Exercise 20.2. Consider the differential equation $\frac{dx}{dt} = cx - x^2$. What are equations that describe the dependence of the equilibrium solution on the value of c? Once you have that figured out plot the bifurcation diagram, with the parameter c along the horizontal axis. This bifurcation is called a *transcritical* bifurcation.

Exercise 20.3. Consider the differential equation $\frac{dx}{dt} = cx - x^3$. What are equations that describe the dependence of the equilibrium solution on the value of c? Once you have that figured out plot the bifurcation diagram, with the parameter c along the horizontal axis. This bifurcation is called a *pitchfork* bifurcation.

Exercise 20.4. (Inspired by Logan and Wolesensky (2009)) Through constructing a bifurcation diagram, determine the behavior of solutions near the origin for the following system:

$$\frac{\vec{x}}{dt} = \begin{pmatrix} 3 & b \\ b & 1 \end{pmatrix} \vec{x}. \tag{20.4}$$

Exercise 20.5. (Inspired by Logan and Wolesensky (2009)) Consider the linear system of equations:

$$\begin{aligned} \frac{dx}{dt} &= -ax - y \\ \frac{dy}{dt} &= -x - ay \end{aligned} \tag{20.5}$$

Construct a bifurcation diagram for this system of equations.

Exercise 20.6. Consider the following highly nonlinear system:

$$\begin{aligned} \frac{dx}{dt} &= -y - x(x^2 + y^2 - 1) \\ \frac{dy}{dt} &= x - y(x^2 + y^2 - 1) \end{aligned} \tag{20.6}$$

We are going to transform the system by defining new variables $x = r\cos\theta$ and $y = r\sin\theta$. Observe that $r^2 = x^2 + y^2$.

a. Consider the equation $r^2 = x^2 + y^2$, where r, x, and y are all functions of time. Apply implicit differentiation to determine a differential equation for $\frac{d(r^2)}{dt}$, expressed in terms of x, y, $\frac{dx}{dt}$ and $\frac{dy}{dt}$.
b. Multiply $\frac{dx}{dt}$ by $2x$ and $\frac{dy}{dt}$ by $2y$ on both sides of the equation. Then add the two equations together. You should get an expression for $\frac{d(r^2)}{dt}$ in terms of x and y.
c. Re-write the equation for the right hand side of $\frac{d(r^2)}{dt}$ in terms of r^2.

d. Use your equation that you found to verify that

$$\frac{dX}{dt} = -X(X-1), \text{ where } X = r^2 \tag{20.7}$$

e. Verify that $X = 1$ is a stable node and $X = 0$ is unstable.
f. As discussed in this section this system has a stable limit cycle. What quick and easy modification to our system could you do to the system to ensure that this is a unstable limit cycle? Justify your work.

Exercise 20.7. Construct a bifurcation diagram for $\frac{dX}{dt} = -X(X-\mu)$, μ is a parameter. Explain how you can apply that result to understanding the bifurcation diagram of the system:

$$\begin{aligned} \frac{dx}{dt} &= -y - x(x^2+y^2-\mu) \\ \frac{dy}{dt} &= x - y(x^2+y^2-\mu) \end{aligned} \tag{20.8}$$

This system is an example of a *Hopf bifurcation*.

Exercise 20.8. (Inspired by Logan and Wolesensky (2009)) Consider following predator-prey model:

$$\begin{aligned} \frac{dx}{dt} &= \frac{2}{3}x\left(1-\frac{x}{4}\right) - \frac{xy}{1+x} \\ \frac{dy}{dt} &= ry\left(1-\frac{y}{x}\right) \end{aligned} \tag{20.9}$$

a. Explain the various terms in this model and their biological meaning.
b. Determine the equilibrium solutions.
c. Evaluate the Jacobian at each of the equilibrium solutions.
d. Construct a bifurcation diagram (with the parameter r) for each of the equilibrium solutions.

Exercise 20.9. (Inspired by Logan and Wolesensky (2009)) The immune response to HIV has been described with differential equations. In the early stages (before the body is swamped by the HIV virions) the dynamics of the virus can be described by the following system of equations, where v is the virus load and x the immune response:

$$\begin{aligned} \frac{dv}{dt} &= rv - pxv \\ \frac{dx}{dt} &= cv - bx \end{aligned} \tag{20.10}$$

a. Explain the various terms in this model and their biological meaning.
b. Determine the equilibrium solutions.
c. Evaluate the Jacobian at each of the equilibrium solutions.
d. Construct a bifurcation diagram for each of the equilibrium solutions.

Part IV

Stochastic Differential Equations

Chapter 21

Stochastic Biological Systems

So far in this course we have studied *deterministic* differential equations. We use the word deterministic because given an initial condition and parameters, the solution trajectory is determined. Now we are going to study *stochastic* differential equations (or SDEs for short) - which means that the differential equation is subject to randomness - either in the parameters (which may cause a change in the stability for a time) or in the variables themselves. Hopefully the previous sections in the textbook imparted a sense (1) that parameter values may change according to the data used to estimate them (parameter estimation) and (2) changing a parameter may drastically alter the long-term dynamics of a system (bifurcation). Learning about stochastic differential equations will extend your knowledge.

Stochastic differential equations are a neat field of study that can be studied using computational approaches. We will give you an introduction to SDEs with some focus on solution techniques. The ideas we will learn can be applied in other contexts. For example, Brownian motion is the foundation to modern physics or can be applied to modeling changes in the stock market.

Understanding how to model SDEs requires learning some new mathematics as well as ways to numerically simulate this mathematics. A lot of SDEs rely on numerical simulation, so we will build up our understanding of some of the results of numerical simulation first.

21.1 A discrete system

Let's focus on an example that involves discrete populations. Moose are large animals (part of the deer family) weighing 1000 pounds that can be found in Northern Minnesota link. Here is a picture of a moose: link. While in the early 2000's the population of the moose was 8000, recent estimates have the numbers at 3000, although that seems to have stabilized link

A starting model that describes their population dynamics is the discrete dynamical system:

$$M_{t+1} = M_t + b \cdot M_t - d \cdot M_t, \tag{21.1}$$

where M_t is the population of the moose in year t, and b the birth rate and d the death rate. This equation can be reduced down to $M_{t+1} = rM_t$ where $r = 1 + b - d$ is the net birth/death rate. This model states that the population of moose in the next year equals the current population, added to any fraction of births and taking away any deaths.

This discrete dynamical system is a little bit different from a continuous dynamical system, but can be simulated pretty easily by defining a function.

```
M0 <- 3000 # Initial population of moose
N <- 5 # Number of years we simulate

moose <- function(r) {
  out_moose <- array(M0, dim = N)
  for (i in 1:(N - 1)) {
    out_moose[i + 1] <- r * out_moose[i]
  }
  return(out_moose)
}
```

Notice how the function `moose` returns the current population of moose after N years with the net birth rate r. Let's take a look at the results for different values of r (Figure 21.1).

```
moose_rates <- tibble(
  years = 1:N,
  r0.4 = moose(0.4),
  r0.8 = moose(0.8),
  r1.1 = moose(1.1)
)


ggplot(data = moose_rates) +
  geom_line(aes(x = years, y = r0.4), color = "red") +
  geom_line(aes(x = years, y = r0.8), color = "blue") +
  geom_line(aes(x = years, y = r1.1), color = "green") +
  labs(
    x = "Years",
    y = "Moose"
  )
```

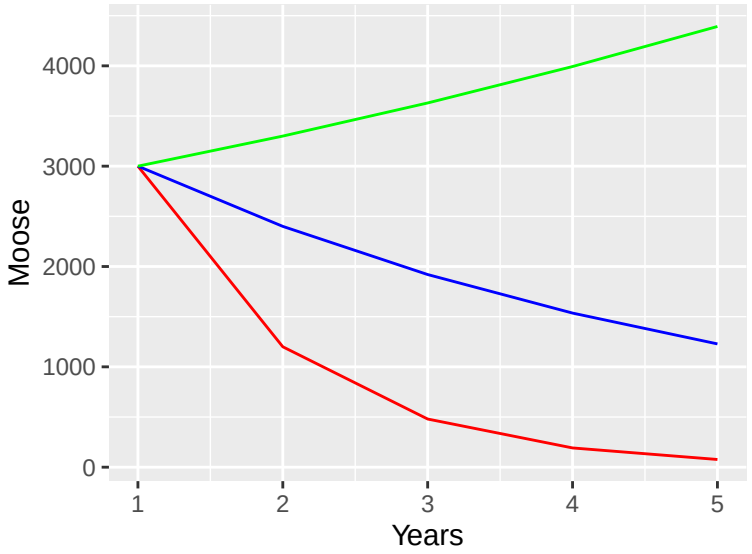


Figure 21.1: Simulation of the moose population with different birth rates.

Let's remind ourselves of what is going on in the code.

- `out_moose <- tibble...` creates a data frame that we can use for plotting. We call several different instances of the `moose` code for different net birth rates. I've decided to label each of those instances with the value of the birth rate `r` for reference.
- The command `geom_line` makes a line plot. Remember that we need to specify both the x (horizontal) and y (vertical) axes. I specified the different colors to distinguish the different birth rates.

Notice how for some values of r the population starts to decline, stay the same, or increase. Let's analyze this system a little more. Just like with a continuous differential equation we want to look for solutions that are in steady state, or ones were the population is staying the same. In other words this means that $M_{t+1} = M_t$, or $M_t = rM_t$. If we simplify this expression this means that $M_t - rM_t = 0$, or $(r-1)M_t = 0$. Assuming that M_t is not equal to zero, then this equation is consistent only when $r = 1$. This makes sense: we know $r = 1 - b - d$, so the only way this can be one is if $b = d$, or the births balance out the deaths.

Ok, so we found our equilibrium solution. However what is the general form of this solution to Equation (21.1)? Just like when we were solving continuous differential equations (Section 7 a starting point was that the solution was exponential. For discrete systems we will do the same, but this time we represent the solution as $M_t = A \cdot v^t$, which is an exponential equation. Since we have $M_0 = 3000$, then $A = 3000$. Using these values of M_0 and A for Equation (21.1) we obtain the following:

$$100 \cdot v^{t+1} = r3000 \cdot v^t \tag{21.2}$$

Our goal is to figure out a value for v that is consistent with this expression. Just like we did with continuous differential equations we can arrange the following equation, using the fact that $v^{t+1} = v^t \cdot v$:

$$3000v^t(v - r) = 0 \tag{21.3}$$

Since we assume $v \neq 0$, the only possibility is if $v = r$. Aha! Our general solution then is

$$M_t = 3000r^t \tag{21.4}$$

We know that if $r > 1$ we have exponential growth exponential decay when $r < 1$ exponential decay, consistent with our results above.

There is some comfort here: just like in continuous systems we find the eigenvalues that determine the stability of the equilibrium solution. For discrete systems the stability is based on the eigenvalue relative to 1 (not 0) - so it is good to be aware of what type of system (discrete or continuous) you determining eigenvalues for! Now that we have an understanding of how this system works we can begin to look at stochasticity.

21.2 Environmental Stochasticity

It may be the case that environmental effects drastically change the net birth rate from one year to the next. For example it is known that in snowy winters the net birth rate changes because it is difficult to find food (Carroll 2013). For our purposes, let's say that in snowy winters r changes from 1.1 to 0.7. This would be a pretty drastic effect on the system - as one case is associated with exponential growth and the other exponential decay.

Because the years when snowy winters occur is random, we need to account for this in our model. One way is to create a conditional statement based on the probability of the moose to adjust to a deep snowpack in a given year. We will define this probability to be on a scale from 0 to 1, where 0 means the moose cannot adjust to a deep snowpack, and 1 they are able to adjust to a deep snowpack. How we implement that is writing a function that draws a uniform random number each year and adjusting the birth rate:

```
# We use the snowfall_rate  as an input variable

moose_snow <- function(snowfall_prob) {
  out_moose <- array(M0, dim = N)
  for (i in 1:(N - 1)) {
    r <- 1.1 # Normal net birth rate
    if (runif(1) > snowfall_prob) {
      r <- 0.7 # Decreased birth rate
    }
    out_moose[i + 1] <- r * out_moose[i]
  }
  return(out_moose)
}
```

Let's take a look at some solutions for different realizations of the moose population over time when the probability to adjust to the snowfall rate varies.

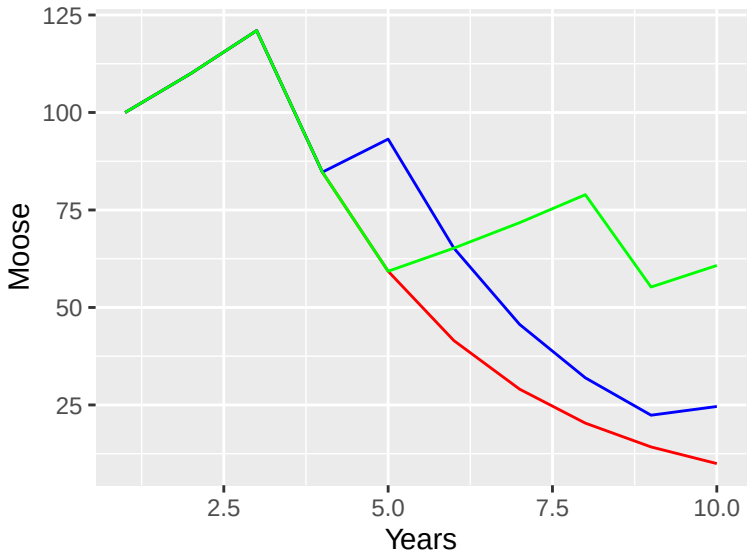


Figure 21.2: Moose populations with different probability of adjusting to deep snowpacks.

Run the above code on your computer. Do you obtain the same result? I hope not! We are drawing random numbers for each year, so you should have different trajectories. While this may seem like a problem, one key thing that we will learn in Section 22 is when we compute *multiple* simulations and then compute an ensemble average we see patterns in the results.

As you can see when the probability of adjusting to deep snowpacks is very high ($p = 0.75$), the population continues exponentially. If that probability is lower, it can still increase, but one bad year does knock the population down.

The moose example introduced how random effects into the dynamical system. Both contexts (environmental or demographic stochasticity) affected the net reproduction rate r, but in different ways. The important lesson is that the *type* of stochasticity matters just as much as *how* it is implemented.

21.3 Discrete systems of equations

Finally, one way that we can extend the moose model is to account for both adult and juvenile moose populations, as with the following model:

$$\begin{aligned} J_{t+1} &= F_M \cdot M_t \\ M_{t+1} &= G_J \cdot J_t + P_M \cdot M_t \end{aligned} \tag{21.5}$$

Equation (21.5) is a little different from (21.1) because it includes juvenile and adult moose populations, which has the following parameters:

- F_M: represents the birth rate of new juvenile moose
- G_J: represents the maturation rate of juvenile moose
- P_M: represents the survival probability of adult moose

We can code up this model using `R` in the following way:

```
M0 <- 900 # Initial population of adult moose
J0 <- 100 # Initial population of juvenile moose

N <- 10 # Number of years we run the simulation
moose_two_stage <- function(f, g, p) {

  # f: birth rate of new juvenile moose
  # g: maturation rate of juvenile moose
  # p: survival probability of adult moose
```

```
  # Create a data frame of moose to return
  out_moose <- tibble(
    years = 1:N,
    adult = array(M0, dim = N),
    juvenile = array(J0, dim = N)
  )

  # And now the dynamics
  for (i in 1:(N - 1)) {
    out_moose$juvenile[i + 1] <- f * out_moose$adult[i]
    out_moose$adult[i + 1] <- g * out_moose$juvenile[i] + p * out_moose$adult[i]
  }

  return(out_moose)
}
```

To simulate the dynamics we just call the function `moose_two_stage` and plot:

```
moose_two_stage_rates <- moose_two_stage(
  f = 0.5,
  g = 0.6,
  p = 0.7
)

ggplot(data = moose_two_stage_rates) +
  geom_line(aes(x = years, y = adult), color = "red") +
  geom_line(aes(x = years, y = juvenile), color = "blue") +
  labs(
    x = "Years",
    y = "Moose"
  )
```

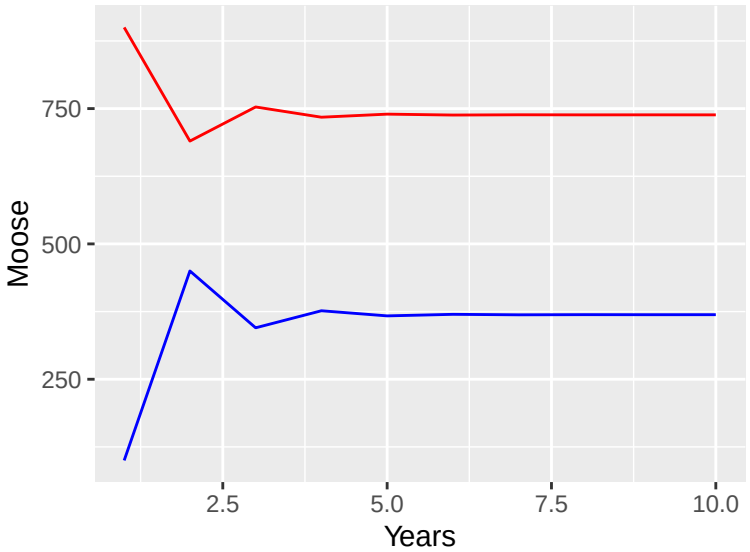


Figure 21.3: Simulation of a two stage moose population model. Adults are in red, juveniles in blue.

Looking at Figure 21.3, it seems like both populations stabilize after a few years. We could further analyze this model for stable population states (in fact, it would be similar to determining eigenvalues as in Section 18).

21.4 Exercises

Exercise 21.1. Re-run the moose population model with probabilities of adjusting to the deep snowpack at $p = 0,\ 0.1,\ 0.9,\ 1$. How does that adjusting the probability affect the moose population after 10 years?

Exercise 21.2. Modify the two stage moose population model (Equation (21.5)) with the following parameters and plot the resulting adult and juvenile populations:

a. $f = 0.6$, $g = 0.6$, $p = 0.7$
b. $f = 0.5$, $g = 0.6$, $p = 0.4$
c. $f = 0.3$, $g = 0.6$, $p = 0.5$

Exercise 21.3. (Inspired by Logan and Wolesensky (2009)) An animal reproduces two, one, or no offspring. The chance it produces one offspring is 0.50, two offspring 0.25, and no offspring 0.25. This animal does not survive after reproducing. Use the function `population` in the `demodeler` library to produce 1000 realizations of this stochastic process over 20 generations. Assume the initial population size is 10 individuals. Comment on the long term dynamics of the population.

Exercise 21.4. (Inspired by Logan and Wolesensky (2009)) You are playing a casino game. If you win the game earn a dollar. If you lose the game you lose one dollar. The probability of winning or losing is 50-50 (0.50). You start the game with \$100. You play the game 200 times. Use the function `casino` in the `demodelr` library to produce 1000 realizations of this stochastic process. Comment on the long term dynamics of your earnings. Then assuming the house win probability is 0.52 figure out how long the game, on average, runs before you are broke. Finally, adjusting the house win probability, hypothesize a function of the length of game as a function of the house win probability.

Exercise 21.5. Modify the two stage moose population model (Equation (21.5)) to account for years with large snowdepths. In normal years, $f = 0.5$, $g = 0.6$, $p = 0.7$. However for snowy years, $f = 0.3$, $g = 0.6$, $p = 0.5$. Generate code that can account for these variable rates (similar to the moose population model). Plot simulations when $N = 10$ and $N = 30$ and comment on the long-term dynamics of the moose.

Exercise 21.6. A population grows according the the growth law $x_{t+1} = r_t x_t$.

a. Determine the general solution to this discrete dynamical system.
b. Plot a sample growth curve with $r_t = 0.86$ and $r_t = 1.16$, with $x_0 = 100$. Show your solution for $t = 50$ generations.
c. Now consider a model where $r_t = 0.86$ with probability $1/2$ and $r_t = 1.16$ with probability $1/2$. Write a function that will predict the population after $t = 50$. Show three or four different realizations of this stochastic process.

Exercise 21.7. (Inspired by Logan and Wolesensky (2009)) A "patch" has area a, perimeter s, and a strip (band) of width w inside the boundary of a from which the animals disperse. Only those in the strip disperse. Let u_t be the number of animals in a at any time t. The growth rate of all the animals in a is r. The rate at which animals disperse from the strip is proportional to the fraction of the animals in the strip, with proportionality constant ϵ, which is the emigration rate for those in the strip.

a. Draw a picture of the situation described above.
b. Explain why the equation that describes the dynamics is

$$u_{t+1} = ru_t - \epsilon\left(\frac{w \cdot s}{a}u_t\right)$$

c. Determine conditions on the parameters r, w, s, ϵ, and a under which the population is growing.

Chapter 22

Simulating and Visualizing Randomness

In Section 21 we examined models for stochastic biological systems, which will lead us into the study of stochastic differential equations (SDEs). A key factor in understanding SDEs is learning basic tools to simulate and visualize results. In this section we will study how to visualize different simulations in `R` and ways to summarize the cohort of simulations over time.

22.1 Ensemble Averages

Consider Figure 22.1, which shows the weather forecast for Kuopio, a city in Finland:[1]

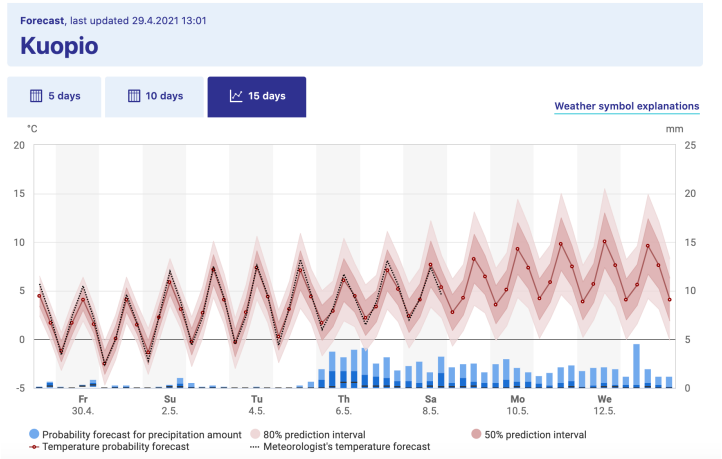


Figure 22.1: Long-term weather forecast for Kuopio, a city in Finland, from the Finnish Weather Institute.

Figure 22.1 shows a great example of what is called an *ensemble average*. The horizontal axis lists the time of day and the vertical axis is the temperature (the precipitation is in bars down below, but not important in this context). The forecast temperature at a given point in time can have a range of outcomes, with the median of the distribution as the "temperature probability forecast"[2]. The red shading states that 80% of the outcomes fall in a given range, so while the median temperature on Tuesday May 5 (labeled as Tu 4.5 - dates are represented as DAY-MONTH-YEAR) is 7 °C, it may range between 4 and 11 °C (39 to 51 °F). Based on the legends given, the 80% confidence interval is between 4 - 11 °C, or the models have 80% confidence for the temperature to be between that range of temperatures.

Because there may be different factors that alter the weather in a particular spot (e.g. the timing of a low-pressure front, or clouds, etc) there are different possibilities for an outcome of the weather forecast. While that may seem like forecasting weather may impossible to do, sometimes these changes lead to small fluctuations in the forecasted weather at a given point. The further out in time the more ensemble average becomes more uncertain (wider bars), as unforeseen events may drastically change the weather in the long term.

[1] I could have picked any city, but a lot of this textbook was written while I was on sabbatical in Kuopio. I highly recommend Finland as a country to visit.

[2] Notice the meteorologist's temperature forecast - sometimes that may diverge from the model outcomes!

Table 22.1: Simulations of a variable at different times.

t	sim1	sim2	sim3
1	3.92	2.32	0.26
2	0.81	0.22	4.28
3	1.59	3.51	3.14
4	4.23	2.41	1.11
5	4.29	2.33	2.26

In this section we will focus on how to take results from a simulation (such as several simulations of the moose population from Section 21) to generate ensemble average plots. Computing an ensemble average is a great way to show your uncertainty or confidence in a model outcome.

22.2 Computing ensemble averages

Let's build up how to compute an ensemble average step by step. For example let's say we have the following data in Table 22.1. Notice how all the simulations (`sim1`, `sim2`, `sim3`) share the variable `t` in common, so it makes sense to plot them on the same axis in Figure 22.3. We call a plot of all of the simulations together a *spaghetti plot*, because it can look like a bowl of spaghetti noodles was dumped all over the plot.

```
my_table %>%
  ggplot() +
  geom_point(aes(x = t, y = sim1)) +
  geom_line(aes(x = t, y = sim1)) +
  geom_point(aes(x = t, y = sim2)) +
  geom_line(aes(x = t, y = sim2)) +
  geom_point(aes(x = t, y = sim3)) +
  geom_line(aes(x = t, y = sim3)) +
  labs(x = "Time", y = "Simulation Value")
```

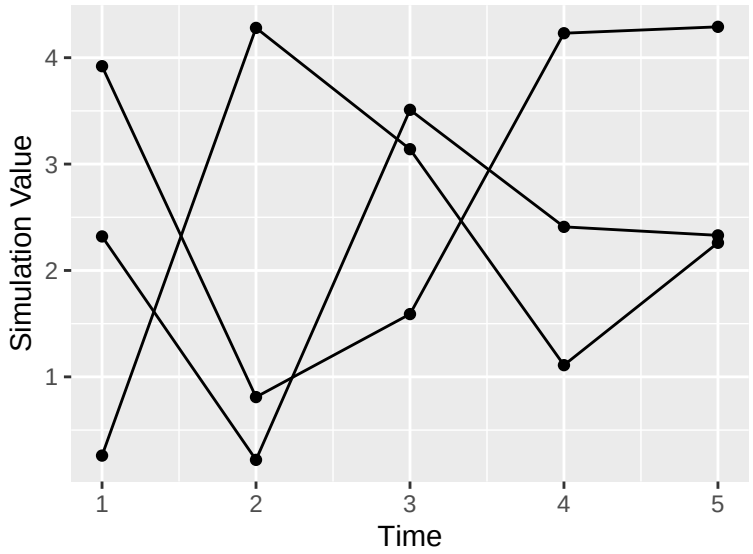


Figure 22.2: Initial Timeseries plot of the three simulations from Tablereftab:simul-table.

One thing to adjust in the code used to produce the spaghetti plot in Figure 22.2 is that the code is really long - we needed a `geom_point` and `geom_line` for each simulation. While this isn't bad when you have three simulations, one lesson we will learn is that the more simulations gives you better confidence in your results. If we have 500 simulations this would be a pain to code! Fortunately there is a command called `pivot_longer` that gathers different columns together (Table 22.2.

```
my_table_long <- my_table %>%
  pivot_longer(cols = c("sim1":"sim3"), names_to = "name", values_to = "value")
```

Table 22.2: Simulations of a variable at different times, condensed into a long table.

t	name	value
1	sim1	3.92
1	sim2	2.32
1	sim3	0.26
2	sim1	0.81
2	sim2	0.22
2	sim3	4.28
3	sim1	1.59
3	sim2	3.51
3	sim3	3.14
4	sim1	4.23
4	sim2	2.41
4	sim3	1.11
5	sim1	4.29
5	sim2	2.33
5	sim3	2.26

Notice how the command `pivot_longer` takes the different simulations (`sim1`, `sim2`, `sim3`) and reassigns the column names to a new column `name` and the values in the different columns to `value`. This process called pivoting creates a new data frame, which is easier to generate the spaghetti plot.

```
my_table_long %>%
  ggplot(aes(x = t, y = value, group = name)) +
  geom_point() +
  geom_line() +
  labs(x = "Time", y = "Simulation Value")
```

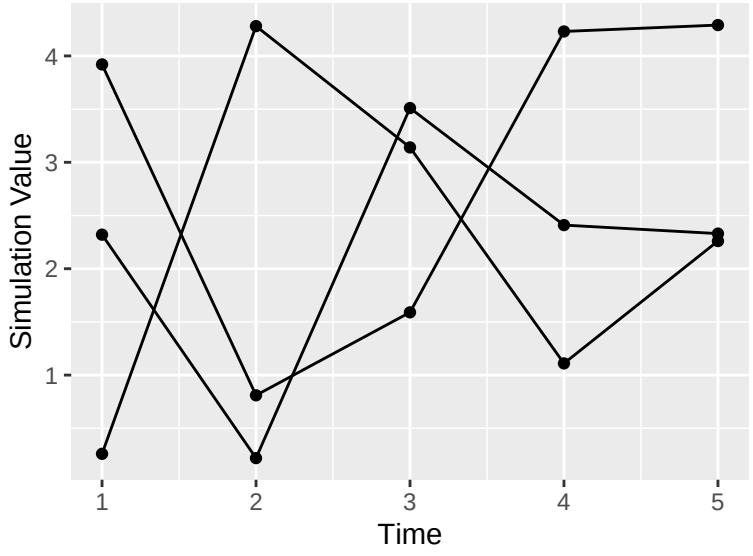


Figure 22.3: Timeseries plot of the three simulations from the pivoted data frame.

22.2.1 Grouping

The next step after pivoting is to collect the time points at $t = 1$ together, $t = 2$ together, and so on. In each of these groups we then compute the mean (average). This process is called *grouping* and then applying a summarizing function to all the members in a particular group (which in this case is the mean). Here is the code to do that process with the data frame `my_table_long`:

Table 22.3: Simulations of a variable at different times, along with the mean value (last column).

t	mean_val
1	2.166667
2	1.770000
3	2.746667
4	2.583333
5	2.960000

```
summarized_table <- my_table_long %>%
  group_by(t) %>%
  summarize(mean_val = mean(value))
```

Let's break this down.

- The command `group_by(t)` means collect similar time points together
- The next line computes the mean. The command `summarize` means to that we are going to create a new data frame column (labeled `mean_val` that is the mean of all the grouped times, from the `value` column.

We can add this mean value to our data (Figure 22.4), represented with a thick red line.

```
my_table_long %>%
  ggplot(aes(x = t, y = value, group = name)) +
  geom_point() +
  geom_line() +
  geom_line(data = summarized_table, aes(x = t, y = mean_val), color = "red", size = 2, inherit.aes = FALSE) +
  geom_point(data = summarized_table, aes(x = t, y = mean_val), color = "red", size = 2, inherit.aes = FALSE) +
  guides(color = "none") +
  labs(x = "Time", y = "Simulation Value")
```

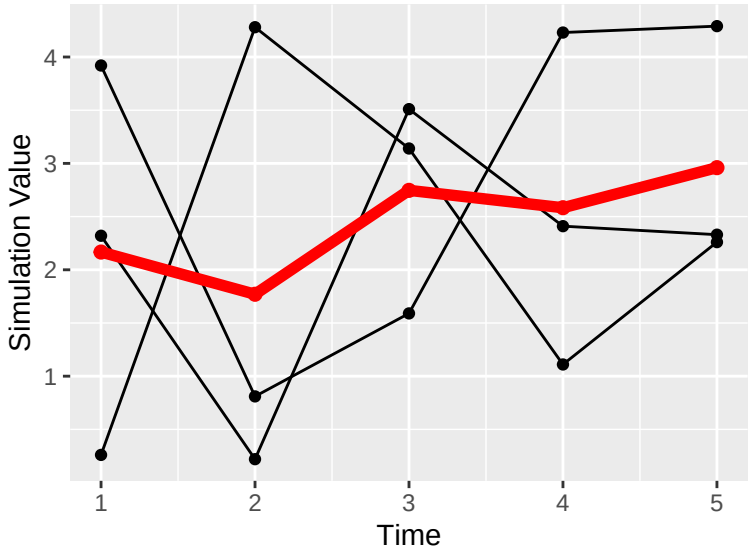


Figure 22.4: Timeseries plot of the three simulations.

Notice how we added the additional `geom_point` and `geom_line` with the option `inherit.aes = FALSE`. This stands for "inherit aesthetics" When you add a new data frame to a plot, the initial aesthetics (such as the `color` or the `group`) are passed on. Setting `inherit.aes = FALSE` allows you to work with a clean slate.

Computing the other parts of the ensemble average requires knowledge of how to compute percentiles of a distribution of values. For our purposes here we will use the 95% confidence interval, so that means the 2.5th, 97.5th percentile (which only 5% of the values will be outside of the specified interval), along with the median value (50th percentile). Let's take a look at the code how do do that:

Table 22.4: Simulations of a variable at different times, along with the median and the 95% confidence interval.

t	q0.025	q0.5	q0.975
1	0.3630	2.32	3.8400
2	0.2495	0.81	4.1065
3	1.6675	3.14	3.4915
4	1.1750	2.41	4.1390
5	2.2635	2.33	4.1920

```
quantile_vals <- c(0.025, 0.5, 0.975)

quantile_table <- my_table_long %>%
  group_by(t) %>%
  summarize(
    q_val = quantile(value,
      probs = quantile_vals
    ),
    q_name = quantile_vals
  ) %>%
  pivot_wider(names_from = "q_name", values_from = "q_val", names_glue = "q{q_name}")
```

While this code is a little more involved, let's break it down piece by piece:

- To make things easier the variable `quantile_vals` computes the different quantiles, expressed between 0 to 1 - so 2.5% is 0.025, 50% is 0.5, and 97.5% is .975.
- As above, we are still looking grouping by the variable `t` and summarizing our data frame.
- However rather than applying the mean, we are using the command `quantile`, whose value is computed with the new column `q_val`. Like the mean, we define which columns we apply the quantile function in the column `value`. We use `probs = quantile_vals` to specify the quantiles that we wish to compute.
- We also create a new column called `q_name` the contains the names of the quantile probabilities.
- The command `pivot_wider` takes the values in `q_val` with the associated names in `q_name` and creates new columns associated with each quantile. This process of making a data frame wider is the opposite of making a skinny and tall data frame with `pivot_longer`. Notice that a key convention in column names is that they don't start with a number, so we *glue* a `q` onto the names of the column.

Wow. This is getting involved. One thing to keep in mind is that the the code as written should be easily adaptable if you need to compute an ensemble average. The last command about pivoting may seem foreign, but learning how to pivot data is a good skill to invest time in. If you take an introductory data science or data visualization course I bet you will learn more about the role of pivoting data - but for now you can just adapt the above code to your needs

Now onto plotting the data. What we will do is plot the ensemble average as a ribbon, so that is a new plot `geom_ribbon`. It requires a few more aesthetics (`ymin` and `ymax`, or the minimum and maximum y values to be plotted).

```
my_table_long %>%
  ggplot(aes(x = t, y = value, color = name)) +
  geom_point() +
  geom_line() +
  geom_line(data = quantile_table, aes(x = t, y = q0.5), color = "red", size = 2, inherit.aes = FALSE) +
  geom_point(data = quantile_table, aes(x = t, y = q0.5), color = "red", size = 2, inherit.aes = FALSE) +
  geom_ribbon(data = quantile_table, aes(x = t, ymin = q0.025, ymax = q0.975), alpha = 0.2, fill = "red", inhe
  guides(color = "none") +
  labs(x = "Time", y = "Simulation Value")
```

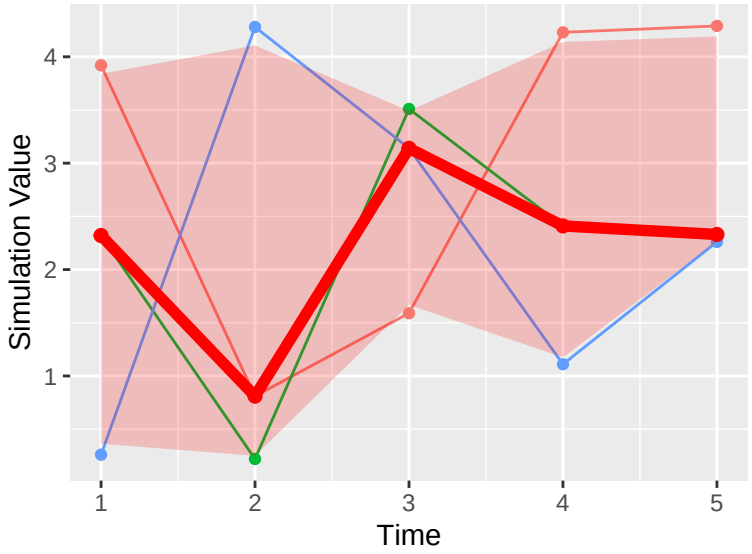


Figure 22.5: Timeseries plot of the three simulations, along with the 95% confidence interval (red).

A lot of this code to produce Figure 22.5 is similar to when we plotted the mean value (Figure 22.4), but we changed the second data frame to be plotted to `quantile_table` and included the `geom_ribbon` command. The command `alpha = 0.2` refers to the transparency of the plot. The `fill` aesthetic just provides the shading (in other words the fill) between `ymin` and `ymax`.

22.3 Doing many simulations and visualizing

Now that we have some idea of a ensemble average, let's put this into practice by doing some simulations of some one and two dimensional equations. What you will learn here is how to generate a simulation

For example consider the following code that produces a solution to the logistic differential equation:

```
logistic_eq <- c(dx ~ r * x * (1 - x / K)) # Define the rate equation

params <- c(r = .8, K = 100) # Identify any parameters

init_cond <- c(x = 3) # Initial condition
soln <- euler(
  system_eq = logistic_eq,
  initial_condition = init_cond,
  parameters = params,
  deltaT = .05,
  n_steps = 200
)

# Plot your solution:

ggplot() +
  geom_line(data = soln, aes(x = t, y = x)) +
  labs(
    x = "Time",
    y = "x"
  )
```

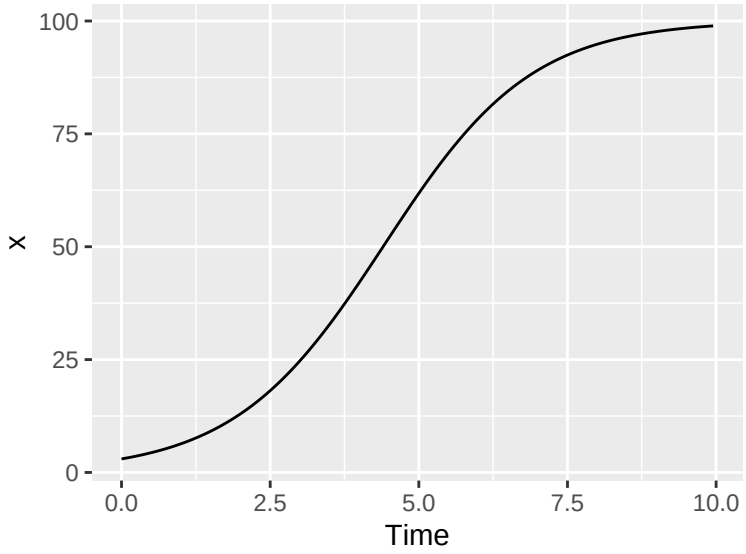


Figure 22.6: Solution to the logistic differential equation.

What if you wanted to investigate the effect of the initial condition on the solution, so the solution was randomly chosen, from a uniform distribution between 0 and 20? Well, to do one simluation is not a small modification of the code:

```
logistic_eq <- c(dx ~ r * x * (1 - x / K)) # Define the rate equation

params <- c(r = .8, K = 100) # Identify any parameters

init_cond_rand <- c(x = runif(1, min = 0, max = 20)) # Random initial condition

soln_rand <- euler(
  system_eq = logistic_eq,
  initial_condition = init_cond_rand,
  parameters = params,
  deltaT = .05,
  n_steps = 200
)

# Plot your solution:

ggplot() +
  geom_line(data = soln, aes(x = t, y = x), color = "black") +
  geom_line(data = soln_rand, aes(x = t, y = x), color = "red") +
  labs(
    x = "Time",
    y = "x"
  )
```

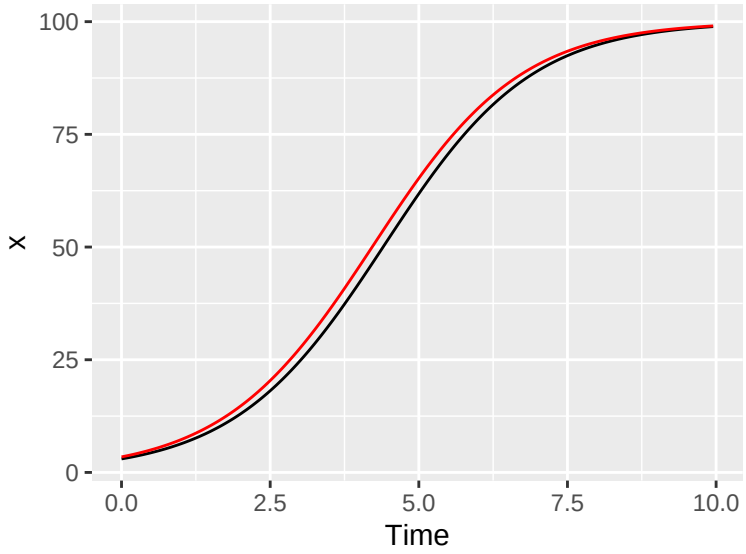


Figure 22.7: Solution to the logistic differential equation with a random initial condition (red). The original solution is also plotted for comparison.

Running several different iterations could quickly grow time consuming if you wanted to do this several times (think hundreds). Fortunately we will can use iteration here to compute and gather several different solutions. I am going to post the code and then we can deconstruct it:

```
n_sims <- 500 # The number of simulations

# Compute solutions
try1 <- rerun(n_sims, c(x = runif(1, min = 0, max = 20))) %>%
  set_names(paste0("sim", 1:n_sims)) %>%
  map(~ euler(
    system_eq = logistic_eq,
    initial_condition = .x,
    parameters = params,
    deltaT = .05,
    n_steps = 200
  )) %>%
  map_dfr(~.x, .id = "simulation")


# Plot these all up together
try1 %>%
  ggplot(aes(x = t, y = x)) +
  geom_line(aes(color = simulation)) +
  ggtitle("Random initial conditions") +
  guides(color = "none")
```

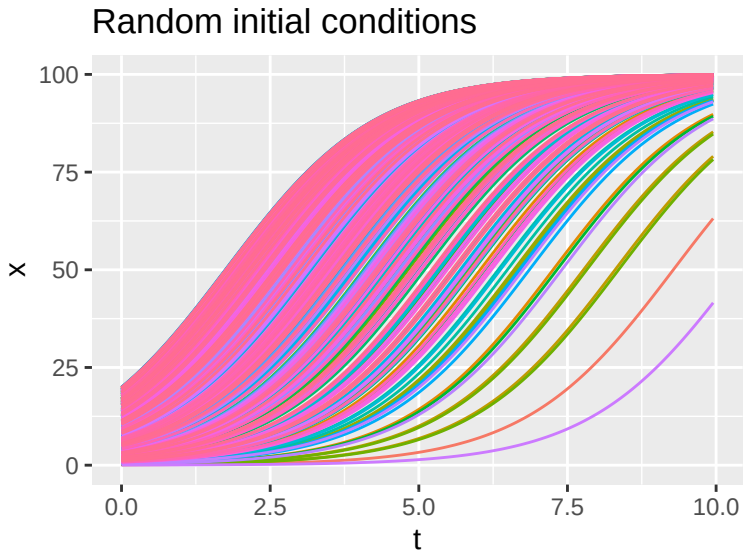


Figure 22.8: Many realiziations of the logistic differential equation with random initial conditions.

Wow! This plot is really interesting - it should show how even though the initial conditions vary between $x = 0$ to $x = 20$, eventually all solutions flow to the carrying capacity $K = 100$ (which is a stable equilbrium solution). Initial conditions that start closer to $x = 0$ take longer, mainly because they are so close to the other equilibrium solution at $x = 0$ (which is an unstable equilibrium solution).

Ok, let's deconstruct this code line by line:

- `rerun(n_sims, c(x=runif(1,min=0,max=20)))` This line does two things: `x=runif(1,min=0,max=20)` makes a random initial condition, and the command `rerun` runs this again for `n_sims` times.
- `set_names(paste0("sim", 1:n_sims))` This line distinguishes between all the different simulations.
- `map(~ euler( ... )` You should be familiar with `euler`, but notice the pronoun `.x` that substitutes all the different initial conditions into Euler's method. The `map` function iterates over each of the simulations.
- `map_dfr(~ .x, .id = "simulation")` This line binds everything up together.

This code applies a new concept called *functional programming*. This is a powerful tool that allows you to perform the process of iteration (do the same thing repeatedly) with uncluttered code. We won't delve more into this here, but I encourage you to read about more programming concepts in https://r4ds.had.co.nz/iteration.html by Wickham and Grolemund

22.4 Exercises

Exercise 22.1. Using the code to produce Figure 22.5:

a. Adjust the `alpha` level to a number between 0 and 1. What does that do to the plot?
b. Adjust the `fill` level to a color of your choosing. A list of R Colors can be found here: http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/

Exercise 22.2. Read the Chapter 12 in R for Data Science: https://r4ds.had.co.nz/tidy-data.html. In this chapter you will learn about tidy data. Explain what tidy data is and the potential uses for pivoting data wider or longer.

Exercise 22.3. Look at the documentation for quantile (remember you can type `?quantile` at the command line to see the associated help for this function.) Change the ensemble average in `quantile_table` to compute the 25%, 50%, and 75% percentile. Finally, produce a ensemble average plot of this percentile. How does your result compare to Figure 22.5?

Exercise 22.4. Take the data frame `try1` from the simulations of the logistic equation and generate an ensemble average (median with 95% confidence interval) of the different simulations and make a plot of the ensemble average without the different simulations.

Exercise 22.5. Using the data frame `my_table`, compare the following code below. The data frame `table1` is skinny and long, and the second data frame `table2` is called short and wide. Why did we need to make this data frame short and wide for plotting?

```
# First code chunk
table1 <- my_table %>%
  rowwise(t) %>%
  summarise(q_val = quantile(c_across(starts_with("sim")),
                             probs = quantile_vals),
            q_name = quantile_vals)

# Second code chunk
table2 <- my_table %>%
  rowwise(t) %>%
  summarise(q_val = quantile(c_across(starts_with("sim")),
                             probs = quantile_vals),
            q_name = quantile_vals) %>%
  pivot_wider(names_from = "q_name",values_from="q_val",names_glue = "q{q_name}")
```

Chapter 23

Random Walks

Sections 21 and 22 introduced the concept of stochastic dynamical systems and ways to compute ensembles averages. In this section we will begin to develop some tools about how to understand stochastics by studying the concept of *random walks*. An excellent, highly readable book on random walks in biology is Howard Berg (1993).

23.1 Random walk on a number line

The conceptual idea of a random walk begins on a number line. Let's begin at the origin (so at $t = 0$ then $x = 0$). Based on this number line we can only to the left or the right, with equal probability. At a given time we decide to move in a direction based on a random number r drawn between 0 and 1 (in `R` we do this with the command `runif(1)`). Figure 23.1 conceptually illustrates this random walk

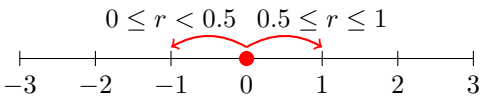
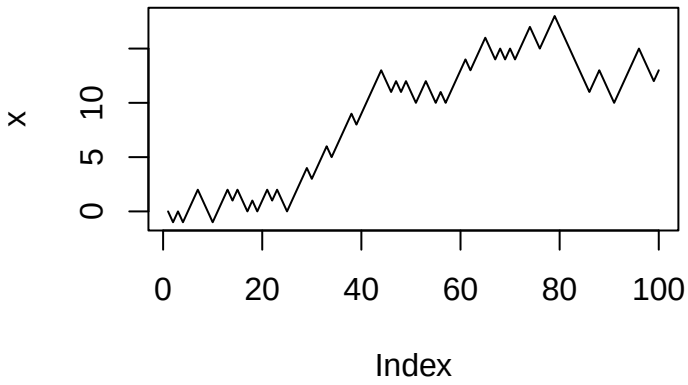


Figure 23.1: Schematic diagram for one-dimensional random walk.

For each iteration of this process we will draw a random number using `runif(1)`. We can code this process using a `for` loop:

```
### Number of steps our random walk takes
number_steps <- 100

### Set up vector of results
x <- array(0, dim = number_steps)

for (i in 2:number_steps) {
  if (runif(1) < 0.5) {
    x[i] <- x[i - 1] - 1
  } # Move right
  else {
    x[i] <- x[i - 1] + 1
  } # Move left
}

# Let's take a peek at our result:

plot(x, type = "l")
```




```
print(mean(x)) # Where our average position was over the time interval
```

```
## [1] 8.84
```

```
print(sd(x)) # Where our standard deviation was over the time interval
```

```
## [1] 5.807901
```

Let's remind ourselves what this code does:

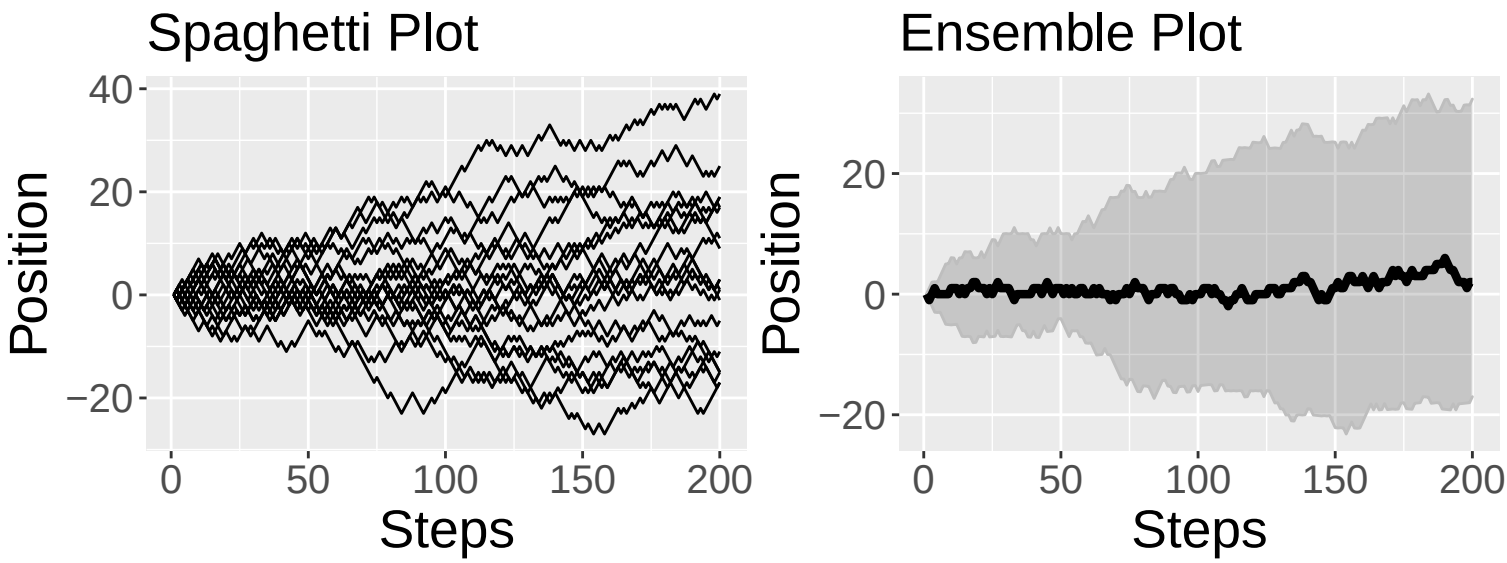
- `number_steps <- 100`: The number of times we draw a random number, referred to *steps*.
- `x <- array(0,dim=number_steps)`: We are going to pre-allocate a vector (`array`) of our results. Values in this array are all set at 0 for convenience.
- The for loop starts at the second step and then either adds or subtracts one from the prevoius position `x[i-1]` and updates the result to `x[i]`.
- `plot(x,type='l')` makes a simple line plot of the results.

Now that you have run this code, try running it again. Do you get the same result? I hope you didn't - because this process is random! It is interesting to run it several times because there can be a wide variance in our results - for some realizations of the sample path, we end up being strictly positive, other times we go negative, and other times we just hover around the middle.

23.2 More realizatons

Similar to section on stochastic simulation, One thing that would be helpful is if we ran the simulation for multiple times, or multiple realizations. The code `random_walk` can help us do that:

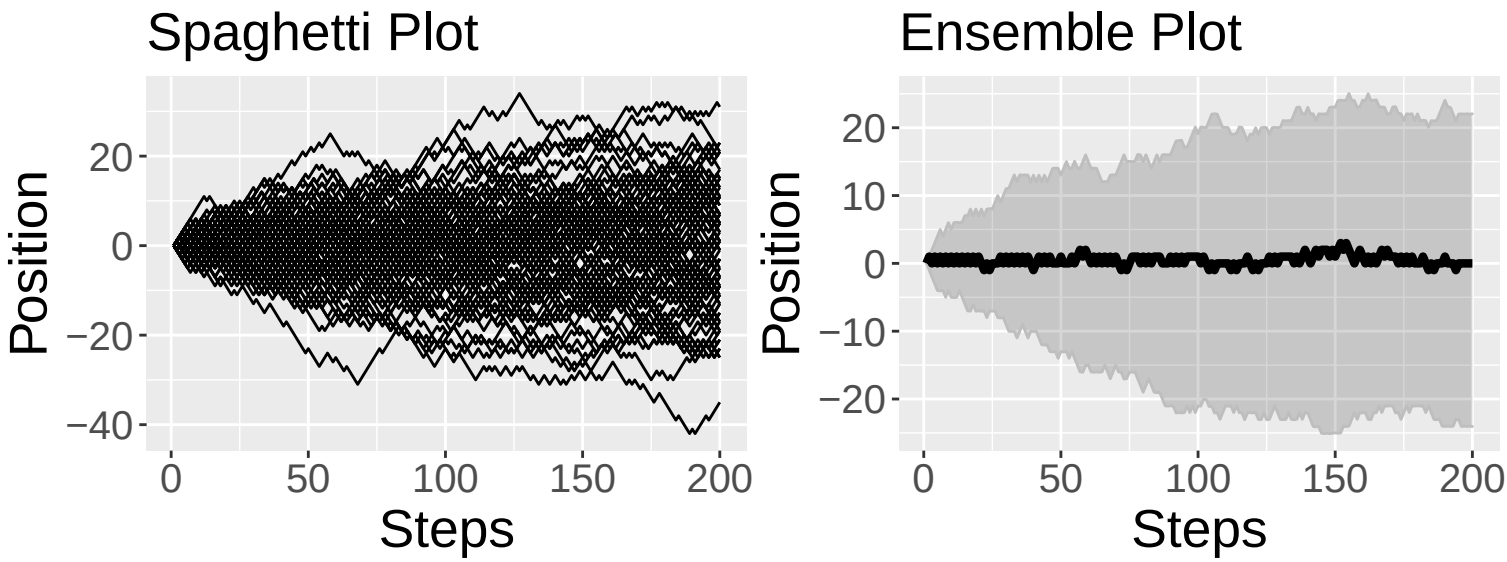
```
number_steps <- 200 # How long we run our random walk
number_realizations <- 20 # How many separate realizations we do

random_walk(number_steps, number_realizations)
```



You will notice two plots get produced: (1) A **spaghetti plot** that plots all the different sample paths for this realization, and (2) a **ensemble plot** that takes the median and 95% confidence interval of the results. (While we learned how to visualize spaghetti plots and compute ensemble averages in the Section 22, for ease of use this code does it all in one.)

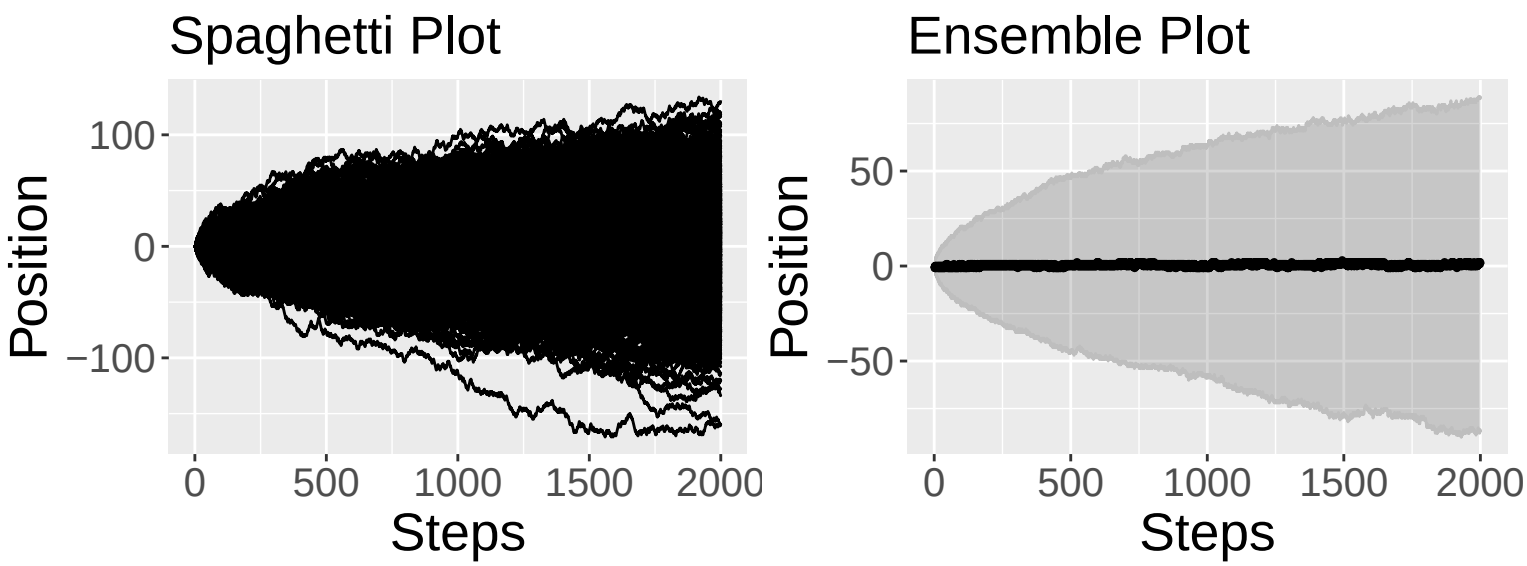
Something interesting looks like it is going on here. The ensemble plot looks like a sideways parabola, but let's check to make sure that is the case. Perhaps rerun `random_walk` but set `number_realizations` to be 100:

```
number_steps <- 200
number_realizations <- 100

random_walk(number_steps, number_realizations)
```



As the confidence interval increases as the number of steps increase, we get an interesting observation. These results suggest that on *average* you go nowhere (in other words, the average position is $x = 0$), but as the number of steps increase, you are very likely to be *somewhere* (in other words, the confidence interval increases as the number of steps increase). The more realizations we can do the more robust this pattern becomes. Consider what occurs to the spaghetti and ensemble plots when the number of realizations is 1000:

```
random_walk(2000, 1000)
```



Let's investigate our observations a little more mathematically.

23.3 Random walk mathematics

Call x^i the position x at step i in a random walk. While we have set this up to be a unit walk, more generally $x^i = x^{i-1} + p(r)\ \Delta x$, with Δx being the jump size (in this case $\Delta x = 1$) and r being a random variable:

$$r = \begin{cases} -1 & p(-1) = 0.5 \\ 1 & p(1) = 0.5 \end{cases} \tag{23.1}$$

Note that in Equation (23.1) r is an example of a (https://en.wikipedia.org/wiki/Bernoulli_distribution)[*Bernoulli Random Variable*] - it can be either 0 or 1, with equal probability. Equation (23.1) is sometimes referred to as the *evolution equation*. If we have multiple simulations then x^i_j is the position at step i for simulation j.

Let's introduce some terminology to help us out here. $\langle x^i \rangle = \dfrac{1}{n} \displaystyle\sum_{j=1}^{n} x^i_j$ is the *expected value* of our position at step i, summed over all of the simulations at that timestep. Notice the connection here to the ensemble average at a given point.

You will learn in a probability theory class that the expected value is a linear operator. Because of this, we can compute the expected value of our evolution equation:

$$\langle x^i \rangle = \langle x^{i-1} + r\ \Delta x \rangle = \langle x^{i-1} \rangle + \langle r\ \Delta x \rangle = \langle x^{i-1} \rangle + \langle r \rangle\ \Delta x \tag{23.2}$$

Let's focus in particular on the second term of this expression: $\langle r \rangle$. Because r is a discrete random variable we can also compute its expected value as well. To calculate $\langle r \rangle$ we adding up the possible outcomes for r (either 1 or -1) when multiplied by their associated probabilities (for this case 0.5 ($r = 1$) or 0.5 ($r = -1$) - note how these probabilities sum to 1):[1]

$$\langle r \rangle = 1 \cdot 0.5 - 1 \cdot 0.5 = 0 \tag{23.3}$$

So, the expected value of the evolution equation is $\langle x^i \rangle = \langle x^{i-1} \rangle$. This may not seem helpful, but let's start thinking of this value from the beginning (that is when $i = 0$, or when we begin at $x = 0$). Now let's evolve to the next timestep. By what we found, then $\langle x^1 \rangle = \langle x^0 \rangle = 0$. Connect these together: $\langle x^1 \rangle = 0$ as well. What do you think happens at $\langle x^2 \rangle$? If you guessed 0, you are correct, because $\langle x^2 \rangle = \langle x^1 \rangle = \langle x^0 \rangle = 0$. In fact, this pattern continues so that $\langle x^i \rangle = \langle x^{i-1} \rangle = 0$.

What does this tell us? Between all of this notation, $\langle x^i \rangle$ is the expected value at each timestep, so we showed that *the expected value at any time* is zero. In other words, *random walk goes nowhere*!

[1] Generally speaking for a discrete random variable x, $\langle x \rangle = \sum p_i \cdot x_i$, with $\sum p_i = 1$.

23.3.1 Random walk variance

Now that we have characterized the expected value or the average displacement let's do the variance of this random walk. We calculate this by computing the mean square displacement, or $\langle (x^i)^2 \rangle$. First we compute the square displacement using the evolution equation and multiplying out:

$$(x^i)^2 = (x^{i-1} + r\ \Delta x)^2 = (x^{i-1})^2 + 2x^{i-1}\ r\ \Delta x + r^2\ \Delta x^2. \tag{23.4}$$

Next what we will compute the expectation $\langle (x^i)^2 \rangle$ of our multiplied expression term by term:

- $\langle (x^{i-1})^2 \rangle$: There is not much we can do with this term.
- $\langle 2x^{i-1}\ r\ \Delta x \rangle$: This term is zero from the expected value:

$$\langle 2x^{i-1}\ r\ \Delta x \rangle = 2x^{i-1}\ \Delta x \cdot 0.5 - 2x^{i-1}\ \Delta x \cdot 0.5 = 0 \tag{23.5}$$

- $\langle r^2\ (\Delta x)^2 \rangle$: We know that r be either -1 or 1, but we then square the value of r in Equation (23.1):

$$r^2 = \begin{cases} -1 \to (-1)^2 = 1 & p(-1) = 0.5 \\ 1 \to (1)^2 = 1 & p(1) = 0.5 \end{cases} \tag{23.6}$$

  So when we compute the expected value of this term we obtain:

$$\langle r^2\ (\Delta x)^2 \rangle = 1\ (\Delta x)^2 \cdot 0.5 + 1\ (\Delta x)^2 \cdot 0.5 = (\Delta x)^2 \tag{23.7}$$

So the end result for the variance is:

$$\left\langle (x^i)^2 \right\rangle = \left\langle (x^{i-1})^2) \right\rangle + (\Delta x)^2 \tag{23.8}$$

In order to understand this let's write out the first few terms of this recursive relationship:

$$\begin{aligned} \left\langle (x^0)^2 \right\rangle &= 0 \\ \left\langle (x^1)^2 \right\rangle &= \left\langle (x^0)^2) \right\rangle + (\Delta x)^2 = (\Delta x)^2 \\ \left\langle (x^2)^2 \right\rangle &= \left\langle (x^1)^2) \right\rangle + (\Delta x)^2 = 2(\Delta x)^2 \\ \left\langle (x^3)^2 \right\rangle &= \left\langle (x^2)^2) \right\rangle + (\Delta x)^2 = 3(\Delta x)^2 \end{aligned} \tag{23.9}$$

There is a pattern here, in fact $\langle (x^i)^2 \rangle = i(\Delta x)^2$. So the variance, or the mean square displacement grows, proportional to the step size. Another way to state this is that the standard deviation (the square root of the variance) is equal to $\pm\sqrt{n}\ \Delta x$, where n is the current step. This matches up with our graphs from earlier since $\Delta x = 1$! Informally this tells us that on *average* you go nowhere, but *eventually* you travel everywhere - how cool!

23.4 Continuous random walks (diffusion)

One final thought can be made here. We are taking discrete steps but we can transform our results to a continuous time analog. Let $t = n\Delta t$ be the approximation from discrete time to continuous time. Equivalently $n = \dfrac{t}{\Delta t}$. With this information we can rearrange the square displacement equation to the following:

$$\left\langle (x^n)^2 \right\rangle = \frac{t}{\Delta t}(\Delta x)^2. \tag{23.10}$$

The quantity $D = \dfrac{(\Delta x)^2}{2\Delta t}$ is known as the *diffusion coefficent*. So then the mean square displacement can be arranged as $\langle (x^n)^2 \rangle = 2Dt$, confirming again that the variance grows proportional to t.

To connect this back to our discussion of stochastics, understanding a random walk helps us to understand how demographic and environmental stochasticity affect a dynamical system and the types of behaviors in the solution this random walk introduces to the system. In the following sections we will investigate the ways in which the random walk connects to stochastic differential equations.

23.5 Exercises

Exercise 23.1. In class we found that the diffusion coefficient is equal to $D = \frac{(\Delta x)^2}{2\Delta t}$.

a. Solve the expression for D in terms of Δt.
b. The diffusion coefficient for oxygen in water is approximately 10^{-5} cm^2 sec^{-1}. Use that value to complete the following table:

Distance (Δx)	μm $= 10^{-6}$ m	10 μm	1 mm	1 cm	1 m
Diffusion time (Δt)					

Report your the diffusion time in an appropriate unit (seconds, minutes, hours, years) accordingly.

c. Navigate to the following website, which lists sizes of different cells: (https://en.wikibooks.org/wiki/Cell_Biology/Introduction/Cell_size)[LINK]. For what cells would diffusion be a reasonable process to transport materials?

Exercise 23.2. You are playing a casino game. If you win the game earn a dollar. If you lose the game you lose one dollar. The probability of winning or losing is 50-50 (0.50). You start the game with $100.

a. Write a one-dimensional random walk to simulate your money after playing the game 50 times. Make a few sample plots.
b. Based on the results of this section, what do you think your long-term expected winnings will be?
c. Now assume the house win probability is 0.52. Modify your random walk to simulate your money after playing the game 50 times. Make a few sample plots.
d. What do you think your long-term expected winnings of this modified game will be? (You may need to play the game for 100, 200 times to see a pattern.)

Exercise 23.3. Compute $\langle r \rangle$ for the following random variable:

$$r = \begin{cases} -1 & p(r) = 0.52 \\ 1 & p(r) = 0.48 \end{cases} \tag{23.11}$$

Exercise 23.4. Compute $\langle r \rangle$ for the following random variable:

$$r = \begin{cases} -1 & p(-1) = q \\ 1 & p(1) = (1-q) \end{cases} \tag{23.12}$$

Exercise 23.5. Consider the following random variable:

$$r = \begin{cases} -1 & p(-1) = 1/3 \\ 0 & p(0) = 1/3 \\ 1 & p(1) = 1/3 \end{cases} \tag{23.13}$$

a. Modify the code for the one dimensional random walk to generate a simulation of this random walk and plot your result. You can do this by applying a `if else` statement as shown in the code chunk below.:

b. Compute $\langle r \rangle$ and $\langle r^2 \rangle$.

c. Based on your last answer, explain how this random variable introduces a different random walk than the one described in this section. In what ways do you tihnk this would change our calculations for the mean and variance of the ensemble simulations?

```
p <- runif(1)
if (p < 1 / 3) {
  x[i] <- x[i - 1] - 1
} else if (1 / 3 <= p & p < 2 / 3) {
  x[i] <- x[i - 1]
} else {
  x[i] <- x[i - 1] + 1
}
```

Exercise 23.6. In this exercise you will write code for a two dimensional random walk.

a. Modify the code for the one dimensional random walk to have both an x and a y position. One way to do this is to create a variable y structured similar to x, and to make a second `if` statement in your for loop that moves `y`.
b. Plot a few different realizations of your sample paths.
c. If we were to compute the mean and variance of the ensemble simulations, what do you think they would be?

Chapter 24

Diffusion

By studying random walks in Section 23 you saw how randomness led to some surprising results, namely that for an unbiased random walk the mean displacement was zero but the variance increased proportional to the step size. In this section we will revisit the random walk problem from another perspective that further strengthens its connection todiffusion.

24.1 Random walk redux

In our random walk derivation in Section 23 we focused on the *position* of a particle on the random walk, based upon prescribed rules. In this scenario we instead are going to consider the *probability* that a particle is at position x in time t. We will denote this probability as $p(x,t)$. In other words, rather than focusing on where the particle is, we focus on the chance that the particle is at a given spot.

A way to conceptualize this is at any given position x, a particle can arrive there from either the left or the right, illustrated in Figure 24.1.

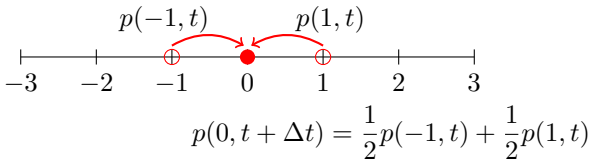


Figure 24.1: Schematic diagram for one-dimensional random walk.

Notice how the the particle in Figure 24.1 could only move in unit increments. To generalize this setup a little more where to increments Δx, the equation that defines the probability p of being at position x at any time t is the following:

$$p(x,t+\Delta t) = \frac{1}{2}p(x-\Delta x,t) + \frac{1}{2}p(x+\Delta x,t) \tag{24.1}$$

The next step to analyze this expression is apply Taylor approximations on each side of Equation (24.1). First let's do a locally linear approximation for $p(x,t+\Delta t)$:

$$p(x,t+\Delta t) \approx p(x,t) + \Delta t \cdot p_t, \tag{24.2}$$

where we have dropped the shorthand $p_t(x,t)$ as p_t. On the right hand side of Equation (24.1) we will compute the 2nd degree (quadratic) Taylor polynomial:

$$\begin{aligned}
\frac{1}{2}p(x-\Delta x,t) &\approx \frac{1}{2}p(x,t) - \frac{1}{2}\Delta x \cdot p_x + \frac{1}{4}(\Delta x)^2 \cdot p_{xx} \\
\frac{1}{2}p(x+\Delta x,t) &\approx \frac{1}{2}p(x,t) + \frac{1}{2}\Delta x \cdot p_x + \frac{1}{4}(\Delta x)^2 \cdot p_{xx}
\end{aligned}$$

With these approximations we can re-write Equation (24.1) as Equation (24.3):

$$\Delta t \cdot p_t = \frac{1}{2}(\Delta x)^2 p_{xx} \rightarrow p_t = \frac{1}{2}\frac{(\Delta x)^2}{\Delta t} \cdot p_{xx} \tag{24.3}$$

Equation (24.3) is called a partial differential equation - what this means is that it is a differential equation with derivatives that depend on two variables (x and t (two derivatives). Equation (24.3) is called the **diffusion equation**. Cool!

We can also simplify Equation (24.3) even more by defining $D = \frac{1}{2}\frac{(\Delta x)^2}{\Delta t}$, so we have $p_t = D \cdot p_{xx}$. Other derivations of the diffusion equation let $\Delta t \rightarrow 0$ and $\Delta x \rightarrow 0$ in the limit, but because of Equation (24.3) is connected to a random walk we will not make that limiting argument.

Determining an exact solution to the diffusion equation requires more study in techniques of partial differential equations, so we will leave that for another time. However, the solution to Equation (24.3) is given by Equation (24.4):

$$p(x,t) = \frac{1}{\sqrt{4\pi Dt}} e^{-x^2/(4Dt)} \tag{24.4}$$

Figure 24.2 shows plots for $p(x,t)$ when $D = 0.5$ (we will call these plots profiles) for different values of t: What Equation (24.4) represents is the probability that the particle is at the position x at time t. In the exercises for this section you will examine more properties of the diffusion equation.

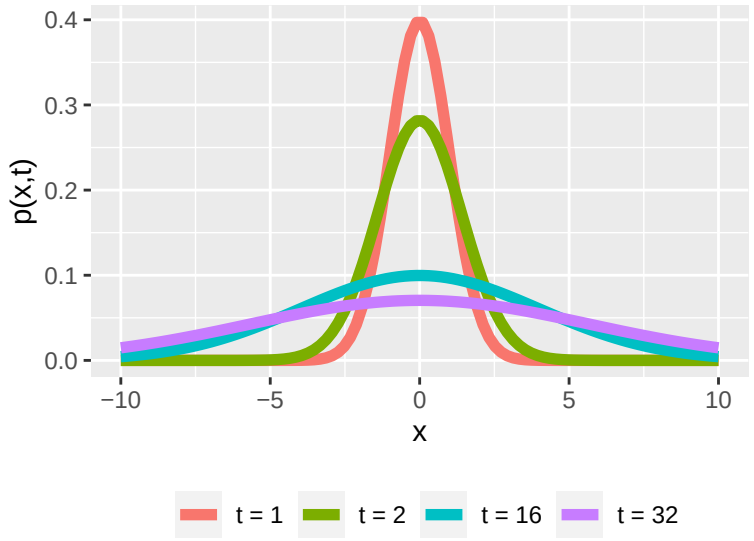


Figure 24.2: Profiles of $p(x,t)$ (solution to the diffusion equation) for different values of t with $D = 0.5$.

As you can see that as time increases the graph of $p(x,t)$ gets flatter - or more uniform. What this tells you is that the longer t increases it is less likely to find the particle at the origin.

24.1.1 Verifying the solution to the diffusion equation

Verifying Equation (24.4) is the solution to Equation (24.3) is a good review of your multivariable calculus skills! As a first step to verifying this solution, let's take the partial derivative with respect to x and t.

First we will compute $\frac{\partial p}{\partial x}$ (also represented as p_x):

$$\begin{aligned} p_x &= \frac{\partial}{\partial x}\left(\frac{1}{\sqrt{4\pi Dt}} e^{-x^2/(4Dt)}\right) \\ &= \frac{1}{\sqrt{4\pi Dt}} e^{-x^2/(4Dt)} \cdot \frac{-2x}{4Dt} \end{aligned}$$

Notice something interesting here: $p_x = p(x,t) \cdot \left(\frac{-x}{2Dt}\right)$.

To compute the second derivative, we have the following expressions by applying the product rule:

$$\begin{aligned} p_{xx} &= p_x \cdot \left(\frac{-x}{2Dt}\right) - p(x,t) \cdot \left(\frac{1}{2Dt}\right) \\ &= p(x,t) \cdot \left(\frac{-x}{2Dt}\right) \cdot \left(\frac{-x}{2Dt}\right) - p(x,t) \cdot \left(\frac{1}{2Dt}\right) \\ &= p(x,t) \left(\left(\frac{-x}{2Dt}\right)^2 - \left(\frac{1}{2Dt}\right)\right) \\ &= p(x,t) \left(\frac{x^2 - 2Dt}{(2Dt)^2}\right). \end{aligned}$$

So far so good. Now computing p_t gets a little tricky because this derivative involves both the product rule with the chain rule in two places (the variable t appears twice in the formula for $p(x,t)$). To aid in computing the derivative we identify two functions $f(t) = \dfrac{1}{\sqrt{4\pi Dt}} = (4\pi Dt)^{-1/2}$ and $g(t) = \dfrac{-x^2}{4Dt} = -x^2 \cdot (4Dt)^{-1}$. This changes $p(x,t)$ into $p(x,t) = f(t) \cdot e^{g(t)}$. In this way $p_t = f'(t) \cdot e^{g(t)} + f(t) \cdot e^{g(t)} \cdot g'(t)$. Now we can focus on computing the individual derivatives $f'(t)$ and $g'(t)$ (after simplification - be sure to verify these on your own!):

$$\begin{aligned} f'(t) &= -\frac{1}{2}(4\pi Dt)^{-3/2} \cdot 4\pi D = -2\pi D\ (4\pi Dt)^{-3/2} \\ g'(t) &= x^2\ (4Dt)^{-2} 4D = \frac{x^2}{4Dt^2} \end{aligned}$$

Assembling these results together, we have the following:

$$\begin{aligned} p_t &= f'(t) \cdot e^{g(t)} + f(t) \cdot e^{g(t)} \cdot g'(t) \\ &= -2\pi D\ (4\pi Dt)^{-3/2} \cdot e^{-x^2/(4Dt)} + \frac{1}{\sqrt{4\pi Dt}} \cdot e^{-x^2/(4Dt)} \cdot \frac{x^2}{4Dt^2} \\ &= \frac{1}{\sqrt{4\pi Dt}} \cdot e^{-x^2/(4Dt)} \left(-2\pi D\ (4\pi Dt)^{-1} + \frac{x^2}{4Dt^2}\right) \\ &= \frac{1}{\sqrt{4\pi Dt}} \cdot e^{-x^2/(4Dt)} \left(-\frac{1}{2t} + \frac{x^2}{4Dt^2}\right) \\ &= p(x,t) \left(-\frac{1}{2t} + \frac{x^2}{4Dt^2}\right) \end{aligned}$$

Wow. Verifying Equation (24.4) is a solution to the diffusion equation is getting complicated, but also notice that through algebraic simplification, $p_t = p(x,t)\left(\dfrac{x^2 - 2Dt}{4Dt^2}\right)$. If we compare p_t to Dp_{xx}, they are equal!

The connections between diffusion and probability are so strong. Equation (24.4) is related to the formula for a normal probability density function (you might want to refer back to Section 9)! In this case, the standard deviation in Equation (24.4) dependent on time (Exercise 24.2). Even though we approached the random walk differently here compared to Section 23, we also saw that the variance grew proportional to the time spent, so there is some consistency.

24.2 Concluding Thoughts

You may be wondering how this discussion of random walks connects back into stochastic differential equations. With the ideas of a random walk developed here and in Section 23, we can now understand how small changes in a variable or parameter affect the solutions to a differential equation.

For example if we consider the following logistic differential equation $\dfrac{dx}{dt} = rx\left(1 - \dfrac{x}{K}\right)$, a naive way to add stochasticity is to add an additional term (which we call "Noise")

$$\frac{dx}{dt} = rx\left(1 - \frac{x}{K}\right) + \text{ Noise } . \tag{24.5}$$

One way that we examine this is by multiplying the dt term over to the right hand side:

$$dx = rx\left(1 - \frac{x}{K}\right)\ dt + \text{ Noise }\ dt \tag{24.6}$$

The first term ($rx\left(1 - \frac{x}{K}\right)\ dt$) is called the "deterministic part" of the equation. The second term (Noise dt) is the "stochastic part." If this "Noise" term represents a random walk, this will affect the solution trajectory. However, we would expect that the *average* ensemble of solutions to behave similarly to the deterministic solution.

A model for this Noise process ties directly into Equation (24.4). and how to build this into our simulation models will be the focus of the final sections.

24.3 Exercises

Exercise 24.1. Through direct computation, verify the following calculations:

a. When $f(t) = \dfrac{1}{\sqrt{4\pi Dt}}$, then $f'(t) = -2\pi D(4\pi Dt)^{-3/2}$

b. When $g(t) = \dfrac{-x^2}{4Dt}$, then $g'(t) = \dfrac{x^2}{4Dt^2}$

c. Verify that $\left(-\dfrac{1}{2t} + \dfrac{x^2}{4Dt^2}\right) = \left(\dfrac{x^2 - 2Dt}{(2Dt)^2}\right)$

Exercise 24.2. The equation for the normal distribution is $f(x) = \dfrac{1}{\sqrt{2\pi}\sigma} e^{-(x-\mu)^2/(2\sigma^2)}$, with mean μ and variance σ^2. Examine the formula for the diffusion equation (Equation (24.4)) and compare it to the formula for the normal distribution. If Equation (24.4) represents a normal distribution, what does μ equal? σ^2?

Exercise 24.3. For this problem you will investigate $p(x,t)$ (Equation @ref(eq:diffusion-equation}) with $D = \dfrac{1}{2}$.

a. Evaluate $\displaystyle\int_{-1}^{1} p(x,10)\,dx$. Write a one sentence description what this quantity represents.

b. Using desmos or some other numerical integrator, complete the following table:

Equation	Result
$\displaystyle\int_{-1}^{1} p(x,10)\,dx =$	
$\displaystyle\int_{-1}^{1} p(x,5)\,dx =$	
$\displaystyle\int_{-1}^{1} p(x,2.5)\,dx =$	
$\displaystyle\int_{-1}^{1} p(x,1)\,dx =$	
$\displaystyle\int_{-1}^{1} p(x,0.1)\,dx =$	
$\displaystyle\int_{-1}^{1} p(x,0.01)\,dx =$	
$\displaystyle\int_{-1}^{1} p(x,0.001)\,dx =$	

c. Based on the evidence from your table, what would you say is the value of $\displaystyle\lim_{t\to 0^+} \int_{-1}^{1} p(x,t)\,dx$?

d. Now make graphs of $p(x,t)$ at each of the values of t in your table. What would you say is occuring in the graph as $\displaystyle\lim_{t\to 0^+} p(x,t)$? Does anything surprise you? (The results you computed here lead to the foundation of what is called the Dirac delta function.)

Exercise 24.4. Consider the function $p(x,t) = \dfrac{1}{\sqrt{4\pi Dt}} e^{-x^2/(4Dt)}$. Let $x = 1$.

a. Explain in your own words what the graph $p(1,t)$ represents as a function of t.

b. Graph several profiles of $p(1,t)$ when $D = 1$, 2, and 0.1. How does the value of D affect the profile?

Exercise 24.5. Consider the function $p(x,t) = \dfrac{1}{\sqrt{\pi t}} e^{-x^2/t}$:

a. Using your differentiation skills compute the partial derivatives p_t, p_x, and p_{xx}.

b. Verify $p(x,t)$ is consistent with the diffusion equation $p_t = \dfrac{1}{4} p_{xx}$.

Exercise 24.6. For the one-dimensional random walk we discussed where there was an equal chance of moving to the left or the right. Here is a variation on this problem.

Let's assume there is a chance v that it moves to the left (position $x - \Delta x$), and therefore a chance is $1 - v$ that the particle remains at position x. The basic equation that describes the particle's position at position x and time $t + \Delta t$ is:

$$p(x, t + \Delta t) = (1 - v) \cdot p(x, t) + v \cdot p(x - \Delta x, t) \tag{24.7}$$

Apply the techniques of local linearization in x and t to show that this random walk to derive the following partial differential equation, called the *advection equation*:

$$p_t = -\left(v \cdot \frac{\Delta x}{\Delta t}\right) \cdot p_x \tag{24.8}$$

Note: you only need to expand this equation to first order

Chapter 25

Stochastic Differential Equations

In this section we will begin to combine our knowledge of random walks to numerically simulate *stochastic differential equations*, or SDEs for short. Here is the good news: much of the content from the previous sections comes into focus here. We are going to focus on a model that you know (a logisitic differential equation) to develop general principles for working with other SDEs.

25.1 The stochastic logistic model

In Section 24 we started to write down the format of a stochastic differential equation, which we will use the logistic equation for context:

$$dx = rx\left(1 - \frac{x}{K}\right)\, dt + \text{ Noise } \, dt \tag{25.1}$$

It is helpful to identify the two different parts of Equation (25.1). The first part is called the *deterministic part*, and it does not involve the "Noise" term: $rx\left(1 - \frac{x}{K}\right)\, dt$. The second part is called the *stochastic part* and is (I bet you guessed it!) the term that contains Noise dt is the "stochastic part."

While just writing Noise dt doesn't seem mathematical, its just a substitute for the type of stochastic process we have. For our purposes we are only going to consider random walks (a.k.a white noise a.k.a. Weiner processes), which we represent in shorthand with $dW(t)$, so that $dW(t) =$ Noise dt. This allows us to re-write Equation (25.1) as a more formal SDE (Equation (25.2)).

$$dx = rx\left(1 - \frac{x}{K}\right)\, dt + dW(t) \tag{25.2}$$

Just to be clear the term $dW(t)$ in Equation (25.2) is short hand to the differential equation $\frac{dW}{dt} =$ Noise , where $W(t)$ is the solution to a Weiner process. This white noise has the following characteristics:

- $W(t)$ is continuous
- $W(0) = 0$
- $W(t) - W(s)$ independent (they call this independent increments)
- $W(t) - W(s)$ is normally distributed with mean 0 and standard deviation $\sqrt{t-s}$.

It does seem odd to write a differential equation in this form (i.e. $dx = ...$ versus $\frac{dx}{dt} = ...$). but a good way to think of this stochastic differential equation is that a small change in the variable x (represented by the term dx) is computed in two ways:

$$\text{Deterministic part: } rx\left(1 - \frac{x}{K}\right)\, dt$$
$$\text{Stochastic part: } dW(t)$$

How does the stochastic part of this differential equation change the solution trajectory? It turns out that the "exact" solutions to problems like these are difficult (we will study a sample of them in 27). Rather than focus on an exact solution technique we

are going to focus on how to apply a numerical method to simulate solution trajectories and then take the ensemble average each of the time points.

25.2 The Euler-Maruyama Method

The Euler-Maruyama method is a variation of Euler's method that accounts for stochasticity and implements the random walk. We are going to build this up step by step.

First we write the dx term as a difference equation: $dx = x_{n+1} - x_n$, where n is the current step of Euler's method. Likewise $dW(t) = W_{n+1} - W_n$, which represents one step of the random walk. We will approximate $W_{n+1} - W_n = \sqrt{\Delta t} Z_1$, where Z_1 is a random number from the standard unit normal distribution (done in `R` as `rnorm(1)`) and Δt is the timestep length. When we put these together we have the following iterative method:

- Given Δt and starting value x_0,
- Then compute the next step: $x_1 = x_0 + r x_0 \left(1 - \frac{x_0}{K}\right) \Delta t + \sqrt{\Delta t}\, Z_1$
- Repeat to step n: $x_1 = x_0 + r x_0 \left(1 - \frac{x_0}{K}\right) \Delta t + \sqrt{\Delta t}\, Z_1$

That is it! We can apply this numerical method for as many steps as we want. I have some defined code that will apply the simulation, but just like `euler` or `systems` there are some things that need to be set first. In order to apply the Euler-Maruyama method you will need to define six things:

- The size (Δt) of your timesteps.
- The number of timesteps you wish to run the method. More timesteps means more computational time. If N is the number of timesteps, $\Delta t \cdot N$ is the total time.
- A function for our deterministic dynamics. For Equation (25.2) this equals $rx\left(1 - \frac{x}{K}\right)$.
- A function for our stochastic dynamics. For Equation (25.2) this equals 1.
- The values of the vector of parameters $\vec{\alpha}$. For the logistic differential equation we will take $r = 0.8$ and $K = 100$.
- The standard deviation (σ) of our normal distribution and random walk. Typically this is set to 1, but can be varied if needed.

Sample code for this stochastic differential equation is shown below, with the resulting trajectory of the solution in Figure 25.1.

```
# Identify the deterministic and stochastic parts of the DE:
deterministic_logistic <- c(dx ~ r*x*(1-x/K))
stochastic_logistic <-  c(dx ~ 1)

# Identify the initial condition and any parameters
init_logistic <- c(x=3)  # Be sure you have enough conditions as you do variables.
logistic_parameters <- c(r=0.8, K=100)   # parameters: a named vector

# Identify how long we run the simulation
deltaT_log <- .05    # timestep length
timeSteps_log <- 200   # must be a number greater than 1

# Identify the standard deviation of the stochastic noise
sigma_log <- 1

# Do one simulation of this differential equation
logistic_out <- euler_stochastic(deterministic_rate = deterministic_logistic,
                                 stochastic_rate = stochastic_logistic,
                                 init_cond = init_logistic,
                                 parameters = logistic_parameters,
                                 deltaT = deltaT_log,
                                 n_steps = timeSteps_log,
                                 sigma = sigma_log)

# Plot out the solution
ggplot(data = logistic_out) +
  geom_line(aes(x=t,y=x))
```

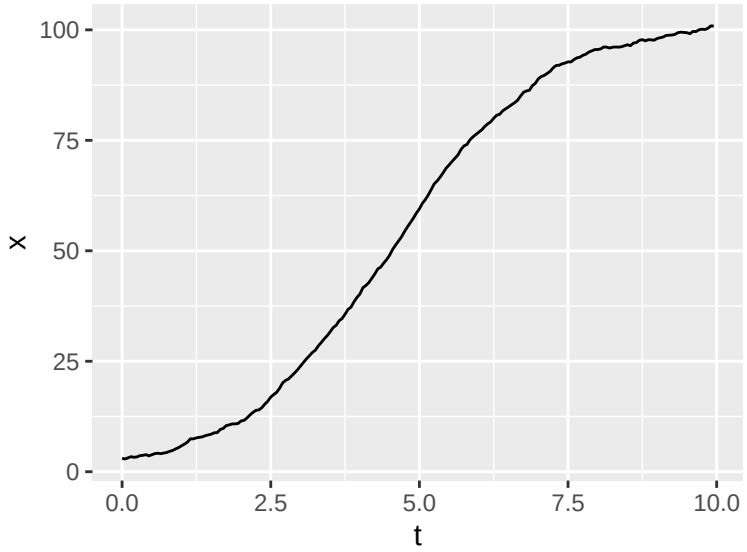


Figure 25.1: One realization of Equation (25.1)

Let's break this code down step by step:

- We identify the deterministic and stochastic parts to our differential equation with the variables `deterministic_logistic` and `stochastic_logistic`. The same structure is used for Euler's method from Section 4
- Similar with Euler's method we need to identify the initial conditions (`init_logistic`), parameters (`logistic_parameters`), Δt (`deltaT_log`), and number of timesteps (`timeSteps_log`).
- The standard deviation of the stochastic noise (σ) is represented with `sigma_log`.
- The command `euler_stochastic` does one realization of the Euler-Maruyama method, returning a vector that we can then plot.

25.2.1 Multiple simulations

Figure 25.1 shows one sample trajectory of our solution. If you re-run this code several times and replot the solution you may see different trajectories plotted. In Section 22 we discussed the idea of stochastic simulation and how to re-run code several times and compute the ensemble average. We can put those ideas to good use here:

```
# Many solutions
n_sims <- 100  # The number of simulations

# Compute solutions
logistic_run <- rerun(n_sims) %>%
  set_names(paste0("sim", 1:n_sims)) %>%
  map(~ euler_stochastic(deterministic_rate = deterministic_logistic,
                         stochastic_rate = stochastic_logistic,
                         init_cond = init_logistic,
                         parameters = logistic_parameters,
                         deltaT = deltaT_log,
                         n_steps = timeSteps_log,
                         sigma = sigma_log)) %>%
  map_dfr(~ .x, .id = "simulation")


# Plot these all up together
  ggplot(data = logistic_run) +
  geom_line(aes(x=t, y=x, color = simulation)) +
  ggtitle("Spaghetti plot for the logistic SDE") +
  guides(color="none")
```

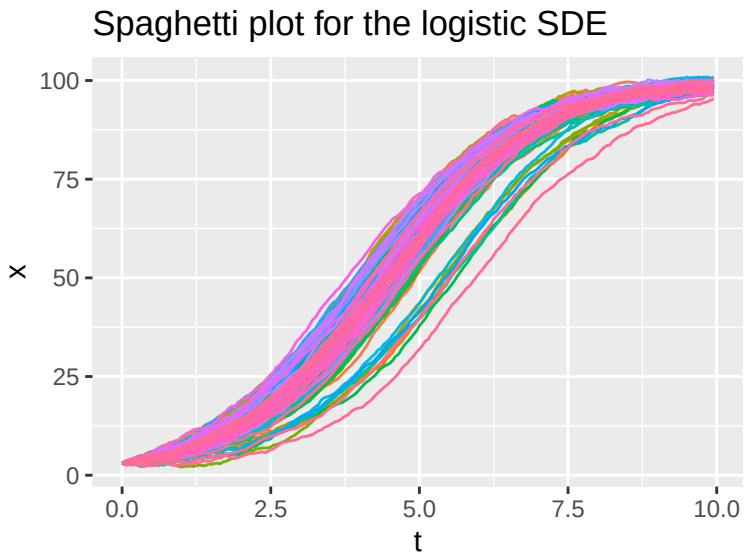


Figure 25.2: Several different realizations of the logistic SDE.

As you may recall, the main engine of the code is contained in the `map( ~ euler_stochastic ... )` which re-runs the codes for the number of times specified in `n_sims`.

Figure 25.3 shows the ensemble average (median and 95% confidence interval) for the different simulations, recycling code from Section 22.

```
### Summarize the variables
summarized_logistic <- logistic_run %>%
  group_by(t) %>%  # All simulations will be grouped at the same timepoint.
  summarise(x = quantile(x, c(0.25, 0.5, 0.75)), q = c("q0.025", "q0.5", "q0.975")) %>%
  pivot_wider(names_from = q, values_from = x)
```

```
## `summarise()` has grouped output by 't'. You can override using the `.groups` argument.
```

```
### Make the plot
ggplot(data = summarized_logistic) +
  geom_line(aes(x = t, y = q0.5)) +
  geom_ribbon(aes(x=t,ymin=q0.025,ymax=q0.975),alpha=0.2) +
  ggtitle("Ensemble average plot for the logistic SDE")
```

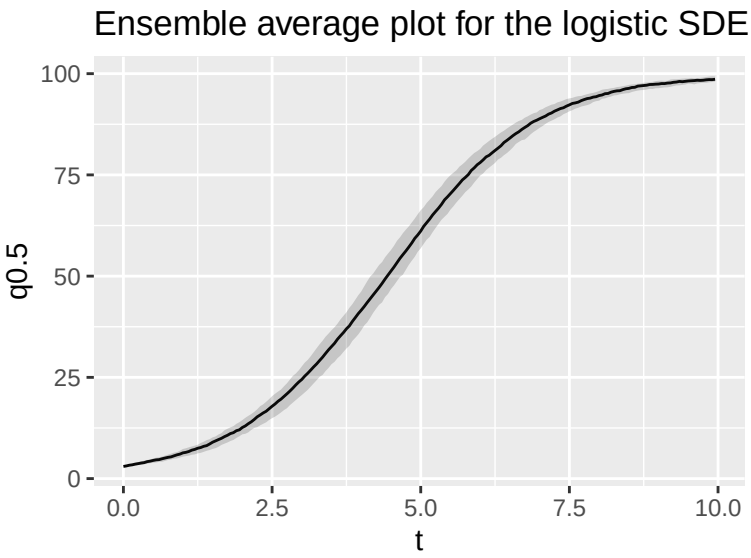


Figure 25.3: Ensemble average plot of the stochastic logistic SDE.

Breaking this code down, the variable `summarized_logistic` first groups the simulations by the variable `t` in order to compute the quantiles across each of the simulations. We then pivot resulting data frame in order to plot the ribbon.

25.3 Adding stochasticity to parameters

Now that we have seen an example in adding stochasticity to the logistic equation, we can also parameters of the differential equation to be stochastic. For example, let's say that the growth rate r in the logistic equation is subject to stochastic effects. How we would implement this by replacing r with $r +$ Noise :

$$dx = (r + \text{ Noise }) \, x \left(1 - \frac{x}{K}\right) \, dt, \tag{25.3}$$

Now what we do is separate out the terms that are multiplied by "Noise" - they will form the stochastic part of the differential equation. The terms that aren't multipled by "Noise" form the deterministic part of the differential equation:

$$dx = rx \left(1 - \frac{x}{K}\right) \, dt + x \left(1 - \frac{x}{K}\right) \text{ Noise } \, dt, \tag{25.4}$$

So we have the following, writing Noise $dt = dW(t)$:

$$\text{Deterministic part: } rx \left(1 - \frac{x}{K}\right) \, dt$$
$$\text{Stochastic part: } x \left(1 - \frac{x}{K}\right) dW(t)$$

There are a few things to notice here. First, the deterministic part of the differential equation is what we would expect without noise added. Second, notice how the stochastic part of the differential equation changed. Let's take a look at the simulations with the Euler-Maruyama method, denoting the deterministic and stochastic parts as `deterministic_logistic_r` and `stochastic_logistic_r` respectively:

```
# Identify the deterministic and stochastic parts of the DE:
deterministic_logistic_r <- c(dx ~ r*x*(1-x/K))
stochastic_logistic_r <-  c(dx ~ x*(1-x/K))

# Identify the initial condition and any parameters
init_logistic <- c(x=3)  # Be sure you have enough conditions as you do variables.
logistic_parameters <- c(r=0.8, K=100)   # parameters: a named vector

# Identify how long we run the simulation
deltaT_log <- .05    # timestep length
timeSteps_log <- 200   # must be a number greater than 1

# Identify the standard deviation of the stochastic noise
sigma_log <- 1

# Do one simulation of this differential equation
logistic_out <- euler_stochastic(deterministic_rate = deterministic_logistic_r,
                            stochastic_rate = stochastic_logistic_r,
                            init_cond = init_logistic,
                            parameters = logistic_parameters,
                            deltaT = deltaT_log,
                            n_steps = timeSteps_log,
                            sigma = sigma_log)

# Plot out the solution
ggplot(data = logistic_out) +
  geom_line(aes(x=t,y=x))
```

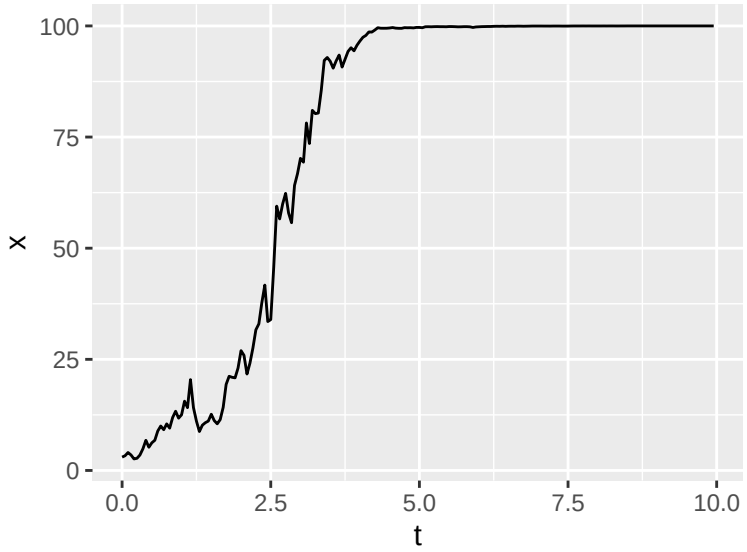


Figure 25.4: One realization of the logistic differential equation with stochasticity in the parameter r.

As we did before, we can run multiple iterations of this stochastic process. We will also compute and plot the ensemble average.

```
# Many solutions
n_sims <- 100  # The number of simulations

# Compute solutions
logistic_run_r <- rerun(n_sims) %>%
  set_names(paste0("sim", 1:n_sims)) %>%
  map(~ euler_stochastic(deterministic_rate = deterministic_logistic_r,
                         stochastic_rate = stochastic_logistic_r,
                         init_cond = init_logistic,
                         parameters = logistic_parameters,
                         deltaT = deltaT_log,
                         n_steps = timeSteps_log,
                         sigma = sigma_log)
) %>%
  map_dfr(~ .x, .id = "simulation")


# Plot these all up together
  ggplot(data = logistic_run_r) +
  geom_line(aes(x=t, y=x, color = simulation)) +
  ggtitle("Spaghetti plot for the logistic SDE") +
  guides(color="none")
```

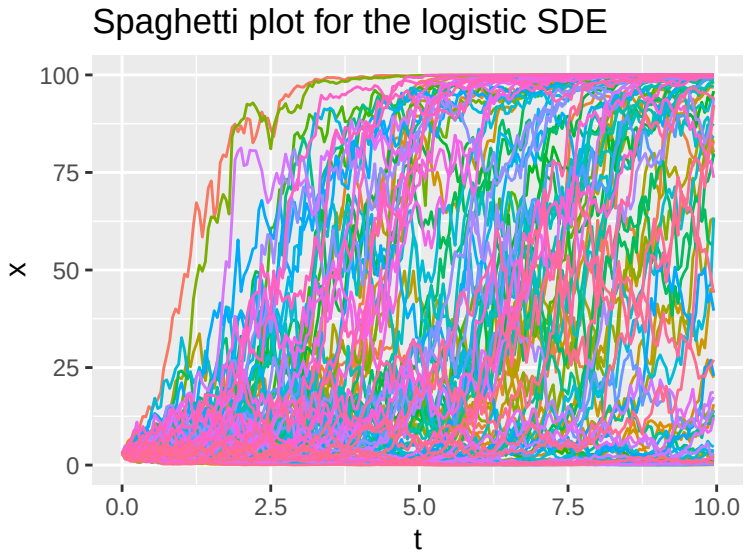


Figure 25.5: Several different realizations of the logistic SDE with stochasticity in the parameter r, along with the ensemble average plot.

```
  ### Summarize the variables
summarized_logistic_r <- logistic_run_r %>%
  group_by(t) %>%  # All simulations will be grouped at the same timepoint.
  summarise(x = quantile(x, c(0.25, 0.5, 0.75)), q = c("q0.025", "q0.5", "q0.975")) %>%
  pivot_wider(names_from = q, values_from = x)
```

```
## `summarise()` has grouped output by 't'. You can override using the `.groups` argument.
```

```
### Make the plot
ggplot(data = summarized_logistic_r) +
  geom_line(aes(x = t, y = q0.5)) +
  geom_ribbon(aes(x=t,ymin=q0.025,ymax=q0.975),alpha=0.2) +
  ggtitle("Ensemble average plot for the logistic SDE")
```

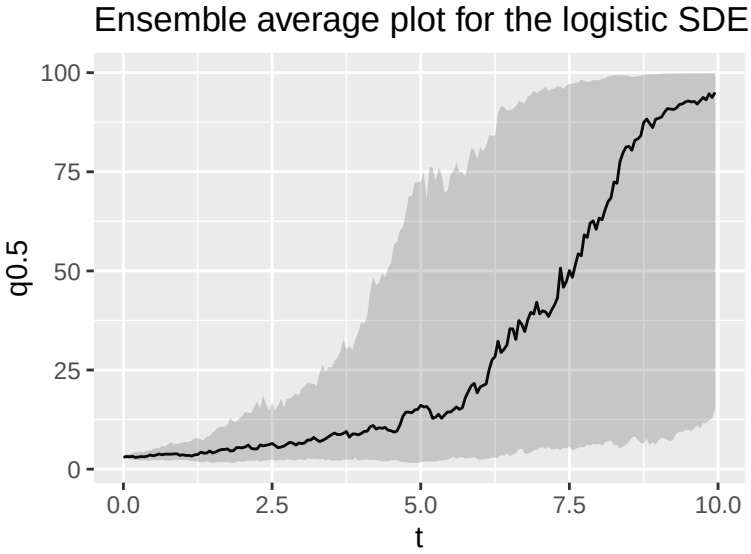


Figure 25.6: Several different realizations of the logistic SDE with stochasticity in the parameter r, along with the ensemble average plot.

Wow! Adding in stochasticity to the parameters changes things. Notice how the dynamics are different - there is a lot more variability in how quickly the solution rises to the steady state of $K = 100$.

25.4 Concluding thoughts

If you start with a known differential equation and want to add stochasticity to a parameter, here is a process:

- Replace whatever parameter with a "parameter + Noise" term (i.e $a \to a +$ Noise).
- Collect terms multiplied by Noise - they will form the stochastic part of the differential equation.
- The deterministic part of the differential equation should be your original differential equation.

The most general form of the stochastic differential equation is: $d\vec{y} = f(\vec{y}, \vec{\alpha}, t)\ dt + g(\vec{y}, \vec{\alpha}, t)\ dW(t)$, where $\vec{y}$ is the vector of state variables you want to solve for, and $\vec{\alpha}$ is your vector of parameters, and $dW(t)$ is the stochastic noise from the random walk.

At a given initial condition, the Euler-Maruyama method applies locally linear approximations to forecast the solution forward Δt time units: $\vec{y}_{n+1} = y_n + f(\vec{y}_n, \vec{\alpha}, t_n) \cdot \Delta t + g(\vec{y}_n, \vec{\alpha}, t_n) \cdot \sigma \cdot \text{rnorm(N)} \cdot \sqrt{\Delta t}$, where `rnorm(N)` is N dimensional random variable from a normal distribution with mean 0.

25.5 Exercises

Exercise 25.1. Consider the logistic differential equation (Equation (25.2)). In this section we set $\sigma = 1$. Re-run the code to generate one simulation with $\sigma = 0.01,\ 2,\ 10$. In each case, how does changing σ affect the simulation run?

Exercise 25.2. Consider the logistic differential equation (Equation (25.2)). In this section we set $\sigma = 1$. Re-run the code to generate 100 simulations with $\sigma = 0.01,\ 2,\ 10$ and then compute the ensemble. In each case, how does changing σ affect the ensemble average?

Exercise 25.3. Return back to the example of adding stochasticity to the parameter r in the logistic differential equation and the resulting spaghetti and ensemble average plots. For these plots we set $\sigma = 1$. What happens to the resulting spaghetti and ensemble plots when $\sigma = 0.01,\ 0.1,\ 10$?

Exercise 25.4. (Inspired by Logan and Wolesensky (2009)) Consider the logistic differential equation: $\dfrac{dx}{dt} = rx\left(1 - \dfrac{x}{K}\right)$. Assume there is stochasticity in the inverse carrying capacity $1/K$ (so this means you will consider $1/K +$ Noise).

a. Identify the deterministic and stochastic part of each of the differential equation.
b. Assume that $x(0) = 3$, $r = 0.8$, $K = 100$, $\Delta t = 0.05$, and $\sigma = 1$. Apply the Euler-Maruyama method to produce a solution, using with 200 timesteps.
c. Now do 500 simulations of this stochastic process and compare the ensemble solution.
d. Contrast your results to when we added stochasticity to the parameter r in the logistic model.

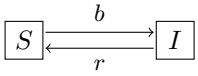


Figure 25.7: The SIS model

Exercise 25.5. (Inspired by Logan and Wolesensky (2009)) An SIS model is one where susceptibles S become infected I, and then after recovering from an illness, become susceptible again. The schematic representing this is shown in Figure 25.7. While you can write this as a system of differential equations, assuming the population size is constant N, this simplifies to the following differential equation:

$$\frac{dI}{dt} = b(N - I)I - rI \tag{25.5}$$

a. Determine the equilibrium solutions for this model and analyze the stability of the equilibrium solutions.
b. Assuming $N = 1000$, $r = 0.01$, and $b = 0.005$, $I(0) = 1$, apply Euler's method to simulate this differential equation over two weeks with $\Delta t = 0.1$. Show the plot of your result.
c. Assume the transmission rate b is stochastic. Write down this stochastic differential equation. Do 500 simulations of this stochastic process with $\sigma = 1$. Contrast this result to the deterministic solution.
d. Assume the recovery rate r is stochastic. Write down this stochastic differential equation. Do 500 simulations of this stochastic process with $\sigma = 1$. Contrast this result to the previous results.

Exercise 25.6. Consider the following Lotka-Volterra (predator prey) model:

$$\begin{aligned} \frac{dV}{dt} &= rV - kVP \\ \frac{dP}{dt} &= ekVP - dP \end{aligned} \tag{25.6}$$

a. Assume that the parameter k is stochastic. Write down the stochastic differential equation, identifying the deterministic and stochastic parts to this system of equations.

b. Apply the Euler-Maruyama method for 100 simulations with $\sigma = 0.01$ with the following values of parameters and step sizes:

- Initial condition: $V(0) = 1$, $P(0) = 3$
- Parameters: $r = 2$, $k = 0.5$, $e = 0.1$, and $d = 1$.
- Set $\Delta t = 0.05$ and $N = 200$.

Exercise 25.7. Consider the following model for zombie population dynamics:

$$\begin{aligned} \frac{dS}{dt} &= -\beta SZ - \delta S \\ \frac{dZ}{dt} &= \beta SZ + \xi R - \alpha SZ \\ \frac{dR}{dt} &= \delta S + \alpha SZ - \xi R \end{aligned} \tag{25.7}$$

a. Let's assume the transmission rate β is a stochastic parameter. With this assumption, group each differential equation into two parts: terms not involving noise (the deterministic part) and terms that are multiplied by noise (the stochastic part)

- Deterministic part for $\frac{dS}{dt}$:
- Stochastic part for $\frac{dS}{dt}$:
- Deterministic part for $\frac{dZ}{dt}$:
- Stochastic part for $\frac{dZ}{dt}$:
- Deterministic part for $\frac{dR}{dt}$:
- Stochastic part for $\frac{dR}{dt}$:

b. Apply the Euler-Maruyama method to do 500 simulations of this stochastic differential equation and compute the ensemble average. For the Euler-Maruyama method apply the following values:

- $\sigma = 0.004$
- $\Delta t = 0.5$.
- Timesteps: 200.
- $\beta = 0.0095$, $\delta = 0.0001$,$\xi = 0.1$, $\alpha = 0.005$.
- Initial condition: $S(0) = 499$, $Z(0) = 1$, $R(0) = 0$.

c. How does making β stochastic affect the disease transmission? d. If we assume that the population is fixed at 500 individuals, what is interesting about your stochastic results?

Chapter 26

Simulating Stochastic Dynamics

In Section 25 we built up a stochastic differential equation from adding in stochasticity (noise) to the parameter values. Another approach is to assume instead the variables are stochastic. Here is the good news: we can still simulate SDEs with stochastic variables, setting them up in a similar way to Section 25. We will generate realizations and then compute the ensemble averages.

26.1 The stochastic logistic model redux

Let's go back to the logistic population model but re-written in a specific way:

$$\frac{dx}{dt} = rx\left(1 - \frac{x}{K}\right) = rx - \frac{rx^2}{K} \tag{26.1}$$

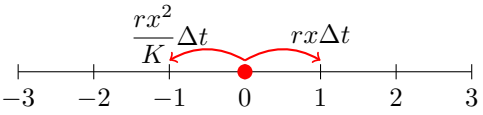
From Equation (26.1), we can obtain a change in the variable x (denoted as Δx) over Δt units by re-writing the differential equation in differential form:

$$\Delta x = rx\Delta t - \frac{rx^2}{K}\Delta t \tag{26.2}$$

$\Delta x = rx\Delta t - \frac{rx^2}{K}\Delta t$. Equation (26.2) is separated into two terms - one that increases the variable x (represented by $rx\Delta t$, same units as x) and one that decreases the variable (represented by $\frac{rx^2}{K}\Delta t$, same units as x). We will consider these changes as on a unit scale, organized via the following table:

Outcome	Probability
$\Delta x = 1$ (population change by 1)	$rx\ \Delta t$
$\Delta x = -1$ (population change by -1)	$\frac{rx^2}{K}\ \Delta t$
$\Delta x = 0$ (no population change)	$1 - rx\ \Delta t - \frac{rx^2}{K}\ \Delta t$

It also may be helpful to think of these changes on a random walk number line:



It may seem odd to think of the different outcomes (Δx equals 1, -1, or 1) as probabilities. Part of the reason why that formulation is useful is to apply concepts from probability theory. Let Y be a random variable with a finite number of finite outcomes $y_1, y_2, \ldots, y_k$ with probabilities $p_1, p_2, \ldots, p_k$, respectively, then the expected value μ of Y is:

$$\mu = E[Y] = \sum_{i=1}^{k} y_i \, p_i = y_1 p_1 + y_2 p_2 + \cdots + y_k p_k.$$

If we apply this definition to the random variable Δx we have:

$$\begin{aligned} \mu = E[\Delta x] &= (1) \cdot \Pr(\Delta x = 1) + (-1) \cdot \Pr(\Delta x = -1) + (0) \cdot \Pr(\Delta x = 0) \\ &= (1) \cdot (rx \ \Delta t) + (-1) \frac{rx^2}{K} \ \Delta t \\ &= rx \ \Delta t - \frac{rx^2}{K} \ \Delta t \end{aligned} \tag{26.3}$$

Notice that the right hand side of Equation (26.3) is the same as the right hand side of the original differential equation (Equation (26.2)))!

Next let's also calculate the variance of Δx, defined for a discrete random variable:

$$\sigma^2 = E[(Y - \mu)^2] = \sigma^2 = E[Y^2] - (E[Y])^2.$$

$$\begin{aligned} \sigma^2 = E[(\Delta x)^2] - (E[\Delta x])^2 &= (1)^2 \cdot \Pr(\Delta x = 1) + (-1)^2 \cdot \Pr(\Delta x = -1) + (0)^2 \cdot \Pr(\Delta x = 0) - (E[\Delta x])^2 \\ &= (1) \cdot (rx \ \Delta t) + (1) \frac{rx^2}{K} \ \Delta t - \left(rx \ \Delta t - \frac{rx^2}{K} \ \Delta t \right)^2 \end{aligned} \tag{26.4}$$

Along with the computation of the variance, We are going to compute the variance to first order in Δt. Because of that, we are going to assume that in σ^2 any terms involving $(\Delta t)^2$ are small, or in effect negligible. While this is a huge simplifying assumption for the variance, but it is useful!

Doing these calcuations we have that $\sigma^2 = (rx \ \Delta t) + \frac{rx^2}{K} \ \Delta t$. Computing the mean and variance will help characterize the ensemble average. In addition the following random walk properties can be applied:

- Since Δx is the sum of many smaller changes we can apply the central limit theorem to characterize Δx as a normal random variable.
- To first order, we assume that Δx follows a probability distribution that is normal with mean μ and variance σ^2.
- The distribution for Δx can be expressed as $\Delta x = \mu + \sigma Z$, where Z is random variable from a unit normal distribution (so in `R` we would use `rnorm(1)`).
- We can simulate Δx as a Wiener process.
- Since $\Delta x = x_{n+1} - x_n$, then we have $x_{n+1} = x_n + \mu + \sigma Z$.

Notice how the last step provides a way to generate a solution trajectory to our differential equation. Cool! To simulate this stochastic process we will use the function `birth_death_stochastic`, which is set up in a similar way to `euler_stochastic` from Section 25. I will append `_log` to denote "logistic" for each of the parts

```
# Identify the birth and death parts of the DE:
birth_rate_log <- c(dx ~ r*x)
death_rate_log <-  c(dx ~ r*x^2/K)

# Identify the initial condition and any parameters
init_log <- c(x=3)  # Be sure you have enough conditions as you do variables.
parameters_log <- c(r=0.8, K=100)   # parameters: a named vector

# Identify how long we run the simulation
deltaT_log <- .05    # timestep length
time_steps_log <- 200   # must be a number greater than 1

# Identify the standard deviation of the stochastic noise
```

```
sigma_log <- 1

# Do one simulation of this differential equation
out_log <- birth_death_stochastic(birth_rate = birth_rate_log,
                                  death_rate = death_rate_log,
                                  init_cond = init_log,
                                  parameters = parameters_log,
                                  deltaT = deltaT_log,
                                  n_steps = time_steps_log,
                                  sigma = sigma_log)

# Plot out the solution
ggplot(data = out_log) +
  geom_line(aes(x=t,y=x))
```

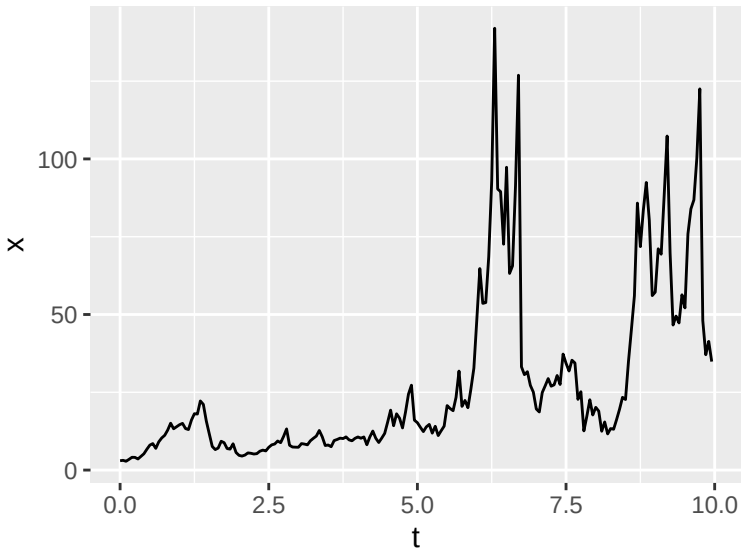


Figure 26.1: One realization of the logistic differential equation with stochastic variables.

Notice how the resulting spaghetti plot shows variation in the variables.

Making an ensemble average plot is also similar to how we computed them in Section 25:

```
# Many solutions
n_sims <- 100  # The number of simulations

# Compute solutions
logistic_sim_r <- rerun(n_sims) %>%
  set_names(paste0("sim", 1:n_sims)) %>%
  map(~ birth_death_stochastic(birth_rate = birth_rate_log,
                               death_rate = death_rate_log,
                               init_cond = init_log,
                               parameters = parameters_log,
                               deltaT = deltaT_log,
                               n_steps = time_steps_log,
                               sigma = sigma_log)
) %>%
  map_dfr(~ .x, .id = "simulation")


# Plot these all up together
  ggplot(data = logistic_sim_r) +
```

```
  geom_line(aes(x=t, y=x, color = simulation)) +
  ggtitle("Spaghetti plot for the logistic SDE") +
  guides(color="none")
```

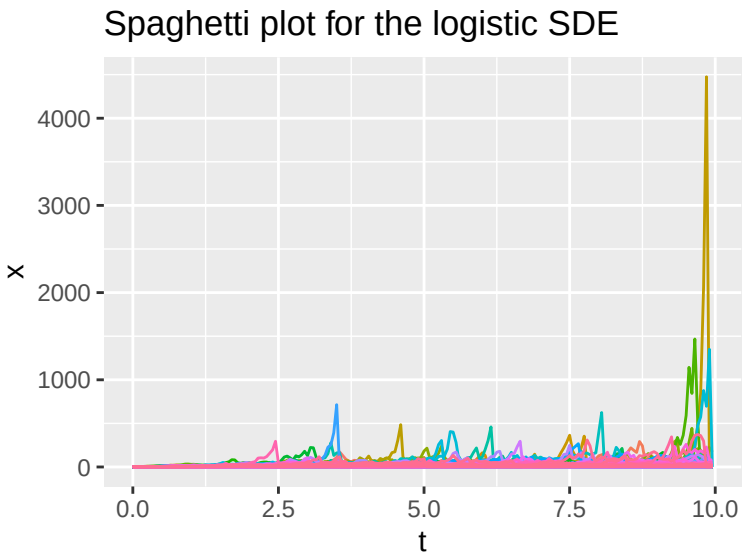


Figure 26.2: Several different realizations of the logistic SDE with stochasticity in the variables, along with the ensemble average plot.

```
  ### Summarize the variables
summarized_logistic_sim_r <- logistic_sim_r %>%
  group_by(t) %>%  # All simulations will be grouped at the same timepoint.
  summarise(x = quantile(x, c(0.25, 0.5, 0.75)), q = c("q0.025", "q0.5", "q0.975")) %>%
  pivot_wider(names_from = q, values_from = x)
```

```
## `summarise()` has grouped output by 't'. You can override using the `.groups` argument.
```

```
### Make the plot
ggplot(data = summarized_logistic_sim_r) +
  geom_line(aes(x = t, y = q0.5)) +
  geom_ribbon(aes(x=t,ymin=q0.025,ymax=q0.975),alpha=0.2) +
  ggtitle("Ensemble average plot for the logistic SDE")
```

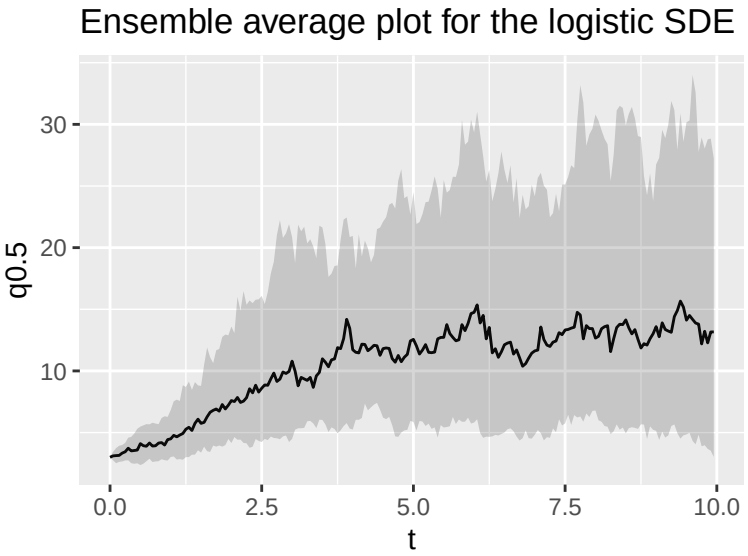


Figure 26.3: Several different realizations of the logistic SDE with stochasticity in the variables, along with the ensemble average plot.

Notice that as before the spaghetti and ensemble average plots are generated. While there are some simulations that may always be zero, in the *ensemble* the median resembles the (deterministic) solution to the logistic differential equation.

The types of stochastic processes we are describing in this section are called "birth-death" processes. Here is another way to think about Equation (26.2):

$$\begin{aligned} rx\ \Delta t &= \alpha(x)\ \text{(birth)} \\ \frac{rx^2}{K}\ \Delta t &= \delta(x)\ \text{(death)} \end{aligned} \tag{26.5}$$

In this way we think of $\alpha(x)$ in Equation (26.5) as a "birth process" and $\delta(x)$ as a "death process." When we computed the mean μ and variance σ^2 for Equation (26.2) we had $\mu = \alpha(x) - \delta(x)$ and $\sigma^2 = \alpha(x) + \delta(x)$ (you should double check to make sure this is the case!). Here is an interesting fact: $\mu = \alpha(x) - \delta(x)$ and $\sigma^2 = \alpha(x) + \delta(x)$ holds up for *any* differential equation where we have identified a birth or death process.

26.2 A stochastic system of equations

Let's examine a stochastic system of differential equations. Here we will return to the lynx-hare model:

$$\begin{aligned} \frac{dH}{dt} &= rH - bHL \\ \frac{dL}{dt} &= ebHL - dL \end{aligned} \tag{26.6}$$

In this case we still split *each equation* into the birth (α) and death (δ) parts:

$$\alpha = \begin{cases} \frac{dH}{dt}: & rH \\ \frac{dL}{dt}: & ebHL \end{cases}$$

$$\delta = \begin{cases} \frac{dH}{dt}: & bHL \\ \frac{dL}{dt}: & dL \end{cases}$$

To simulate this stochastic process, the setup of the code is similar to previous ways we solved systems of differential equations. Since we have a system of equations to compute the expected value and variance requires additional knowledge of matrix algebra, but that is already included in the function `birth_death_stochastic`

```
# Identify the birth and death parts of the DE:
birth_rate_pred <- c(dH ~ r*H, dL ~ e*b*H*L)
death_rate_pred <-  c(dH ~ b*H*L, dL ~ d*L)

# Identify the initial condition and any parameters
init_pred <- c(H=1, L=3)  # Be sure you have enough conditions as you do variables

# Identify the parameters
parameters_pred <- c(r = 2, b = 0.5, e = 0.1, d = 1)

# Identify how long we run the simulation
deltaT_pred <- .05    # timestep length
time_steps_pred <- 200   # must be a number greater than 1

# Identify the standard deviation of the stochastic noise
sigma_pred <- .1


# Do one simulation of this differential equation
out_pred <- birth_death_stochastic(birth_rate = birth_rate_pred,
                                   death_rate = death_rate_pred,
                                   init_cond = init_pred,
                                   parameters = parameters_pred,
                                   deltaT = deltaT_pred,
                                   n_steps = time_steps_pred,
                                   sigma = sigma_pred)

# Visualize the solution
ggplot(data = out_pred) +
  geom_line(aes(x=t,y=H),color='red') +
  geom_line(aes(x=t,y=L),color='blue') +
labs(x='Time',
     y='Lynx (red) or Hares (blue)')
```

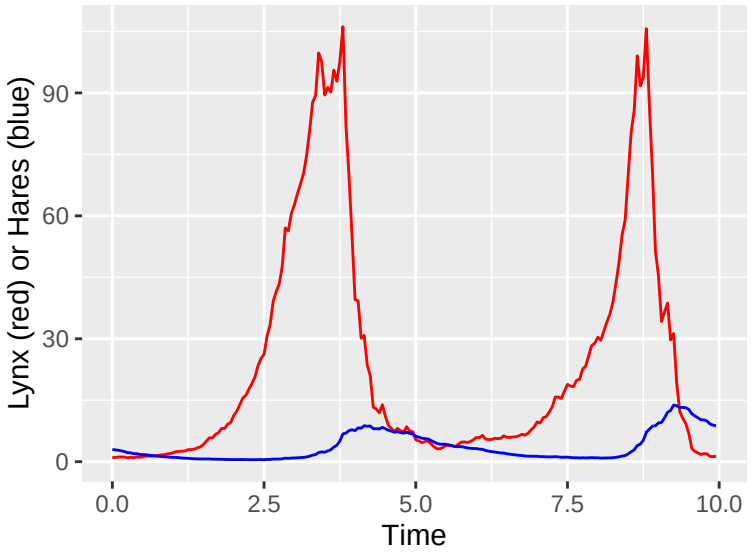


Figure 26.4: One realization of the stochastic predator-prey model.

Excellent! Notice how Figure 26.4 has stochasticity in the variables for *both* H and L. The next step would be to do many simulations and then compute the ensemble average. First let's compute the individual simulations:

```
# Many solutions
n_sims <- 100  # The number of simulations

# Compute solutions
pred_sim <- rerun(n_sims) %>%
  set_names(paste0("sim", 1:n_sims)) %>%
  map(~ birth_death_stochastic(birth_rate = birth_rate_pred,
                                        death_rate = death_rate_pred,
                                        init_cond = init_pred,
                                        parameters = parameters_pred,
                                        deltaT = deltaT_pred,
                                        n_steps = time_steps_pred,
                                        sigma = sigma_pred)
) %>%
  map_dfr(~ .x, .id = "simulation")
```

The above code may take a while to complete - but this is certainly shorter than computing these all by hand! In order to make the spaghetti plot, we will plot the variables H and L separately.

```
# Plot the hares first:
ggplot(data = pred_sim) +
  geom_line(aes(x=t,y=H,color=simulation)) +
  ggtitle("Spaghetti plot for hares in the predator-prey SDE") +
  guides(color="none")
```

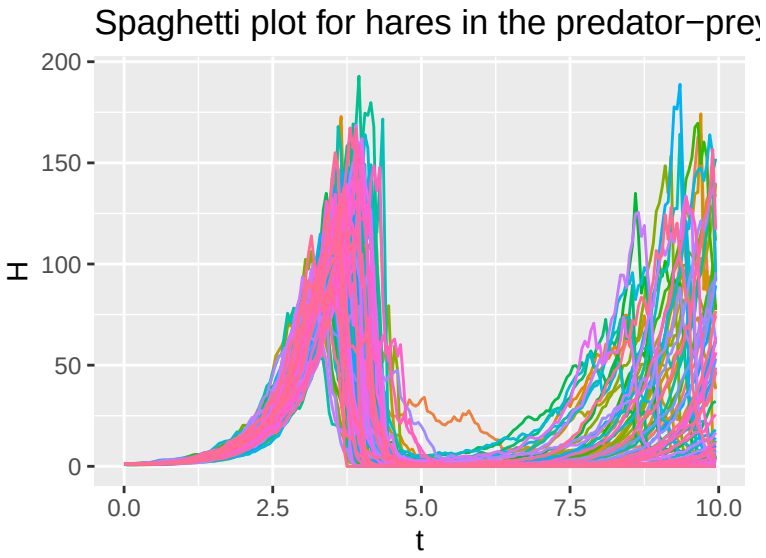


Figure 26.5: Several different realizations of the stochastic predator-prey system.

```
# Then plot the lynx:
ggplot(data = pred_sim) +
  geom_line(aes(x=t,y=L,color=simulation)) +
  ggtitle("Spaghetti plot for lynx in the predator-prey SDE") +
  guides(color="none")
```

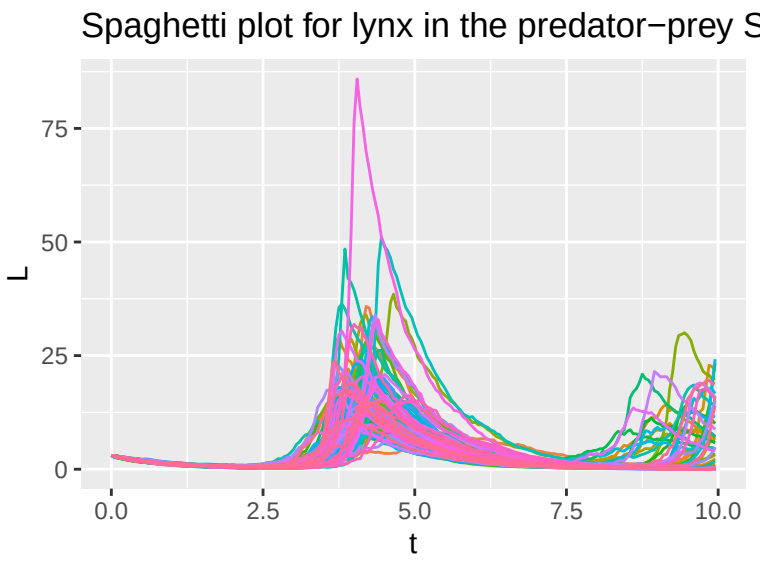


Figure 26.6: Several different realizations of the stochastic predator-prey system.

For the ensemble average plots, we will first utilize the `pivot_longer` command, group and then summarize:

```
 ### Summarize the variables
summarized_pred_sim <- pred_sim %>%
  pivot_longer(cols=c("H","L")) %>%
  group_by(name,t) %>%  # All simulations will be grouped at the same timepoint.
  summarise(value = quantile(value, c(0.25, 0.5, 0.75)), q = c("q0.025", "q0.5", "q0.975")) %>%
  pivot_wider(names_from = q, values_from = value)
```

```
## `summarise()` has grouped output by 'name', 't'. You can override using the `.groups` argument.
```

```
# Let's take a look at the resulting data frame
glimpse(summarized_pred_sim)
```

```
## Rows: 400
## Columns: 5
## Groups: name, t [400]
## $ name   <chr> "H", "H", "H", "H", "H", "H", "H", "H", "H", "H", "H", "H", "H"~
## $ t      <dbl> 0.00, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.5~
## $ q0.025 <dbl> 1.0000000, 0.9739683, 0.9704827, 0.9850577, 0.9916619, 1.031269~
## $ q0.5   <dbl> 1.000000, 1.024874, 1.046321, 1.050366, 1.075298, 1.140154, 1.1~
## $ q0.975 <dbl> 1.000000, 1.080082, 1.091934, 1.139265, 1.212339, 1.248552, 1.3~
```

Let's break this down:

- The data frame `pred_sim` has columns `simulation`, `t`, `H`, and `L`. We want to group *each* variable by the time `t`. In order to do that efficiently we need to pivot the data frame longer. By applying the command `pivot_longer(cols=c("H","L"))` we will still have 4 columns, but this time they are called `simulation`, `t`, `name`, and `value`. The column `name` is a categorical variable that is either `H` or `L` (the columns we made longer), and the column `value` has the numerical values for both at each time.
- We then group by the column `name` first, and then by `time`.
- The remainder of the code is similar to how we computed the ensemble average above.

The tricky part is that resulting columns of `summarized_pred_sim` are `name`, `t`, and the quantile values. This may seem a challenge to plot, but we can easily make a small multiples plot with the command `facet_grid`:

```
ggplot(data = summarized_pred_sim) +
  geom_line(aes(x = t, y = q0.5)) +
  geom_ribbon(aes(x=t,ymin=q0.025,ymax=q0.975),alpha=0.2) +
  facet_grid(. ~ name)
```

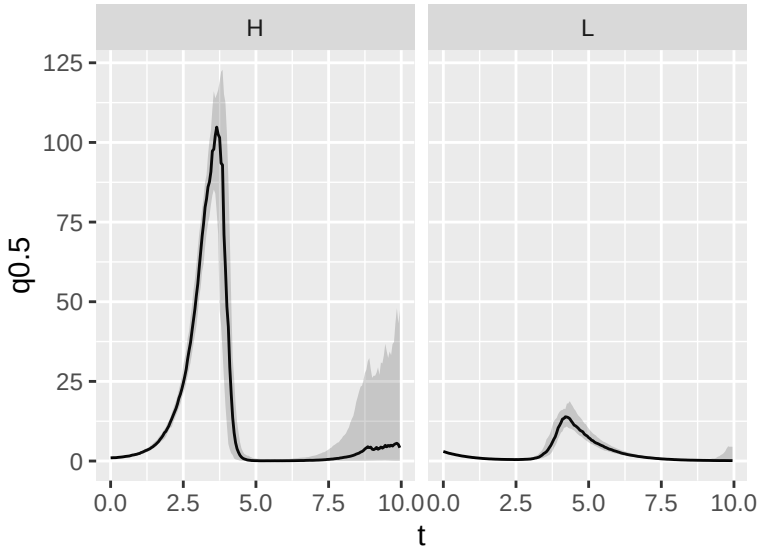


Figure 26.7: Ensemble average plot of the predator-prey system.

The last command `facet_grid(. ~ name)` splits the plot up into different panels (or facets) by the column `name`. So easy to make!

If you are feeling a little overwhelmed by all these new plotting commands - don't worry. You can easily modify these examples for a different systme of differential equations. You've got this!

26.3 Generalizing the approach.

As mentioned previously, another way to think about the logistic differential equation as what as known as a "birth-death process."

Let's call the part of the differential equation that contributes to a positive rate as a "birth process" and parts that contribute to a negative rate as a "death process." If we have the differential equation

$$\frac{dx}{dt} = \alpha(x) - \delta(x)$$

Then we would simulate the birth death process with $\mu = \alpha(x) - \delta(x)$ and $\sigma^2 = \alpha(x) + \delta(x)$.

For a multivariable system of equations the process is the same, however because we have a system of equations the calculations for the expected value and variance require more knowledge of matrix algebra which is beyond the scope here. The provided code does take this into account.

26.4 Exercises

Exercise 26.1. Return back to one simulation of the logistic differential equation. (Figure 26.1). For this plot we set $\sigma = 1$. What happens to the resulting spaghetti and ensemble plots when $\sigma = 0,\ 0.01,\ .1,\ 10$?

Exercise 26.2. Return back to the example of one simulation of the stochastic predator prey (Figure 26.4). For these plots we set $\sigma = .1$. What happens to the resulting spaghetti and ensemble plots when $\sigma = 0,\ 0.01,\ 1,\ 10$?

Exercise 26.3. For the predatory-prey simulation of stochastic variables we used `group_by(name,t)` to begin summarizing our variables. Modify the code so you apply `group_by(t,name)` and generate the ensemble plot. Do you receive a similar result?

Exercise 26.4. For the logistic differential equation consider the following splitting of $\alpha(x)$ and $\delta(x)$:

$$\begin{aligned} \alpha(x) &= rx + \frac{rx^2}{2K} \\ \delta(x) &= \frac{rx^2}{2K} \end{aligned} \tag{26.7}$$

Simulate this SDE using the same values of parameters for the logistic example and compare your results.

Exercise 26.5. (Inspired by Logan and Wolesensky (2009)) Let $R(t)$ denote the rainfall at a location at time t, which is a random process. Assume that probability of the change in rainfall from day t to day $t + \Delta t$ is the following:

change	probability
$\Delta R = \rho$	$\lambda \Delta t$
$\Delta R = 0$	$1 - \lambda \Delta t$

a. With this information, compute $\mathrm{E}[\Delta R]$ and variance of ΔR.
b. Simulate this stochastic process. Use $R(0) = 0$ and run 500 simulations of this stochastic process. Set $\lambda\rho = 18$ and $\sqrt{\lambda\rho^2} = 16$.

Exercise 26.6. Consider the following model for zombie population dynamics (Smith? 2014):

$$\begin{aligned} \frac{dS}{dt} &= -\beta SZ - \delta S \\ \frac{dZ}{dt} &= \beta SZ + \xi R - \alpha SZ \\ \frac{dR}{dt} &= \delta S + \alpha SZ - \xi R \end{aligned} \tag{26.8}$$

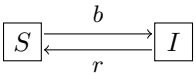
a. Determine the birth and death terms for the zombie model so that you could encode this as a birth/death process:

- Birth part for $\frac{dS}{dt}$:
- Death part for $\frac{dS}{dt}$:
- Birth part for $\frac{dZ}{dt}$:
- Death part for $\frac{dZ}{dt}$:
- Birth part for $\frac{dR}{dt}$:

- Death part for $\dfrac{dR}{dt}$:

b. Use `birth_death_stochastic` to perform 500 simulations of this stochastic differential equation. Please assume the following values of the parameters and stochastic method:

- $\sigma = 0.004$
- $\Delta t = 0.5$.
- Timesteps: 200.
- $\beta = 0.0095$, $\delta = 0.0001$,$\xi = 0.1$, $\alpha = 0.005$.
- Initial condition: $S(0) = 499$, $Z(0) = 1$, $R(0) = 0$.



Exercise 26.7. Consider the stochastic differential equation $dS = (1 - S) + \sigma dW(t)$, where σ controls the amount of stochastic noise. For this stochastic differential equation what is $E[S]$ and $\text{Var}(S)$?

Exercise 26.8. (Inspired by Logan and Wolesensky (2009)) An SIS model is one where susceptibles S become infected I, and then after recovering from an illness, become susceptible again. The schematic representing this is shown in Figure **??**. While you can write this as a system of differential equations, assuming the population size is constant N we have the following differential equation:

$$\frac{dI}{dt} = b(N - I)I - rI \tag{26.9}$$

a. Identify $\alpha(I)$ and $\delta(I)$ for this model.
b. Assuming $N = 1000$, $r = 0.01$, and $b = 0.005$, $I(0) = 1$, simulate this differential equation over two weeks with $\Delta t = 0.1$. Show the plot of your result.

Exercise 26.9. Consider the equation

$$\Delta x = \alpha(x)\ \Delta t - \delta(x)\ \Delta t$$

If we consider Δx to be a random variable, show that the expected value μ equals $\alpha(x)\ \Delta t - \delta(x)\ \Delta t$ and the variance σ^2, to first order, equals $\alpha(x)\ \Delta t + \delta(x)\ \Delta t$.

Chapter 27

Solving Stochastic Differential Equations

Stochastic differential equations arise when we consider variation (think randomness) in a biological model. Through simulation and examining the ensemble average we found that the solution to a stochastic differential equation is an *distribution* of solutions. For this final section we will examine how we can characterize solutions to stochastic differential equations. We will introduce methods to develop exact solutions to a stochastic differential equation. You will learn about some powerful mathematics that hopefully you will want to study at some point in the future!

27.1 Meet the Fokker-Planck Equation

Let's start with a general way to express a stochastic differential equation:

$$dx = a(x,t)\,dt + b(x,t)\,dW(t) \tag{27.1}$$

The "solution" to this SDE will be a probability density function $f(x,t)$. Our goal is to have a function that describes the evolution of $f(x,t)$ in both time and space. Based on our work with birth-death processes, the probability density function $f(x,t)$ should have the following properties:

- $E[f(x,t)]$ is the deterministic solution to $\dfrac{dx}{dt} = a(x,t)$.
- $\text{Var}[f(x,t)]$ is proportional to $b(x,t)$.

We determine the probability density function through solution of the following differential equation, which is called the **Fokker-Planck Equation**:

$$\frac{\partial f}{\partial t} = -\frac{\partial}{\partial x}\left(f(x,t)\cdot a(x,t)\right) + \frac{1}{2}\frac{\partial^2}{\partial x^2}\left(\, f(x,t)\cdot (b(x,t))^2 \,\right) \tag{27.2}$$

We can write this equation in shorthand, dropping the dependence of x and t for $f(x,t)$, $a(x,t)$ and $b(x,t)$: $f_t = -(f\cdot a)_x + \frac{1}{2}(f\cdot b^2)_{xx}$.

Example 27.1. Consider the SDE $dx = dW(t)$ and apply the Fokker-Planck equation to characterize the solution $f(x,t)$.

Solution. In this case $a(x,t) = 0$ and $b(x,t) = 1$, so the Fokker-Planck Equation is:

$$f_t = \frac{\sigma^2}{2} f_{xx}.$$

This equation should look familiar - it is the partial differential equation for diffusion![1] The solution to this SDE is given by Equation (27.3).

$$f(x,t) = \frac{1}{\sqrt{2\pi\sigma^2 t}} e^{-x^2/(2\sigma^2 t)} \tag{27.3}$$

[1] To remind you, the solution to $p_t = Dp_{xx}$ is $p(x,t) = \frac{1}{\sqrt{4\pi Dt}} e^{-x^2/(4Dt)}$. So in this case $D = \sigma^2/2$.

I didn't mention the initial condition for Equation (27.3). For this SDE the initial condition is a special function called the Dirac delta Function. We denote the Diract delta function as $\delta(x)$. This function is a special type of probability density function, which you may study in a courses such as Functional Analysis (or Analysis). We can plot the evolution of this solution in Figure 27.1:

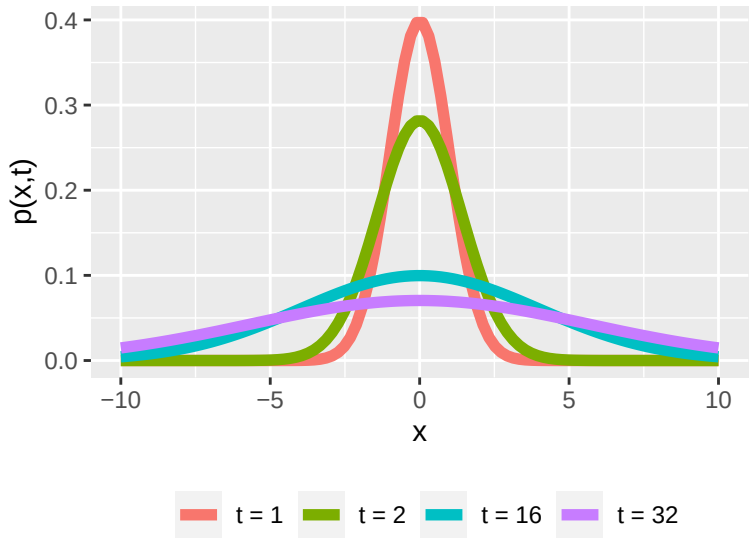


Figure 27.1: The solution to SDE $dx = dW(t)$.

Now that we have a handle on the SDE $dx = dW(t)$, let's extend this next example a little more.

27.1.1 Another Fokker-Planck Equation

Consider the SDE $dx = r\ dt + \sigma\ dW(t)$, where r and σ are constants. As a first step, let's take a look at the deterministic equation: $dx = r\ dt$. This is the differential equation $\dfrac{dx}{dt} = r$, which has a linear function $x(t) = rt + x_0$ as its solution.

We will apply the Fokker-Planck equation to characterize the solution $f(x,t)$. In this case, the Fokker-Planck equation is

$$f_t = -rf_x + \frac{\sigma^2}{2} f_{xx} \tag{27.4}$$

Equation (27.4) is an example of a *diffusion-advection equation*. Amazingly this equation can be reduced to a diffusion equation through a change of variables and application of the multivariable change of variables. Let's get to work to figure this out.

First, let $z = x - rt$ and $\tau = t$. This change of variables may seem odd, but our goal here is to write $f(x,t) = f(z,\tau)$ and to develop expressions for f_t f_x, and f_{xx} with this change of variables. But in order to do that, we will need to apply the *multivariate chain rule* (see Figure 27.2).

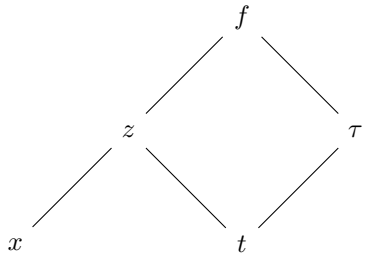


Figure 27.2: Multivariable chain rule

By the multivariable chain rule we can develop expressions for f_t and f_x:

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial \tau} \cdot \frac{\partial \tau}{\partial t} + \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial t}$$
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial x}$$

Now let's consider the partial derivatives $\frac{\partial \tau}{\partial t}$, $\frac{\partial z}{\partial t}$, and $\frac{\partial z}{\partial x}$.

Remember that $z = x - r\tau$. By direct differentiation $z_\tau = -r$ and $z_x = 1$. Also since $\tau = t$ then $\tau_t = 1$. With these substitutions, we can now re-write f_t and f_{xx}:

$$\frac{\partial f}{\partial t} = \frac{\partial f}{\partial \tau} \cdot \frac{\partial \tau}{\partial t} + \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial t} = \frac{\partial f}{\partial \tau} - r\frac{\partial f}{\partial z}$$
$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial z} \rightarrow \frac{\partial^2 f}{\partial x^2} = \frac{\partial^2 f}{\partial z^2}$$

So if we re-write our original Fokker-Planck equation with the variables z and τ we have:

$$f_t = -rf_x + \frac{\sigma^2}{2} f_{xx}$$
$$f_\tau - rf_z = -rf_z + \frac{\sigma^2}{2} f_{zz}$$
$$f_\tau = \frac{\sigma^2}{2} f_{zz}$$

Ok: I'll admit that doing change of variables may seem like it doesn't help the situation. But guess what: with this change of variables our Fokker-Planck equation becomes a diffusion equation in the variables z and τ! So if we can write down the solution with the variables z and τ, we can write the solution $f(x,t)$! Here how we do this: $f(z,\tau) = \frac{1}{\sqrt{2\pi\sigma^2\tau}} e^{-z^2/(2\sigma^2\tau)}$, and then transform back into the original variables x and t:

$$f(x,t) = \frac{1}{\sqrt{2\pi\sigma^2 t}} e^{-(x-rt)^2/(2\sigma^2 t)} \tag{27.5}$$

Now that we have an equation, next let's visualize the solution. Let's take a look at some representative plots in Figure 27.3:

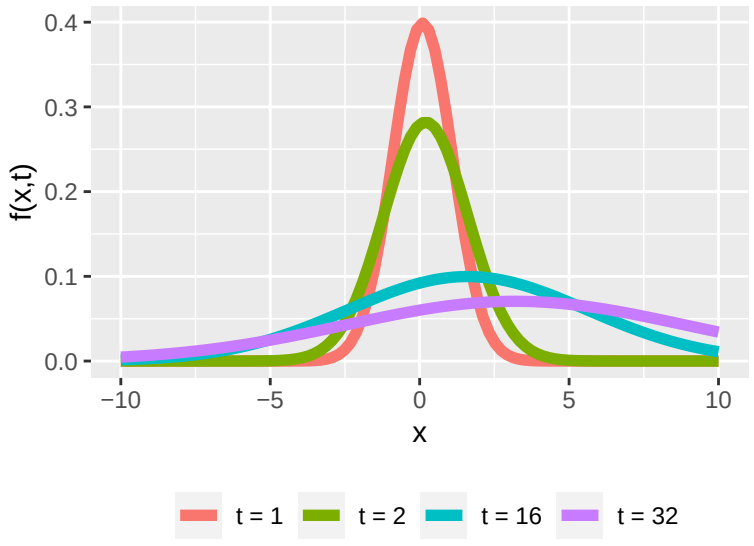


Figure 27.3: Representative plots for the solution to the SDE $dx = r\ dt + \sigma\ dW(t)$.

Based on what we know of this distribution is that it should look like a normal distribution with mean $\mu = rt$ and variance $\sigma^2 t$. What the mean and variance tells us that the mean is shifting and growing more diffuse as time increases. Remember that our solution to the deterministic equation was linear, and the mean of our distribution grows linearly as well!

Also notice in Figure 27.3 as t increases the solution shifts ("advects") to the right. The differential equation is an example of a *diffusion-advection* equation, or a solution that drifts as time increases.

The examples we study here are just a few examples of how to build a deeper understanding of stochastic processes and differential equations. There is a lot of power in understanding the theoretical distribution for some test cases. Stochastic differential equations is a fascinating field of study with a lot of interesting mathematics - I hope what you learned here will make you want to study it further!

27.2 Exercises

Exercise 27.1. (Inspired by Logan and Wolesensky (2009)) Let $R(t)$ denote the rainfall at a location at time t, which is a random process. Assume that probability of the change in rainfall from day t to day $t + \Delta t$ is the following:

change	probability
$\Delta R = \rho$	$\lambda \Delta t$
$\Delta R = 0$	$1 - \lambda \Delta t$

The stochastic differential equation generated by this process is $dR = \lambda\rho\, dt + \sqrt{\lambda\rho^2}\, dW(t)$.

a. What is the Fokker-Planck partial differential equation for the probability distribution $f(R, t)$?
b. What is a formula that solves the Fokker-Planck partial differential equation?
c. Make some representative plots of the solution as it evolves over time

Exercise 27.2. A particle is moving in a gravitational field but still allowed to diffuse randomly. In this case the stochastic differential equation is $dx = -g\, dt + \sqrt{D}\, dW(t)$.

a. What is the Fokker-Planck partial differential equation for the probability distribution $f(x, t)$?
b. Based on the work done in this section, what is the equation for the probability distribution $f(x, t)$?

Exercise 27.3. Consider the stochastic differential equation $dS = (1 - S) + \sigma dW(t)$, where σ controls the amount of stochastic noise.

a. First let $\sigma = 0$ so the equation is entirely deterministic. Classify the stability of the equilibrium solutions for this differential equation.
b. Still let $\sigma = 0$. Apply separation of variables to solve this differential equation.
c. Now let $\sigma = 0.1$. Do 100 realizations of this stochastic process, with initial condition $S(0) = 0.5$. What do you notice?
d. Now try different values of σ larger and smaller than 0.1. What do you notice?
e. What is the Fokker-Planck partial differential equation for the probability distribution $f(S, t)$?

Exercise 27.4. (Inspired by Logan and Wolesensky (2009)) Consider the differential equation $x' = \lambda x - c\mu x^2$, which is similar to a logistic differential equation. The *per capita* rate equation for this differential equation is $\dfrac{x'}{x} = \lambda - c\mu x$.

a. Assume there is noise to this per capita rate, i.e. $\dfrac{x'}{x} \to \dfrac{x'}{x} +$ Noise. With this revised equation, what are the deterministic and stochastic parts?
b. What is the Fokker-Planck partial differential equation for the probability distribution $f(x, t)$?
c. The *steady-state distribution* assumes that $f_t = 0$, so $f(x, t) \to f(x)$. Through direct verification, show that $f(x) = Dx^{\alpha-1}e^{-\beta x}$ is a solution to the steady-state distribution, where $\alpha = 2\lambda - 1$ and $\beta = 2c\mu$.
d. With $\lambda = c = \mu = D = 1$, make a graph of $f(x)$ and graph it below:

Exercise 27.5. (Inspired by Gardiner (2004)) A type of chemical reaction is $X + A \leftrightarrow 2X$, where A acts like an enzyme. The stochastic differential equation that describes this scenario is:

$$dX = \left(AX - X^2\right)\, dt + \left(AX + X^2\right) dW(t) \tag{27.6}$$

a. What is the Fokker-Planck equation for this stochastic differential equation?
b. The steady-state distribution for this process is $p(X) = e^{-2X}(A + X)^{4A-1}X^{-1}$. With $A = 1$ make a plot of this distribution.

Exercise 27.6. Models of cell membranes take account for the energy needed for ions and other materials to cross the cell membrane, usually expressed as a membrane potential $U(x)$, where x is the current position of a particle distance. The probability p of the particle being at position x at time t is given by the Fokker-Planck equation:

$$v\frac{\partial p}{\partial t} = \frac{\partial}{\partial x}\left(U'(x)p\right) + kT\frac{\partial^2 p}{\partial x^2}, \tag{27.7}$$

where k is Boltzmann's constant and T is the temperature.

a. Write the Fokker-Planck equation in steady state.
b. Show that a solution for the steady-state equation is $p(x) = Ce^{-\frac{U(x)}{kT}}$
c. If the steady-state distribution is normal, what is function will $U(x)$ be?

References

Akaike, Hirotugu. 1974. "A New Look at the Statistical Model Identification." *IEEE Transactions on Automatic Control* 19 (6): 716–23. https://doi.org/10.1109/TAC.1974.1100705.

Berg, Howard. 1993. *Random Walks in Biology*. Princeton University Press.

Berg, Hugo van den. 2011. *Mathematical Models of Biological Systems*. Illustrated edition. Oxford ; New York: Oxford University Press.

Beven, Keith, and Jim Freer. 2001. "Equifinality, Data Assimilation, and Uncertainty Estimation in Mechanistic Modelling of Complex Environmental Systems Using the GLUE Methodology." *Journal of Hydrology* 249 (1-4): 11–29. https://doi.org/10.1016/S0022-1694(01)00421-8.

Burnham, Kenneth P., and David R. Anderson, eds. 2002. *Model Selection and Multimodel Inference*. New York, NY: Springer New York.

Carroll, Cameron Jewett. 2013. "Modeling Winter Severity and Harvest of Moose: Impacts of Nutrition and Predation." PhD thesis, Fairbanks, Alaska: University of Alaska Fairbanks.

Devore, Jay L., Kenneth N. Berk, and Matthew A. Carlton. 2021. *Modern Mathematical Statistics with Applications*. Third. Springer Texts in Statistics. Springer International Publishing. https://doi.org/10.1007/978-3-030-55156-8.

Gardiner, C W. 2004. *Handbook of Stochastic Methods for Physics, Chemistry, and the Natural Sciences*. 3rd ed. springer.

Gause, G. F. 1932. "Experimental Studies on the Struggle for Existence: I. Mixed Population of Two Species of Yeast." *Journal of Experimental Biology* 9 (4): 389–402.

Keener, James, James Sneyd, S. S. Antman, J. E. Marsden, and L. Sirovich, eds. 2009. *Mathematical Physiology*. Vol. 8/1. Interdisciplinary Applied Mathematics. New York, NY: Springer New York. https://doi.org/10.1007/978-0-387-75847-3.

Logan, J. David, and William Wolesensky. 2009. *Mathematical Methods in Biology*. 1st ed. Hoboken, N.J: Wiley.

Richey, Matthew. 2010. "The Evolution of Markov Chain Monte Carlo Methods." *The American Mathematical Monthly* 117 (5): 383–413.

Scholz, Gudrun, and Fritz Scholz. 2014. "First-Order Differential Equations in Chemistry." *Chemtexts* 1 (1): 1. https://doi.org/10.1007/s40828-014-0001-x.

Schwartz, G. 1978. "Estimating the Dimensions of a Model." *Annals of Statistics* 6 (2): 461–64.

Sinay, Laura, and Leon Sinay. 2006. "A Simple Mathematical Model for the Effects of the Growth of Tourism on Environment."

Smith?, Robert J., ed. 2014. *Mathematical Modelling of Zombies*. Ottawa: University of Ottawa Press.

Stenseth, Nils Chr, Wilhelm Falck, Ottar N. Bjørnstad, and Charles J. Krebs. 1997. "Population Regulation in Snowshoe Hare and Canadian Lynx: Asymmetric Food Web Configurations Between Hare and Lynx." *Proceedings of the National Academy of Sciences* 94 (10): 5147–52. https://doi.org/10.1073/pnas.94.10.5147.

Thornley, John H M, and Ian R Johnson. 1990. *Plant and Crop Modelling: A Mathematical Approach to Plant and Crop Physiology*. Oxford Science Publications.