Learning Modules in Evolutionary Ecology

Ron Bassar

2024-03-20

Contents

1 Introduction **2**

1.1 General Approach and Data 2

1.2 How to do science 5

2 Introduction to Data Analysis **8**

2.1 Approach 8

Data Organization 8

Work Flow 8

Reproducible Analysis 9

Goals for this section 9

2.2 PART I 10

2.3 PART II 11

1. Working directly with R: Click on the R icon 11

2. Working in RStudio: Click on the RStudio icon 11

3. Preliminary work in the Console Window in RStudio 11

4. Your first script 15

2.4 PART III. Linear regression and linear models 18

Background to simple linear regression 18

Conducting a simple linear regression 23

Tests of the statistical hypotheses 25

2.5 PART IV. Testing the hypothesis that predation selects for higher growth rates 27

Part IV Exercises: 32

3 Population Growth, Mean Fitness, and Changing Fitness Measures **33**

3.1 Introduction 33

3.2 PART I: Exponential and Logistic Population Growth in Continuous and Discrete Time 35

Exponential Growth in Continuous Time 35

Logistic Population Growth in Continuous Time 38

Geometric Population Growth 40

Part I: Exercises 42

3.3 Part II. Using real data 43

Part II: Exercises 47

3.4 Part III. Population growth as mean individual fitness 48

Part III Exercises 50

4 Age-structure and Fitness **53**

4.1 Introduction 53

4.2 Part I. Age-structured Population Growth 54

Per-generation Rate of Increase (R_0) 54

Per time-step population growth rate, λ 56

Geometric rate of increase, λ: brute force method 2 61

Geometric rate of increase, λ: Analytical method 63

4.3 Part II. Beyond Population Growth: Stable Age Distributions and Reproductive Value 65

Stable Age Distribution and Reproductive Value From the Life Table 66

Stable Age Distribution and Reproductive Value From the Matrix Model . . 69

Part II Exercises 72

5 The Analysis of Marked Individuals **73**

5.1 Introduction 73

5.2 Part I: A simple mark-recapture analysis–the Cormack-Jolly-Seber (CJS) model for survival and probability of capture 74

Part I Exercises 80

5.3 Part II: POPAN analysis for survival, probability of capture, recruitment, and population size 80

Part II Exercises 85

6 Pedigrees, Breeding Values, and Heritability **87**

6.1 Introduction 88

6.2 Part I: The Pedigree 89

6.3 Part II. The Animal Model 91

6.4 Part III. Calculating additive genetic variance (i.e. variance in breeding values) and heritability 93

Part III Exercises 98

7 Natural Selection on SLAM **100**

7.1 Introduction 100

7.2 Part I. Natural Selection on SLAM Using the Breeder's Equation 101

7.3 Part II. Change in the Phenotypic and Breeding Values 106

Part II Exercises 110

7.4 Part III. The Robertson-Price Identity 110

Part III Exercises 114

8 Species Coexistence **115**

8.1 Part I: Visualizing Competition and Coexistence 115

Species coexistence in discrete time 115

Creating a function to explore different scenarios 118

Part I Exercises 122

8.2 Part II: Invasion Analysis 122

Part II Exercises 126

8.3 Part III: When Invasion Analysis Fails 127

Part III Exercises 129

8.4 Part IV: Empirical Tests 129

Part IV Exercises 134

Chapter 1

Introduction

The lectures and discussions will expose you to ecological and evolutionary theory and empirical support for these theories. The overall purpose of these learning modules are to reinforce these concepts by giving you hands on experience with methods used by researchers. For the experimental components, we will make use of existing data from experimental and natural populations. Thus the focus of the modules is less about skills in data collection more about designing experiments that test hypotheses and how to analyze data that has been collected. For the theoretical modules, the purpose is to go from a question, to potential answers (hypotheses), to a mathematical expression that are used to derive predictions that can then be tested with experiments.

Both of these goals involve the use of the computer. However, this is not a computer science course, so we will learn to use the computer as a tool, but nothing more.

1.1 General Approach and Data

The data we will use for the empirical work comes from a species that has been used extensively to address questions in evolutionary ecology, the Trinidadian guppy (*Poecilia reticulata*). Guppies inhabit streams and rivers throughout the Caribbean island of Trinidad and occur along gradients of predation intensity. These gradients occur along the southern and northern slopes of the Northern Range Mountains. The Northern Range Mountains of Trinidad offers a natural laboratory for studying interactions between ecology and evolution. The rivers draining these mountains flow over steep gradients punctuated by waterfalls that create distinct fish communities above and below waterfall barriers. Species diversity decreases upstream as waterfalls block the upstream dispersal of some fish species. The succession of communities is repeated in many, parallel drainages, providing researchers with natural replicates of the evolutionary process.

These streams also offer the opportunity of performing experimental studies of evolution because rivers can be treated like giant test tubes, as fish can be introduced into portions of stream bracketed by waterfalls to create *in situ* experiments (Endler 1978; 1980). Downstream guppies co-occur with a diversity of predators, which prey on the adult fish (high

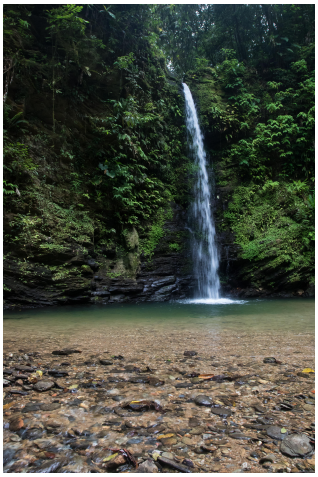


Figure 1.1: Barrier waterfall. Credit: Joshua Goldberg

predation, or HP). Waterfalls often exclude predators but not guppies, so when guppies are found above waterfalls they have greatly reduced predation risk and increased life expectancy (low predation, or LP). Hart's killifish, *Rivulus hartii*, the only other fish found in many of these localities, rarely preys on guppies and tends to focus on the small, immature size classes (Haskins et al. 1961; Endler 1978). In some headwater streams, the killifish is the only fish species present because they are capable of overland travel on rainy nights (killifish only, or KO localities). Population genetic analyses reveal that at least some of these rivers represent independent replicates of the evolution of guppies adapted to HP and LP environments (Alexander, et al. 2006) and that LP and HP populations are more genetically distinct than expected under migration-drift equilibrium (Barson et al. 2009).

Figure 1.2: A predator of guppies, *Crenicichla alta*. Credit: Unknown

Guppies adapted to HP environments mature at an earlier age, devote more resources to reproduction, produce more offspring per brood and produce significantly smaller offspring than LP guppies (Reznick and Endler 1982; Reznick et al. 1996). All of these differences are consistent across replicate HP-LP comparisons in multiple watersheds, and are also consistent with predictions derived from theory that models how life histories should evolve in response to selective predation on juveniles (LP environments) versus adults (HP environments) (Gadgil and Bossert 1970; Law 1979; Michod 1979; Charlesworth 1994). HP and LP guppies also differ in male coloration (Endler 1978), courtship behaviour (Houde 1997), schooling behaviour (Seghers 1974; Seghers and Magurran 1995), morphology (Langerhans and DeWitt 2004), swimming performance (Ghalambor et al. 2004), and diet (Zandona et

al. 2011). Male coloration evolves in response to the combined, conflicting influences of natural and sexual selection (Endler 1978; Endler 1980). Laboratory studies confirm that the differences in life histories, coloration, behaviour and body shape have a genetic basis (Endler 1980; Reznick 1982; Reznick and Bryga 1996; O'Steen et al. 2002). Genetic diversity is consistently greater in the higher-order streams than in the headwaters. This pattern, combined with the observation that guppies periodically invade or are extirpated from headwaters implies a dynamic process of invasion of and adaptation to low predation environments.

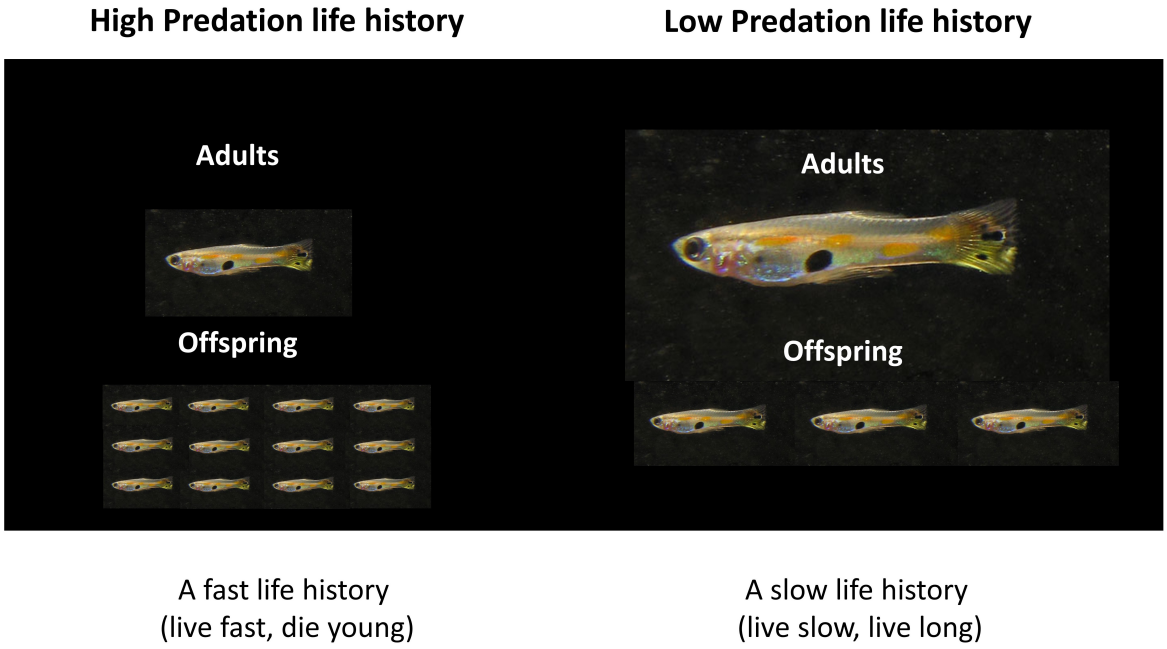


Figure 1.3: Life history differences between HP and LP guppies. Credit: Unknown

One of the really neat aspects of guppies that sets them apart from other organisms is that we can study them in natural habitats using comparative analyses, in artificial streams (mesocosms), and in the laboratory. They also have relative short generation times (~5 months). There are few other organisms that can be studied across these three levels of realism.

Moreover, because guppies occur in most but not all habitats on the island, we can exploit the presence of barrier waterfalls to perform experimental studies of evolution in natural populations. We can manipulate the mortality risks of guppies by transplanting them from high predation localities below barrier waterfalls, into previously guppy-free portions of streams above barrier waterfalls. In this way, we can simulate a natural invasion. Introduced guppies evolve delayed maturation and reduced reproductive allocation, as seen in natural LP communities. In previous experimental introductions, male traits evolved in four years or less. Our inferences that guppies had evolved were derived from laboratory experiments performed on the laboratory-reared grandchildren of wild-caught parents collected from the

introduction sites and the ancestral HP sites. Other attributes of guppies, including behavior, also evolved rapidly. These results argue that the presence or absence of predators imposes intense selection on life histories and other features of guppy phenotypes.

Mark-recapture studies on natural populations support the role of predators in shaping guppy evolution, but at the same time suggest that resource availability is important. HP guppies experience substantially higher mortality rates than LP guppies, which suggests that predator-induced mortality is a candidate cause for the evolution of the HP phenotype. However, guppy populations in low predation sites tend to have higher population densities, slower individual growth rates and size distributions shifted towards larger fish. These differences in population structure are attributable in part to demography: HP guppies have higher birth and death rates. They are also attributable to evolved differences in life histories: LP guppies mature at a later age and have lower birth rates. Thus, removal of predators causes not only changes in the age-specific probabilities of death, but also leads to increased population sizes and lower resource availability in LP locations.

We will be working directly with much of the data used to test various hypotheses about population dynamics, evolution in guppy populations, and coexistence with killifish.

1.2 How to do science

Science always starts with a question. For example, "How does predation influence how fast organisms grow?" The next step in the process is to develop a *Biological Hypothesis*. A biological hypothesis is a possible answer to the question, but ought not to be a simple answer such as "It should make it increase". There needs to be an explanation for why. For example, one hypothesis in life history theory suggests that organisms that live with predators should grow faster than organisms without predators because larger size can reduce the risk of predation. This hypothesis makes the prediction that if we can measure the growth of closely related organisms that live with and without predators, then the ones that live with predators should have higher growth rates. Yet, this may not be the only hypothesis that makes that same prediction. For example, another hypothesis may say that if fecundity is positively related to size, then species that are subject to high rates of predation grow quickly because it allows them have more offspring before they are killed by predators. Both hypotheses make the same prediction and so it is often important to develop further predictions that can discriminate between the two.

Once you have your biological hypotheses and predictions, then you can design an experiment that can test the prediction. In the next section, we will do this using growth data on guppies from high and low predation populations in artificial streams over a 28-day period. There are 16 artificial streams that represent a common environment–everything is the same between the artificial streams besides the fish that are in them. There are two factors in the experiment, each with two levels. The first factor is the predation regime of origin (LP or HP). The second factor is the number of fish in the artificial stream. This also has two levels (12 or 24). Each combination of predation regime and density is equally represented in the data. This type of experiment is called a 2 x 2 factorial design and is a workhorse in

experimental biology. You decide that you will measure all the fish before they go into the experiment and measure them again after 28 days to measure how much they grew. How do you go about objectively saying whether you accept or reject your biological hypothesis?

Figure 1.4: Artifical streams (mesocosms) in Trinidad.

The objective method for deciding whether you accept or reject your hypothesis is the realm of statistics. There are numerous types of statistics out there. Yet, in almost every case, a major goal of statistics is to provide methods that allow us to accept or reject hypotheses. Yet the relationship between testing a hypothesis in statistics and in biology are not equivalent. The distinction is often glossed over and so it is worth spending a moment on this.

Hypotheses in statistics come in two flavors. The first is the null hypothesis, often denoted H_0. The second flavor is the alternative hypothesis, H_a. The null hypothesis in statistics is the hypothesis of no effect. For example, there is not an association between predation and growth. The alternative hypothesis is that there is an association between predation and growth. The subtlety is that the biological hypothesis is not that there should be an association between predation and growth, rather this is the prediction from the biological hypothesis. So statistics test predictions of biological hypotheses. The other distinction is that the goal of statistics is not to test the prediction *per se*, but the focus is on rejecting the null hypotheses, the one of no effect. This may seem a bit odd, but the consequence of this is that we never can say that the alternative hypothesis is true, only that the null is false or rejected. This, in turn, has important consequences for our conclusions from the analyses. For example, if we reject the null hypothesis of no effect, then we can say the results are consistent with our biological hypothesis. However, we cannot say our hypothesis is true. Because, remember, there may be other hypotheses that yield similar predictions. Truth, then in science, comes from eliminating alternative biological hypotheses. Once all others have been eliminated, then we arrive at the truth.

Throughout this course, we will be using several types of statistical tests, but none more than linear regression and its variations. Below we will go through what linear regression is all about. We will then test the predictions from the growth/predation hypothesis using the experimental design above.

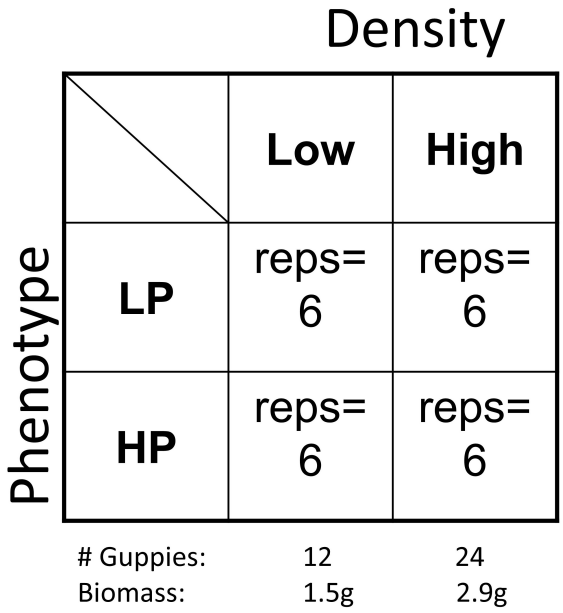


Figure 1.5: A 2 x 2 factorial design with LP and HP guppies at two densities in mesocosms. Values inside the cells represent the number of articial streams (mesocosms) with that treatment combination. Credit: Unknown

Chapter 2

Introduction to Data Analysis

2.1 Approach

An important part of Evolutionary and Ecology is to seek predictable patterns and to understand their underlying causes. Much of this process involves quantitative work: collecting, analyzing and presenting data in a clear and organized way, with the goal to discover fundamental insights into how nature works. This laboratory focuses on the mechanics of this process: how to chose a statistical method, organize data, to follow a workflow, and to present findings that are evidence-based, will have impact, and will also be reproducible and verifiable.

This introductory section stresses three main points about the general approach to data analysis: data need to be organized; the analysis should follow a well-structured work flow; and the overall process needs to be reproducible. To accomplish this, we will be using program R and R Studio.

Data Organization

The key point here is that data and analyses should be organized so that you can find your way to them at a later time. It is best to lay out a set of folders in a logical way, and to think about how you will do this before you start. One well-tested system is to make a set of folders or directories with a consistent and hierarchical structure. For the laboratory in this course, it would make sense to have a separate folder for each laboratory, then within that folder have three standard folders: a **Data Folder** that contains your Excel workbooks and .csv files; a **Results Folder** with your R Scripts, and a **Documents Folder** with your written reports or results, for example your worksheets or lab report.

Work Flow

Data analysis tends to be easier and less fraught with error if you develop a consistent work flow. In general, it is best think of the work flow to have three parts. For empirical work,

visualize your data. Before you do any analyses, look at your data by making figures. This will allow you detect errors, but most important to think about patterns and to gain some intuitive ideas about what is "going on" in your data. Second fit and examine a statistical model. The main point here is to find a model to test your hypothesis or to support one or more ideas, but especially to evaluate carefully whether the data meet the assumptions of model. In most cases in statistics there is a series of well-established diagnostic check-offs to help you to find your way through this process. Third, interpret the model; this means to evaluate the salient patterns that the analysis reveals in the data and, most important, to think about the biological significance of the result.

Reproducible Analysis

The third point is that your work must be reproducible. Standards now demand that you be able to document how you obtained your findings. The key point about reproducibility is that you make an electronic record of every step. Start with an archival copy of your data - a "good" copy (for example in a .csv file) that you do not edit. Do all of your analysis using a script. Make your script permanent, repeatable, annotated, shareable, and archived.

The idea is that it is critical that others can understand and repeat the analyses. But you will find this approach extremely important if you find an error in your data (you can simply rerun the script with the correction), or if you need to do a similar analysis later.

A good habit is to react with alarm when you start to manipulate data by hand (for example, in an Excel Workbook); every time you do that you generate an opportunity for cryptic error, or at best a very tedious time if you need to repeat the work. Aim to move your analyses to the script stage as quickly as possible.

Goals for this section

The goal of this section is to get organized for analyzing and interpreting data in Evolutionary and Ecology. First, we do this by getting R and RStudio set up, by doing some preliminary work with RStudio or R, and then by working on the worksheet itself.

This module therefore has three parts:

Part 1. Set things up: install R and RStudio on your computer and organize your directories for the course. For the lab, we will be working in one of the computer labs. This section is for you to set up R and RStudio on your personal computer.

Part 2. Work with R (or RStudio). Follow the instructions on this handout to execute a few commands on R using the Console Window of R or RStudio: execute some simple expressions, call some functions, and to use the Data Window of RStudio to become familiar with a few of the important data structures in R.

Part 3. Learn basic linear models. In this section, I will introduce you to the basics of a linear model, which is the foundation for much of the analyses we will be doing latter on.

It is important to get these introductory skill down as we will build on them considerably throughout the labs. Please ask questions if something does not make sense.

2.2 PART I

Set things up: install R and RStudio and organize your work directories. You should do this before class and come see me if you have trouble. There is also lots of help online. Just type your question and follow it with "in R".

1. Install R on your computer.

Navigate to "cran.r-project.org" and follow the instructions for downloading for Mac, Windows, or Linux (Figure 1).

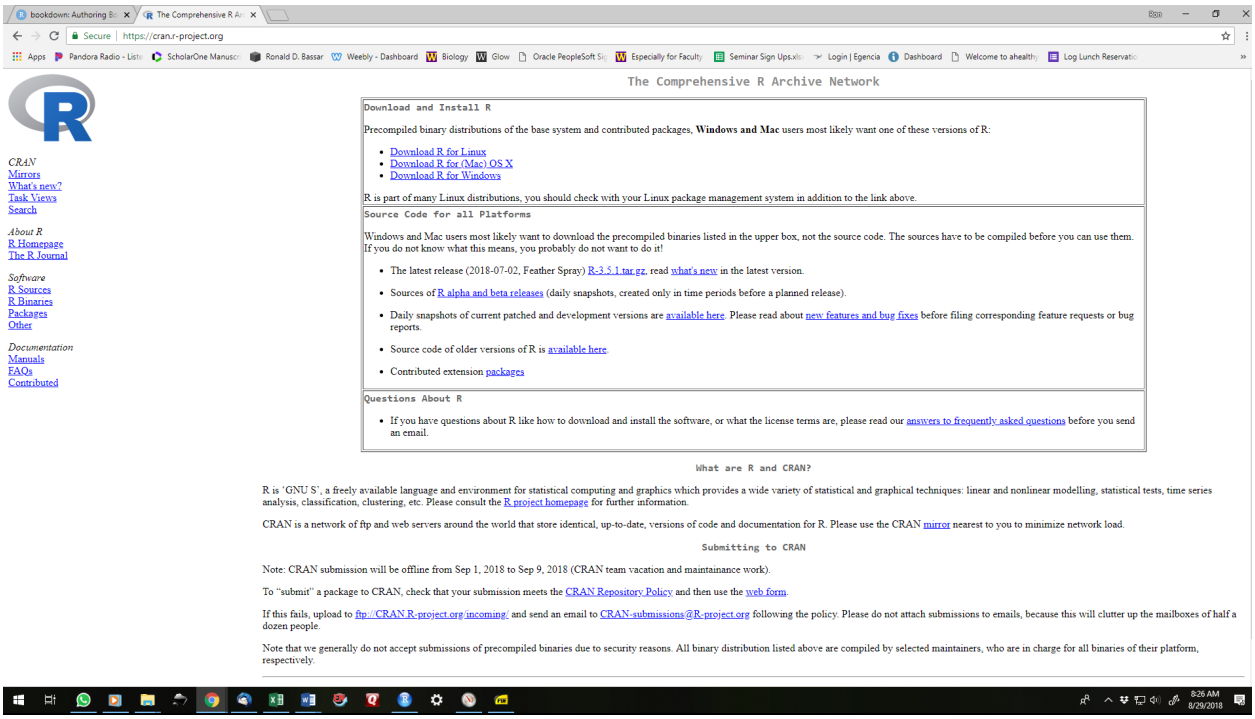


Figure 2.1: The cran R web page where you can find the links to download the latest version of R.

2. Install RStudio

Navigate to "www.rstudio.com/products/rstudio/download/". You will be given the option to download several versions of RStudio. You want the free one.

2.3 PART II

Working with RStudio or R and some examples of R commands and data structures.

RStudio is a "wrapper" function that runs R: it organizes your workspace on the screen so that it is easy to see your work - the comments below assume you are working with RStudio. If you prefer to work directly with R, start with step 1 and skip step 2; to work with RStudio, start with step 2. Either way you can quickly set up a Console Window and a text file or window for your script. RStudio also has two additional windows that can be useful.

1. Working directly with R: Click on the R icon

You will see the Console Window. Use File/New Document to bring up a text tile where you can write, run and save your R scripts. Plot windows will appear when you make plots (you can copy these and paste them into Word documents), and you can view objects (including data) by entering them by name into the Console Window.

2. Working in RStudio: Click on the RStudio icon

You will see the RStudio Screen with 3 windows: Console window (left), Script Window (upper left), Data Window (upper right), and a window (lower right) with a series of tabs: most important for this worksheet are the Help and Plot Tabs}

3. Preliminary work in the Console Window in RStudio

The console window is where you can type in and execute commands (see Figure 2), and where the results of command executions are displayed. This is the best place to try out some R commands to see what they do. But - and this is very important! - once you start to do analyses, always use the Script Window, where you can develop your analyses and build a permanent documented record of your work. More on this later. Although we start with the Console Window, you should aim to do all of your work using the script window. The logic is that your aim is to minimize the work that you do by hand and which therefore has no permanent record.

You can type R commands into the Console Window; hit Return and R will execute the command and return the result.

An excellent tutorial site for basic R is at: http://tryr.codeschool.com; it is worth visiting later.

For this module, the section below covers some of the kinds of expressions, functions and data structures you will use in the worksheets.

Let's start by using the console in RStudio like a calculator.

Start by typing the following into the console and then hit the "Enter" key:

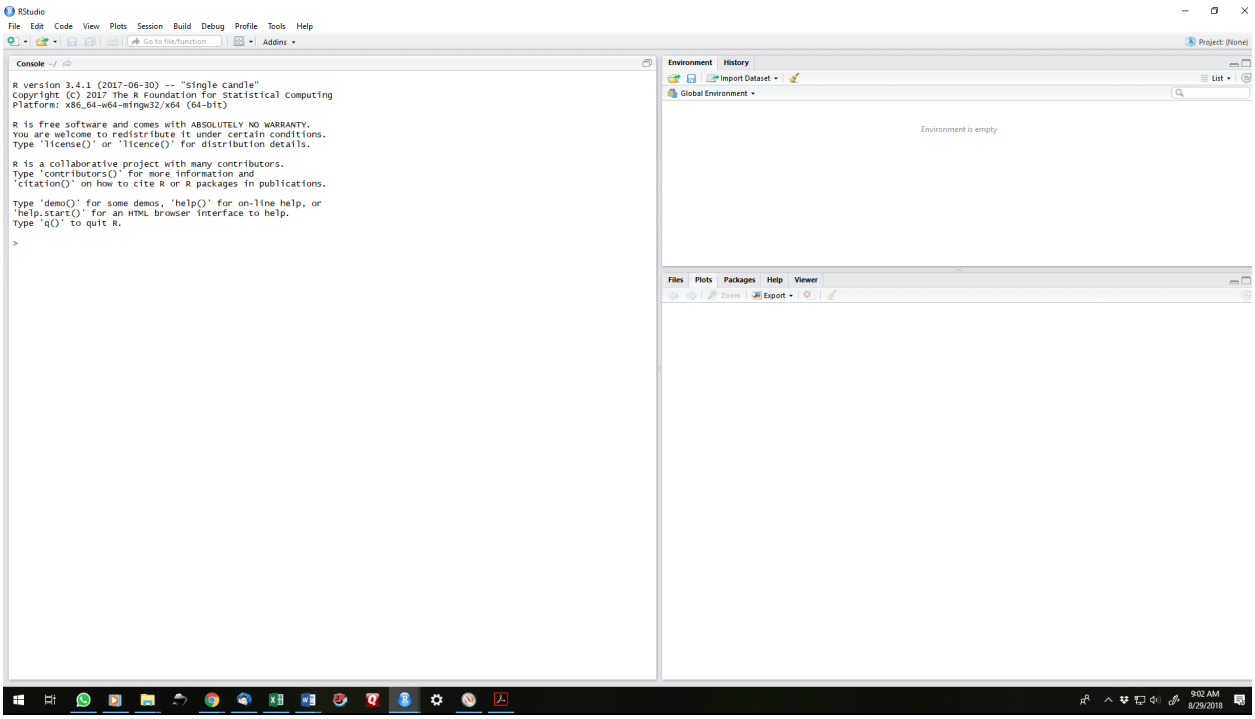


Figure 2.2: The RStudio application. The left panel is called the console.

```
1 + 1
```

```
## [1] 2
```

First, anything in gray in this tutorial represents what you type in. The stuff after the double hashs (##) is what RStudio should return. Note that in RStudio the ##'s are not there.

What happens if RStudio does not return the answer, but instead a plus appears? For example, type in "1 +". This situation, RStudio returns a "+". This means that the line of code you are writing is incomplete. Simply type another "1" on this new line. RStudio returns the answer.

Let's try some others:

```
#Multiplication
2 * 4
```

```
## [1] 8
```

```
#Division
3/8
```

```
## [1] 0.375
```

```
#Subtraction
10.5 - 3.6
```

```
## [1] 6.9
```

```
#Powers
10^3
```

```
## [1] 1000
```

```
#Logarithm (Natural)
log(10)
```

```
## [1] 2.302585
```

```
#Logarithm (base 10)
log10(10)
```

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

```
#Trigonometric
sin(pi)
```

```
## [1] 1.224606e-16
```

```
#Logical
3<10
```

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

Most of this is pretty straightforward, but there are a few things to note. First, if you know logarithms you might be puzzled that the natural logarithm is "log" and not "ln", as in some other programs. This is just the way R works.

The last example is an example of a logical question: "is 3 less than 10". There are many other logical questions that can be asked. For example '==' means "are things equal". We will see other examples of this and how they are used later.

The log, log10, and sin examples are also examples of what are called functions. Anything in R with text followed by something within parentheses is a function. The part that it inside the parentheses are called the argument. R uses functions to do all kinds of things for you. That is there is some behind the scenes stuff going on to calculate log(10). Some functions have single arguments, but others may have 2 or more. Each argument is separated by a comma. A very useful one is:

```
seq(from=0,to=10,by=1)
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10
```

which returns a sequence of numbers from 0 to 10. You could alter this to read:

```
seq(from=0,to=10,by=0.5)
```

```
##  [1]  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0
## [16]  7.5  8.0  8.5  9.0  9.5 10.0
```

which gives the same range of numbers, but every 0.5 instead.

We can also take these sequences and give them a name. To name something in R simply write:

```
x <- seq(from=0,to=10,by=1)
```

x is what is called an object. Now if I type 'x' into the console, then it should return the sequence

```
x
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10
```

You can manipulate objects as you would numbers. For example:

```
x <- 5
y <- 10
x + y
```

```
## [1] 15
```

You can also see that 'x' and 'y' show up in the box at the upper right of your screen, the "Global Environment". This is R's memory.

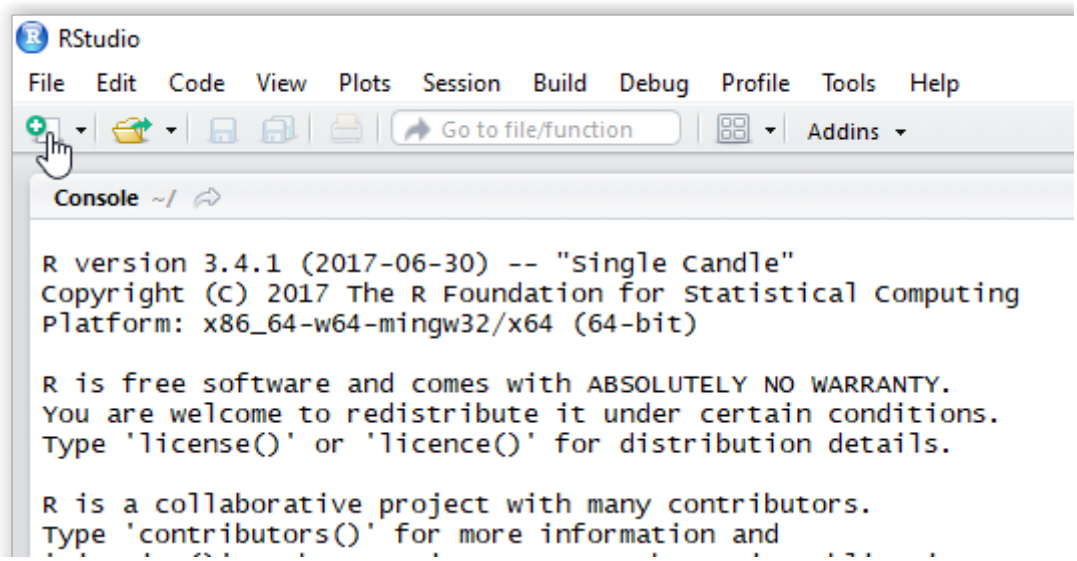


Figure 2.3: Click on the icon to open a new script.

4. Your first script

So far we have been using the console to learn how to use R like a giant calculator. Now, we will do this but we want to keep a record of what we have done. Doing so involves writing scripts. We also want to learn how to organize our workspace.

First, move the mouse to the upper left of the screen (as in Figure 3) and open a new script.

Your screen should now look similar to Figure 4, with four sections on the screen. The console has moved down to the lower left and a new blank script appears in the upper left.

Within the new script, you can type in the commands as we had done before. To run the line of code, place the cursor on the line and hit the 'run' key (Figure 5) or "Ctrl + Enter".

The real power of the scripts is that you can type stuff into it, run it, and save the code for later use, modification, or to share with others. Let's try this. In the script, type the following:

```
x <- 5
y <- 10
z <- x * y / x^2
```

Then press the save icon as in Figure 6.

You will be prompted to select a location on your computer to save the script. Start by setting up your directory for the laboratory in this module.

The aim here is to make sure that you can locate your work easily-organization is an important part of reproducibility. The most important person to understand your work at a later time is you! For example, you will need to refer to your scripts when are preparing your write-ups, and good directory structure will make this process feasible.

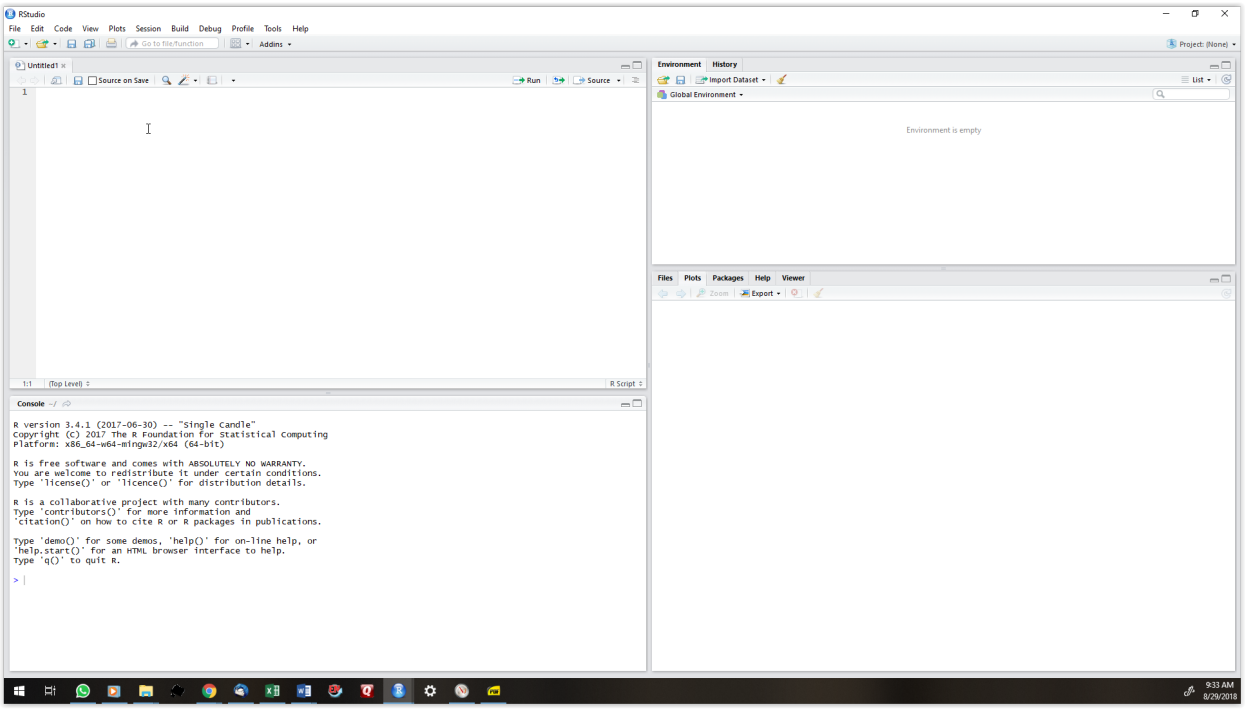


Figure 2.4: Computer screen with new blank script opened.

Use an organization structure that works for you. One strategy suggested in the literature (Noble 2009) is to make an overall directory for all your work in the course with a separate subdirectory for each separate project (for example, each worksheet). Within each project subdirectory, have the following three directories:

-A Data directory - contains data files including Excel workbook and .csv files

-A Results directory - contains the R scripts that read in and analyze the .csv files

-A Documents directory - contains written documents that summarize your work.

Go ahead and set up a location for you to store your work now. For now, create a folder in the Documents folder called 'Ecology'. Then within that one create another folder called "Lab 1". You can then save this script within this folder with the name "My first script".

Congratulations, you have written your first script!

Now, what we want to do is to setup a script so that we can begin analyzing data and making figures. To do this, open a new script. There are a couple of things that should go at the top of just about every script you will write. The first is:

```
rm(list=ls())
```

rm(list=ls()) is a function that clears R's brain. Each time you begin a new session, you want to make sure that there are not any left overs from a previous working session.

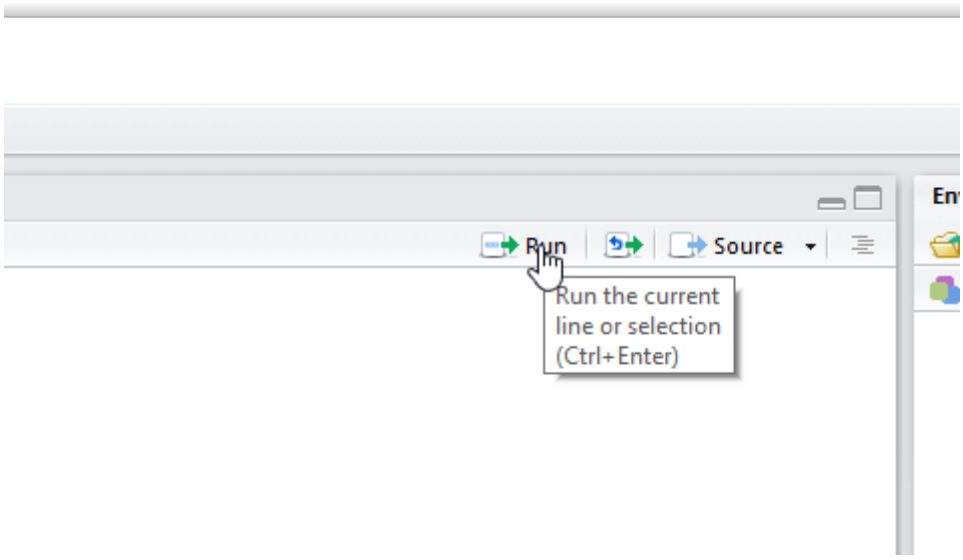


Figure 2.5: Click the 'Run' icon to run the line of code.

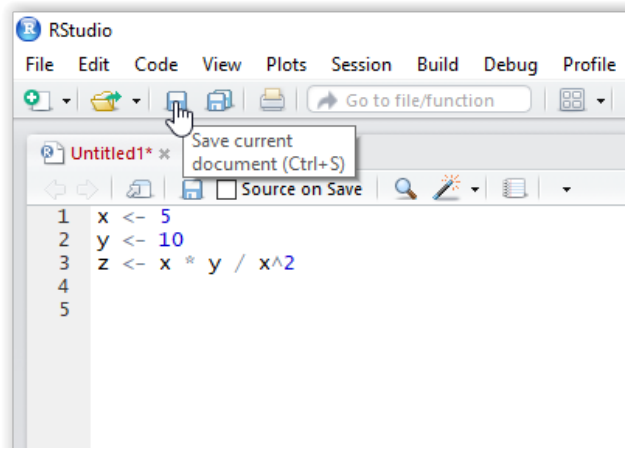


Figure 2.6: Save the script.

The second thing to write is a line of code that tells R where the working directory is on the computer. The working directory is the default location that R will look for stuff it needs. For now, let's make the same folder we created. Type:

```
setwd("~/Evolutionary Ecology/Lab 1")
```

You probably have guessed by now that 'setwd()' is a function, and the bit within the parentheses is the argument that tells R where the directory is located.

Now, let's save the script. Call it 'Lab 1 script'. Once you are done with this, your script should look just like figure 2.7.

This all we need to do for now. We will start with this with every script that we write in the labs.

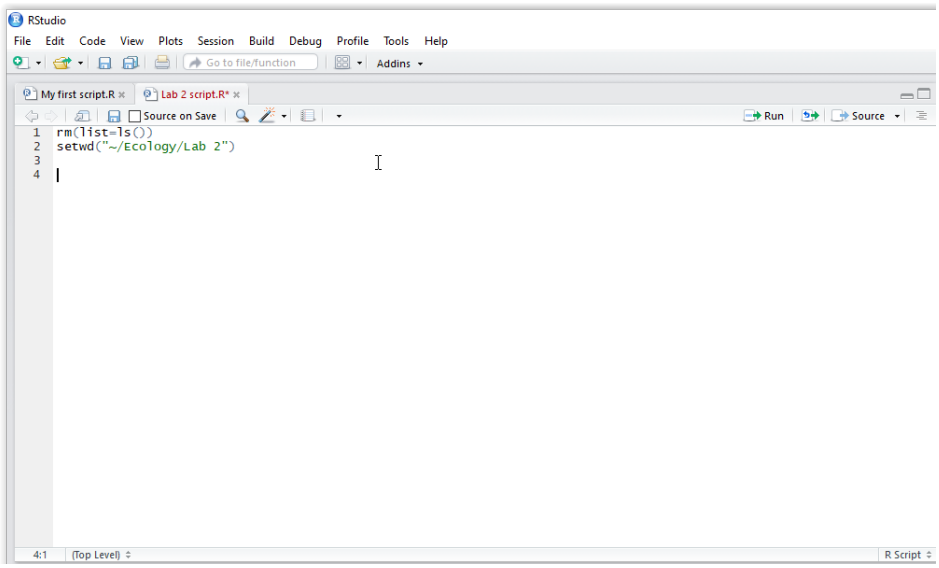


Figure 2.7: Lab 1 script (so far).

2.4 PART III. Linear regression and linear models

Linear regression is an important statistical tool to have in your toolkit. It is mostly used to analyze whether there is a causal relationship between two continuous variable (i.e. real numbers with with decimal places like heights, weights, etc). Learning regression is not only an important tool in and of itself, but is also can be used for other types of analyses when one or more of the variables are categorical (i.e. red, blue, or orange). There are many types of regression, but the simplest is simple linear regression, which is where we will start.

We will start by testing the biological hypothesis that fish growth declines with increased body size. This is a very simple hypothesis, but the purpose here is to focus on the mechanics of the statistics. Later, we will use this same data to test a more biologically interesting hypothesis.

Background to simple linear regression

Did you ever wonder why the formula for a straight line was drummed into your head in K-12 education and why we often continue to do so in first-year and sophomore level classes? It because it is the foundation for many many mathematical expressions, particularly in statistics. We can write the formula for a straight line as:

$$y = \alpha + \beta x \tag{2.1}$$

There are two variables and two parameters.The response variable is y and the predictor variable is x. The two parameters are α and β. The intercept of the line is α and the slope

of the line is β. The variables come from the data and we will try to fit the line to the data to yield estimates of the parameters.

Let's look at some data, which shows the somatic growth (in mm) of guppies over a 28-day period in artificial streams. First get the data,

```
rm(list=ls())

setwd("~/Evolutionary Ecology/Lab 1")

data <- read.csv("./Data/mesodata.csv")
head(data)
```

```
##     spp block channel guppy.phenotype density color.combo initial.length
## 1 guppy     1       1               HP     low          OY           11.30
## 2 guppy     1       1               HP     low          OR           15.30
## 3 guppy     1       1               HP     low          RY           18.04
## 4 guppy     1       1               HP     low          RB           17.20
## 5 guppy     1       1               HP     low          RR           21.50
## 6 guppy     1       1               HP     low          RO           22.84
##   final.length growth
## 1        16.26   4.96
## 2        19.19   3.89
## 3        20.95   2.91
## 4        21.21   4.01
## 5        23.01   1.51
## 6        23.93   1.09
```

Each row of the data is one for a fish that was in the experiment. There are a fair number of columns there, but the two we are presently concerned with are the "initial.length" and "growth" columns. The initial.length is the size (in mm) at the start of the experiment and "growth" is how much they changed in length over the experiment. How do we measure the length of a fish? We will use standard length (SL), which is a measure from the tip of the snout to the location where the fin rays contact the musculature in the tail.

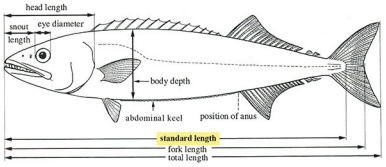


Figure 2.8: Standard length (SL)

The initial.length is our x and growth is our y. It is always important to start everything you do by plotting the data. There is no substitute for looking directly at the data. We can plot "growth" against "initial.length"

```
plot(data$initial.length,data$growth,xlab='Initial Length (mm)',ylab='Growth (mm)')
```

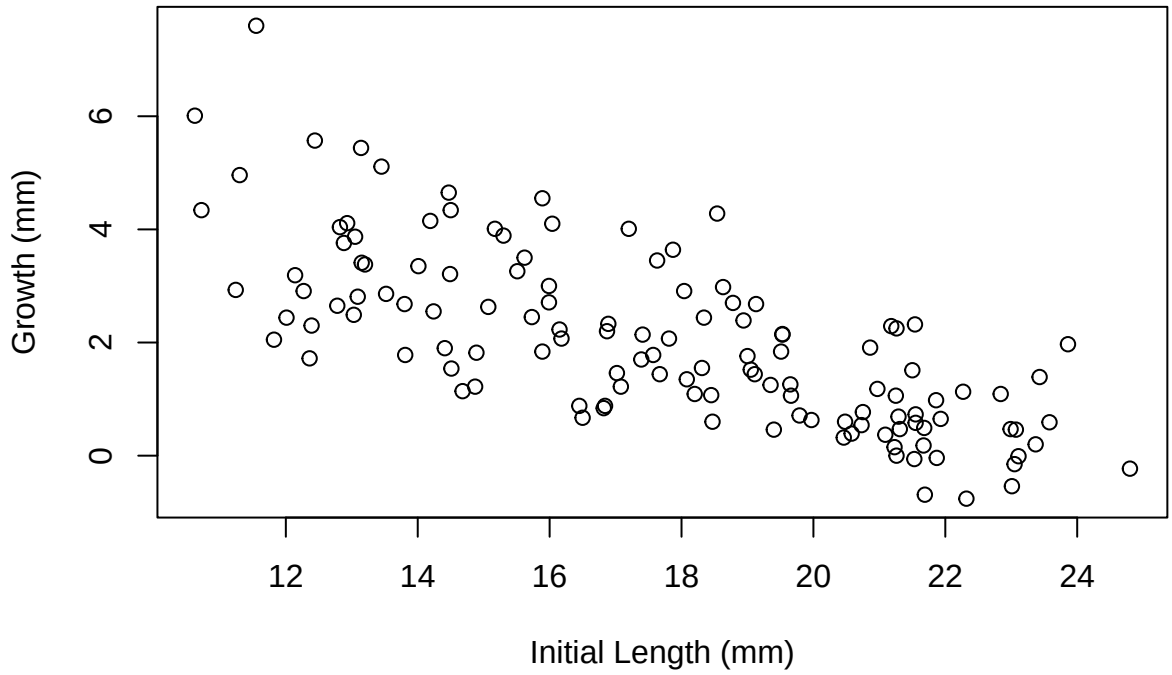


Figure 2.9: Growth versus initial length in the experiment.

In the code for the plot, the name of the data file (data), followed by '$', and the name of the column lets us select the data in that column. For example, to get all the initial length data:

```
data$initial.length
```

```
##   [1] 11.30 15.30 18.04 17.20 21.50 22.84 19.53 10.62 17.81 19.11 16.04 19.51
##  [13] 21.09 22.27 11.24 12.39 13.03 14.68 12.14 13.15 14.89 16.18 16.89 17.39
##  [25] 18.45 18.08 20.73 23.01 20.97 24.80 18.95 14.51 14.24 13.80 15.07 14.49
##  [37] 16.82 12.88 16.45 18.20 19.05 20.48 21.93 20.46 21.87 23.11 21.55 13.52
##  [49] 14.01 16.15 15.62 13.45 15.99 18.94 17.63 18.34 19.53 21.25 23.58 22.99
##  [61] 12.36 12.78 12.27 13.20 13.05 16.87 19.65 15.89 15.17 17.41 18.31 21.67
##  [73] 20.75 23.37 23.07 21.26 23.86 13.14 17.87 19.13 18.54 21.86 12.44 15.51
##  [85] 14.50 18.63 21.18 21.68 23.05 10.72 14.41 15.99 16.50 17.08 19.66 19.40
##  [97] 19.35 19.97 22.32 21.29 21.31 11.55 14.47 15.89 18.78 21.54 21.26 11.82
## [109] 12.01 13.09 17.67 18.47 13.81 21.53 16.84 14.87 17.02 17.57 20.58 19.79
## [121] 21.55 21.69 23.05 12.82 12.93 14.19 15.73 19.00 21.23 23.43 20.86
```

What we want to get is an objective set of parameters from the data that are the best possible estimates for these data. In modern statistics, the convention is to use the *maximum likelihood estimates* of the parameters. In other words, given the data and assuming the relationship is linear, we want to find the values of the slope and the intercept that are the most likely given the data.

For our simple linear regression, we have several assumptions.

1) The variance in y is constant (i.e. the variance does not change as y gets larger).
2) The predictor variable x (initial.length) is measured without error.
3) The difference between a measured value of y and the value predicted by the model for the same value of x is called a *residual*. These are the green lines in the figure below.
4) Residuals are measured on the scale of y.
5) The residuals are normally distributed.

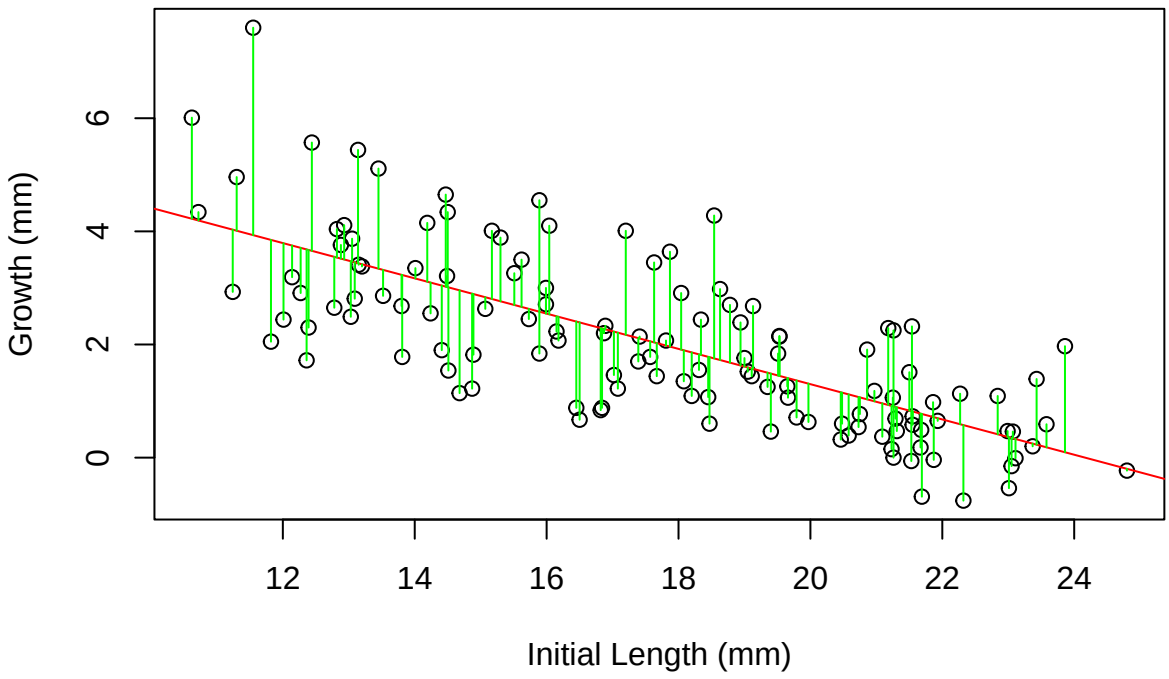


Figure 2.10: Growth versus initial length in the experiment with the predicted relationship in red and residuals in green.

If the above is true, then the maximum likelihood is given by the methods of *least squares*. This will not always be the case and we will see later in the class other situations. Although this is not a statistics class, least squares is among the most straightforward maximum likelihood estimator to understand, so let's take a quick look at it.

The residuals are the vertical distances between the data (open circles) and the red line. Each residual is a distance, d, and the value predicted by the model, $\hat{y}$, evaluated at the appropriate value of x:

$$d = y - \hat{y} \tag{2.2}$$

Now replace the predicted value, $\hat{y}$ by its formula: $\hat{y} = \alpha + \beta x$ to get:

$$d = y - \alpha - \beta x \tag{2.3}$$

Our measure of fit is the sum of the squared distance (residual).

$$\sum d = \sum (y - \alpha - \beta x)^2 \tag{2.4}$$

The sum of the raw residuals will always be zero, but the sum of their squares is not. The values of α and β that minimize the sum of the differences between the predicted values and the data, d, provides the best fit to the data. Another way to think about this is to imagine rotating the predicted line around

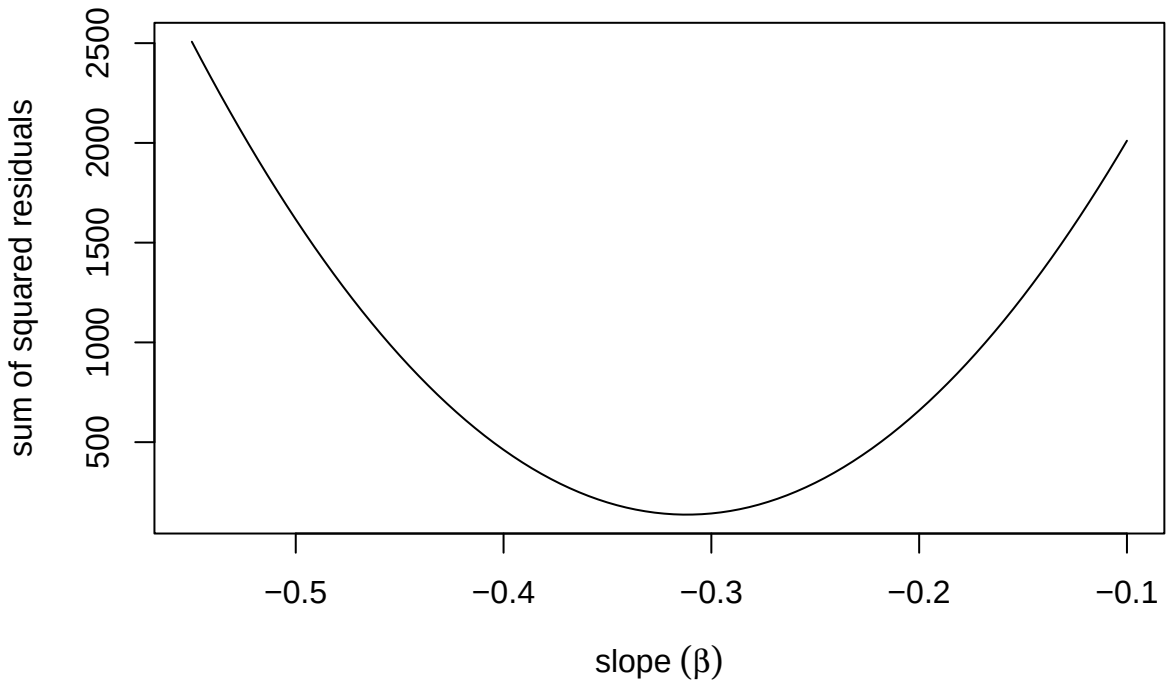


Figure 2.11: Sum of the squared residuals versus the slope.

The sum of the squared residuals is also called the *sum of squared errors* or the SSE. If you have taken derivative calculus, you know that you can find the point of the curve where the derivative $\frac{dSSE}{db}$ (slope of the line) is equal to zero, which is at the minimum point in the curve. Fortunately, we do not need to do the calculus as these relationships are well understood. It turns out all we need to find the maximum likelihood estimate of the slope is three quantities that are closely related to the variance and covariances of the two variables. These are:

$$SSY = \sum (y - \bar{y})^2 \tag{2.5}$$

$$SSX = \sum (x - \bar{x})^2 \tag{2.6}$$

$$SSXY = \sum (y - \bar{y})(x - \bar{x}) \tag{2.7}$$

The bar over the variables is the mean of the variable. The first two are the sums of the squared deviations of each of the variables from their means and the second is sum of the products of the deviations from the mean of the two variables. The maximum likelihood estimate of the slope is then simply:

$$b = \frac{SSXY}{SSX} \tag{2.8}$$

which is the covariance of x and y divided by the variance in x. This may seem familiar. For example, the heritability of a trait is the slope of the line between the mean offspring values and the mid-parent values. This slope is precisely what you are calculating here: the covariance between the mean offspring values and the mid-parent values divided by the phenotypic variance.

Conducting a simple linear regression

Let's put the background to linear regression aside and focus on actually using the computer to implement this and how to interpret the results. R has several functions that will run a linear regression. We will stick with "glm()" for now.

To estimate the equation of the best fit line between growth and the initial.length, type:

```
mod <- glm(growth ~ 1 + initial.length,data=data)
```

The "1" means to include an intercept parameter. We have saved the output as 'mod'. Type 'mod' to view the output.

```
mod
```

```
##
## Call:  glm(formula = growth ~ 1 + initial.length, data = data)
##
## Coefficients:
##    (Intercept)  initial.length
##         7.5345         -0.3118
##
## Degrees of Freedom: 128 Total (i.e. Null);  127 Residual
##   (2 observations deleted due to missingness)
## Null Deviance:       299.4
## Residual Deviance: 136.7     AIC: 379.6
```

This tells us that the estimate for the y-intercept is 7.5345 and the estimate for the slope is -0.3118. R uses slightly different language here than we have been using, but the "Residual Deviance" is the sum of the squared residuals or the SSE. We actually want a little more information than this from the model.

```
summary(mod)
```

```
##
## Call:
## glm(formula = growth ~ 1 + initial.length, data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9606  -0.7276  -0.1360   0.6771   3.6668
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     7.53449    0.45638   16.51   <2e-16 ***
## initial.length -0.31180    0.02536  -12.29   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.076358)
##
##     Null deviance: 299.42  on 128  degrees of freedom
## Residual deviance: 136.70  on 127  degrees of freedom
##   (2 observations deleted due to missingness)
## AIC: 379.56
##
## Number of Fisher Scoring iterations: 2
```

Gives us a bit more information. Now in addition to the estimates for each parameter, we also have a standard error, a t-value, and p-value. The standard error is a measure of how good our estimate of the parameter actually is. Smaller standard errors are better estimates than larger ones. The t-value is the parameter estimate divided by the standard error, which in effect says if the estimate is much larger than our error, then we can have some confidence that we are doing a good job estimating the parameter. This also gives us the Residual Deviance (SSE). The other value of interest here is the "Dispersion Parameter for the Gaussian Family". This values is the residual variance, i.e. the SSE divided by the number of degrees of freedom (127). Yet, one question remains: How good is good enough?

Tests of the statistical hypotheses

The biological hypothesis we are addressing is whether fish growth declines with increased body size. So far, we have just been asking about whether there is a relationship between initial size and growth. We used regression to come up with the parameters for the slope of a line that describes the relationship between these two variables. How confident are we that the relationship we are estimating reflects the truth? The statistical null hypothesis is one of no effect, which means that the null says that the slope should be 0. The alternative is that it is not 0. Recall from our discussion above is that all we can do with the statistics is to reject or fail to reject the null.

Ultimately, what we want to have some probability of rejecting the null hypothesis. Because our statement about the null will ultimately have a probability attached to it, we need talk about probability distributions. For almost all of the work we will do in this class the analyses rely upon a t-distribution. A t-distribution is a probability distribution that is bell-shaped.

The width of the t-distribution depends on the degrees of freedom. Degrees of freedom are a difficult concept to grasp in some analyses. Perhaps the simplest way to explain them is the number of independent pieces of information that went into calculating the estimate. For our data we have 129 independent data points. In regression like this, the number of degrees of freedom is the total number of observations minus the number of parameters to estimate. This gives us 127 degrees of freedom.

Now each of these distributions has the property that the area under the curve is equal to 1. Hence they are probability density distributions. The values along the x-axis are the possible t-values. The areas to the right of any number tell us the probability of getting that t-value or higher by chance. Likewise, the areas to the left of any t-value tell us the chance of getting that t-value or lower solely due to chance. For us, our t-value of -12.29 is off the chart to the left, meaning there is a very small probability of getting that value by chance alone. The probability to the left of our t-value is the p-value. Technically, it is the one-sided p-value. Students that have taken stats will know that to get the two-sided p-value, it is twice this value. The p-value technically tells us the probability that we are rejecting a true null hypothesis (Type I error). When this value is very small (usually less than 0.05), we can be certain to reject the null in favor of the alternative. Our very low p-value gives us confidence that we can reject the null hypothesis that there is not a relationship between initial size and growth rate.

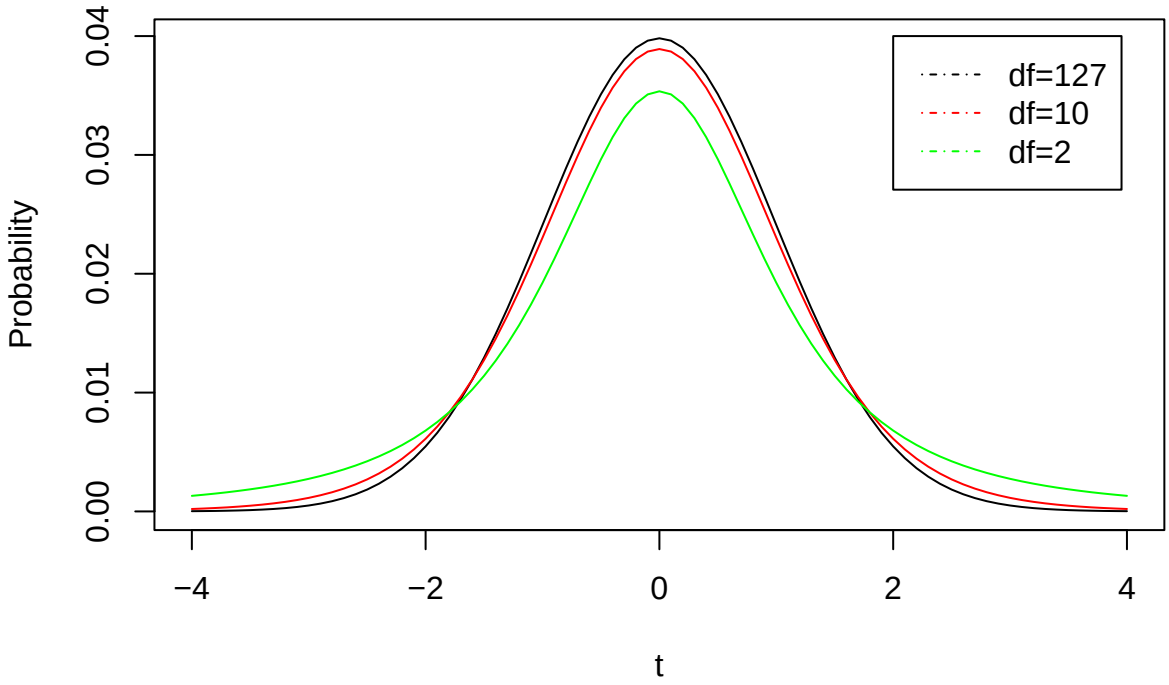


Figure 2.12: A student t-distribution with 3 different degrees of freedom. The area under each curve is 1 corresponding to all possible values of t.

The reason why the degrees of freedom are important is that we need to have a higher t-value to make conclusions when we have lower degrees of freedom. Thus, the less data you have the larger the effect needs to be to be confident that the estimate is different from 0.

A note on *probability levels*. Keep in mind that statistical tests cannot tell us whether or not our conclusions are correct, but instead give us the probability that we are wrong. Each statistical test gives us the probability of falsely concluding that a difference exists, when in reality there is no difference. The generally accepted significance level in scientific work is 5%. That means that 5% of the time we will conclude that differences exist when there really is no difference. We can be more confident that our conclusions are correct when statistical tests give us probability levels of 1% or lower. These probabilities are usually written like this: *p = 0.05 for a 5% chance of being wrong, p = 0.01 for a 1% chance of being wrong.*

2.5 PART IV. Testing the hypothesis that predation selects for higher growth rates

We are now in a position to test whether predation causes selection for and the evolution of higher growth rates. Let's look at the head of our data file again.

```
head(data)
```

```
##     spp block channel guppy.phenotype density color.combo initial.length
## 1 guppy     1       1               HP     low          OY          11.30
## 2 guppy     1       1               HP     low          OR          15.30
## 3 guppy     1       1               HP     low          RY          18.04
## 4 guppy     1       1               HP     low          RB          17.20
## 5 guppy     1       1               HP     low          RR          21.50
## 6 guppy     1       1               HP     low          RO          22.84
##   final.length growth
## 1        16.26   4.96
## 2        19.19   3.89
## 3        20.95   2.91
## 4        21.21   4.01
## 5        23.01   1.51
## 6        23.93   1.09
```

There is a column called guppy.phenotype that is coded with LP or HP, for low predation and high predation phenotypes, respectively. One of the powerful things about regression is that it forms the basis for what are called general linear models. General linear models allow us to include categorical predictors into an analysis alongside continuous ones. Let's try this using guppy.phenotype.

```
mod <- glm(growth ~ 1 + initial.length + guppy.phenotype, data=data)
summary(mod)
```

```
##
## Call:
## glm(formula = growth ~ 1 + initial.length + guppy.phenotype,
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8774  -0.6722  -0.1758   0.6530   3.7479
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        7.48372    0.45833  16.328   <2e-16 ***
## initial.length    -0.31443    0.02545 -12.355   <2e-16 ***
## guppy.phenotypeLP  0.20193    0.18348   1.101    0.273
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.074571)
##
##     Null deviance: 299.42  on 128  degrees of freedom
## Residual deviance: 135.40  on 126  degrees of freedom
##   (2 observations deleted due to missingness)
## AIC: 380.33
##
## Number of Fisher Scoring iterations: 2
```

Now we have an additional line that says guppy.phenotypeLP. It has a parameter estimate, standard error, t-value, and p-value, just like the others. The parameter itself has a value of 0.20193. But, how does R know what to do with the values in the guppy.phenotype column? And why do they have a parameter? What does this mean? It turns out that R does not use LP and HP directly. It changes HP to 0 and LP to 1. Let's see how this works. Let's forget about size for the moment so that we are just dealing with guppy.phenotype. Our new linear regression equation is:

$$y = \alpha + \phi p \tag{2.9}$$

where p stands for phenotype of guppy and ϕ is the parameter for this variable. Now some of the p's are 0 and some are 1's, corresponding to HP and LP, respectively. If p is 0 (HP), then only α is being estimated. Hence α corresponds to HP fish. So now, the intercept corresponds to the value for HP fish. If ϕ is 1 (LP), then we are estimating $\alpha + \phi$. This is

the value for LP fish. So ϕ tells us the difference between HP and LP. This is nice because this is exactly what we want to test.

Let's look back at our output. guppy.phenotypeLP means that, on average, we observed LP guppies to have grown 0.20193 mm more than HP guppies. Do we have much confidence in this difference? No, the p-value is 0.273. Here we failed to reject the null hypothesis that this not a difference in the growth rates of HP and LP guppies. And hence, we failed to find evidence to support the biological hypothesis that predation selects for higher growth rates. Again, this does not mean that the hypothesis is not correct, just that we failed to find evidence for it.

Often times we fail to reject the null (and find evidence for our biological hypothesis) because there are other factors that impede us from seeing it. This is sometimes an experimental design and statistical power issue. Other times, it is because the biological hypothesis is conditioned upon certain circumstances. This conditioning often means that a more refined biological hypothesis is needed. In lecture, we have talked about r and K-selection. There were figures that showed how which genotype is fittest depends upon the density of individuals in the population. The crossed lines are indicative of an interaction between the factors, in this case between density and predation. An alternative hypothesis to the predation selects for higher growth rates hypothesis is that growth rates depend on both predation and resource availability. Organisms that live in locations with predators can grow fast because they have high levels of resources and do not need to be efficient at gathering them. In contrast, organisms that live without predators will have low resource availability and will need to be efficient at resource acquisition. This predicts that high predation guppies should grow well at low densities, but be worse growers at high densities. The opposite will be true for low predation guppies. We can test this hypothesis using an interaction term in the linear models.

```
mod <- glm(growth ~ 1 + initial.length + density +
             guppy.phenotype + guppy.phenotype:density , data=data)
summary(mod)
```

```
##
## Call:
## glm(formula = growth ~ 1 + initial.length + density + guppy.phenotype +
##     guppy.phenotype:density, data = data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.52045  -0.48789   0.00806   0.42893   2.33303
##
## Coefficients:
##                                Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     7.06240    0.33722  20.943  < 2e-16 ***
## initial.length                 -0.32151    0.01854 -17.345  < 2e-16 ***
```

```
## densitylow                     1.91801    0.20458   9.375  4.1e-16 ***
## guppy.phenotypeLP              0.40076    0.16095   2.490  0.01410 *
## densitylow:guppy.phenotypeLP -0.88507    0.28777  -3.076  0.00259 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.5692476)
##
##     Null deviance: 299.415  on 128  degrees of freedom
## Residual deviance:  70.587  on 124  degrees of freedom
##   (2 observations deleted due to missingness)
## AIC: 300.3
##
## Number of Fisher Scoring iterations: 2
```

Well, now there you go. Here there is a significant interaction between guppy.phenotype and density (which is a proxy for resource availability), meaning that how predation influences growth rate depends on the density (i.e. resource availability). Let's plot the means of this to look to see what it looks like.

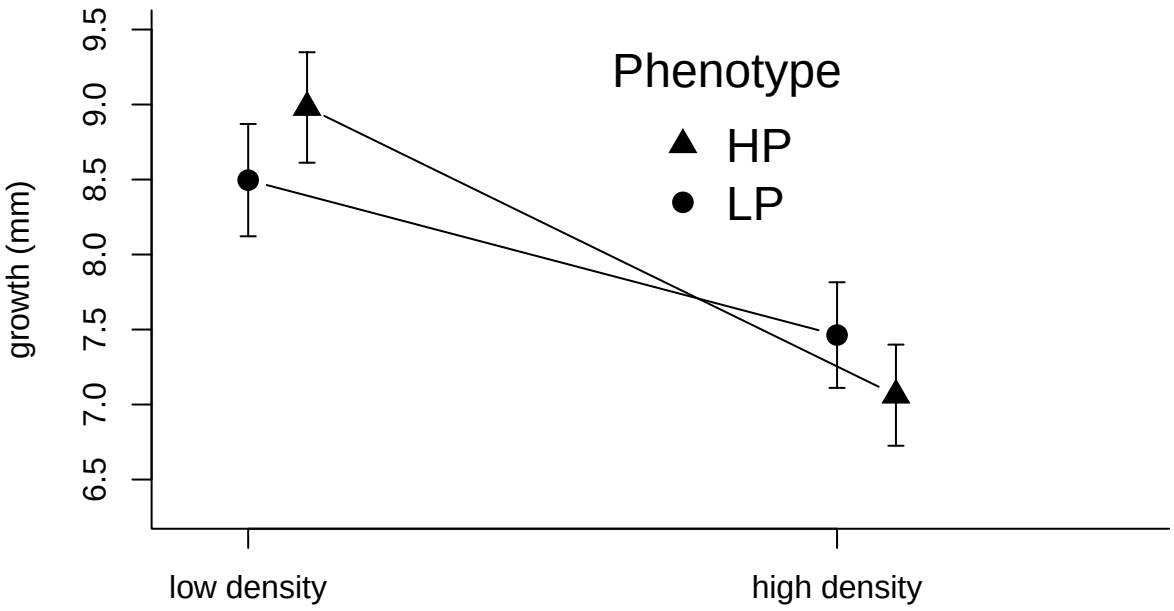


Figure 2.13: Somatic growth in the factorial experiment.

Now, why does this work? Again, lets go back to our equation for the regression and add two more terms. One for density (low or high) and one for the interaction between phenotype and density. Again, we will ignore initial size for this explanation.

$$y = \alpha + \phi p + \delta d + \iota p d \tag{2.10}$$

Now, assume again that a p of 0 stands for HP and a p of 1 is for LP. Additionally, a d of 1 stands for low density and a d of 0 stands for high density. Now there are more than two combinations of possibilities. Let's start with the case for when p and d are both zero. This results in:

$$\alpha \tag{2.11}$$

when p and d are zero this is for HP fish at high densities. And so the α parameter is for HP at high density. As before, we can add in the ϕ.

$$\alpha + \phi p \tag{2.12}$$

This results in $\alpha + \phi$ or the value for LP at high density. It is at high density because we did not change d. Now, instead of p, let's change d from 0 to 1, which is for low density.

$$\alpha + \delta d \tag{2.13}$$

Now the results is $\alpha + \delta$, which is the value for HP at low density. So far, we have three combinations. The final one is to change both p and d to 1's from 0. This is then for LP at low density and is:

$$\alpha + \phi p + \delta d + \iota p d \tag{2.14}$$

Thus, the value for LP at low density is $\alpha + \phi + \delta + \iota$. The interpretation of ι is that it expresses how much the pattern between the two phenotypes depends on density. Take another look at the plot above. At low density, HP, has higher growth, but at high density, LP have higher growth. Another way to look at this is that the interaction tells you how much the two lines cross in the figure. If the relative growth rates of the two phenotypes did not depend on density, then the lines would be parallel.

Because the the interaction is less than 0.05, we reject the null statistical hypothesis for the interaction that the lines are parallel and accept the alternative that they are not parallel.

Part IV Exercises:

You can rip this sheet off and hand it in with your answers. No coding is necessary.

1. One reason for looking at growth rate is that it can be used as a proxy for fitness. If this is the case and growth is a good measure of the population growth rate under different conditions (e.g. density), then what might you be able to conclude about which phenotype of guppy should evolve in low and high density habitats?

2. Imagine you start off a population with a mixture of low and high predation guppies at low densities. If somatic growth is again a good proxy for the fitness of each phenotype and phenotypes do not interbreed, how would you expect the relative proportions of each phenotype to change over time?

Chapter 3

Population Growth, Mean Fitness, and Changing Fitness Measures

3.1 Introduction

The figure below shows the population dynamics of four guppy populations over a 142 month period. The dynamics differ considerably from what we have been talking about in lecture. All of that has involved elegant formulas with smooth curves. The figure on the front is reality. The models are abstractions of this reality. One important thing to remember is that "All models are wrong, but some are useful". The meaning here is that the point of models are to boil something down to its bare essence so it, and the consequences of changing it, can be studied. Still going from these elegant models to real data can be quite challenging. We will begin this process in this lab.

We will do so by getting familiar with the population growth equations that we have been talking about in lecture. In the first part, we will make up some data (parameters for the equations) and look to see how our assumptions change the types of growth that we get. In the second section we will be using actual data from a well-studied population of Trinidadian guppies (*Poecilia reticulata*). Guppies in Trinidad have been the subject of much research in Evolutionary and Ecology. Partly because we can study them in the wild and in the laboratory. The fact that we can do this sets them apart from many species that are studied in Evolutionary Biology (e.g. *Drosophila*) or ecology (e.g. Red Deer).

The data we will be using comes from a long-term study of individual guppies living in four populations that were introduced into guppy-free parts of four streams in 2008. We census the populations each month to count the number of individuals, and other things. For our lab today, the fact that very few guppies were introduced means that they grew very rapidly, but then appear to level off. We will use data from one of the streams to test whether the population does indeed exhibit constrained population growth. Future labs will this data at a more sophisticated level to address other questions in evolutionary ecology. In part three, we will see how population growth emerges from the growth of individual clones, each with different phenotypes. This will allow us to see how the growth rate of the population, the

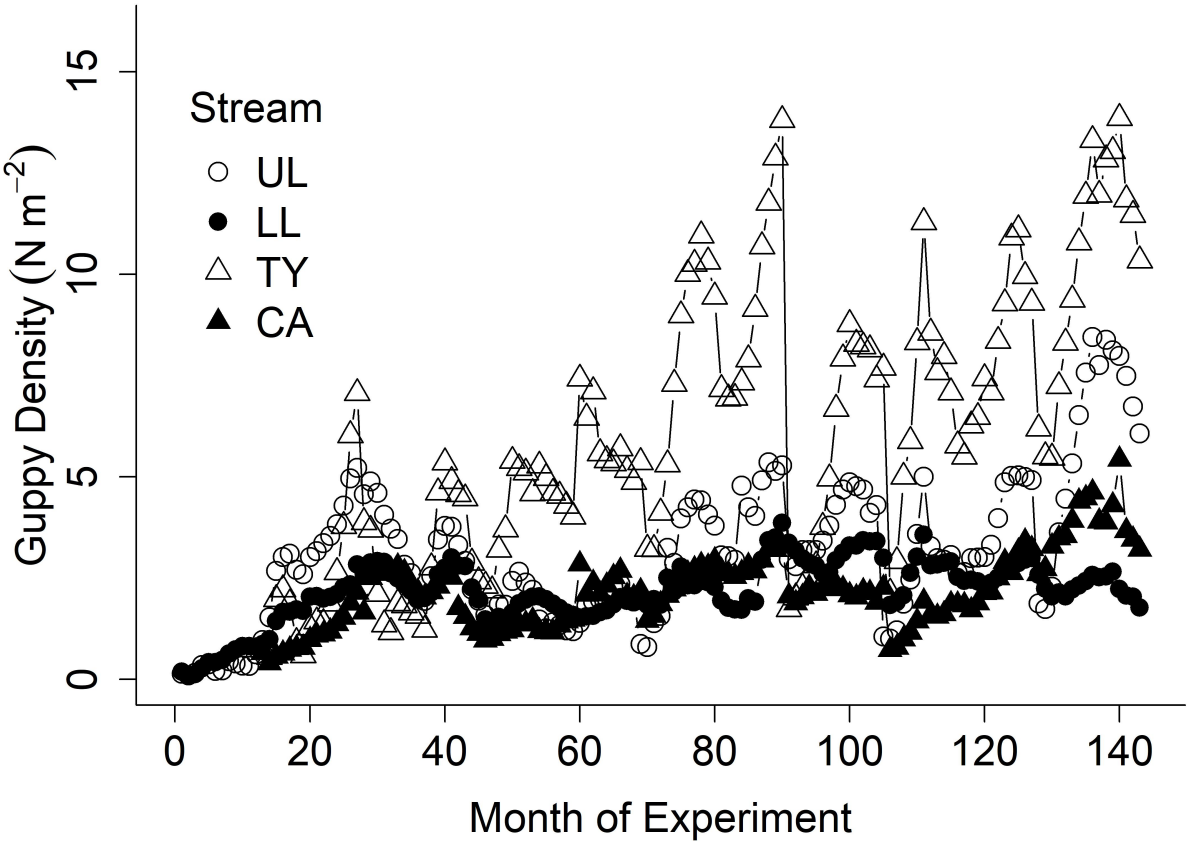


Figure 3.1: Population densities of guppies in four experimental streams over the first 142 months of the experiment.

mean and variance phenotypic values, and population size changes as evolution proceeds.

3.2 PART I: Exponential and Logistic Population Growth in Continuous and Discrete Time

In this section and the next, we will be creating some simple exponential and logistic growth curves. We will calculate the population sizes over some time period using the computer and compare it to the analytical solutions.

Clear R's brain and set the working directory.

```
rm(list=ls())
```

```
setwd("~/Evolutionary Ecology/Lab 2")
```

Exponential Growth in Continuous Time

The first thing we need to do is to set the amount of time. We can do this with the following piece of code that creates a sequence of the numbers from 0 to 200, taken every 0.01.

```
times <- seq(0,200,by=0.01)
```

To see this, type the following:

```
head(times)
```

```
## [1] 0.00 0.01 0.02 0.03 0.04 0.05
```

The function head allows you to see the first 6 times.

Next we need to create an array to store the population sizes.

```
N <- array(NA,c(length(times)))
```

This creates an empty vector that has the same length as times. Take a look at the first 6.

```
head(N)
```

```
## [1] NA NA NA NA NA NA
```

Next, we set the initial number of individuals (i.e. at time t=0) by putting a value into the first position of the empty vector.

```
N[1] <- 1
```

Then, set the birth and death rates. We can start with these values, but here and throughout, you can play around with these to see how it changes things.

```
b <- 0.1
d <- 0.05
```

The next part is to actually project the population through time. We can do this using a loop. A loop tells the computer to do something over and over again, looping around to the beginning each time. Each time it goes through the loop, it will be calculating the growth rate and the population size at the next time step. In order to project using the continuous time model, we to calculate the population growth rate (the first line) and then use this to approximate what the population size will be at the next time step. This approximation depends on the time-step we use. In the second line, the bit of code "(times[2] - times[1])" tells the computer the difference between the time periods are. We will that our approximation bets better as we have a smaller time-step.

```
for (t in 1:c(length(times)-1)){ # The minus 1 is there so that we
                                 #project only to the last time step.

  dN.dt <- (b-d)*N[t]
  N[t+1] <- dN.dt * (times[2] - times[1]) + N[t]

}
```

We can then plot the output with the following code.

```
plot(times, N)
```

Then, let's take a look at how this compares to the analytical solution.

```
N[length(times)]
```

```
## [1] 21971.49
```

The analytical solution is:

```
exp((b-d)*200) * N[1]
```

```
## [1] 22026.47
```

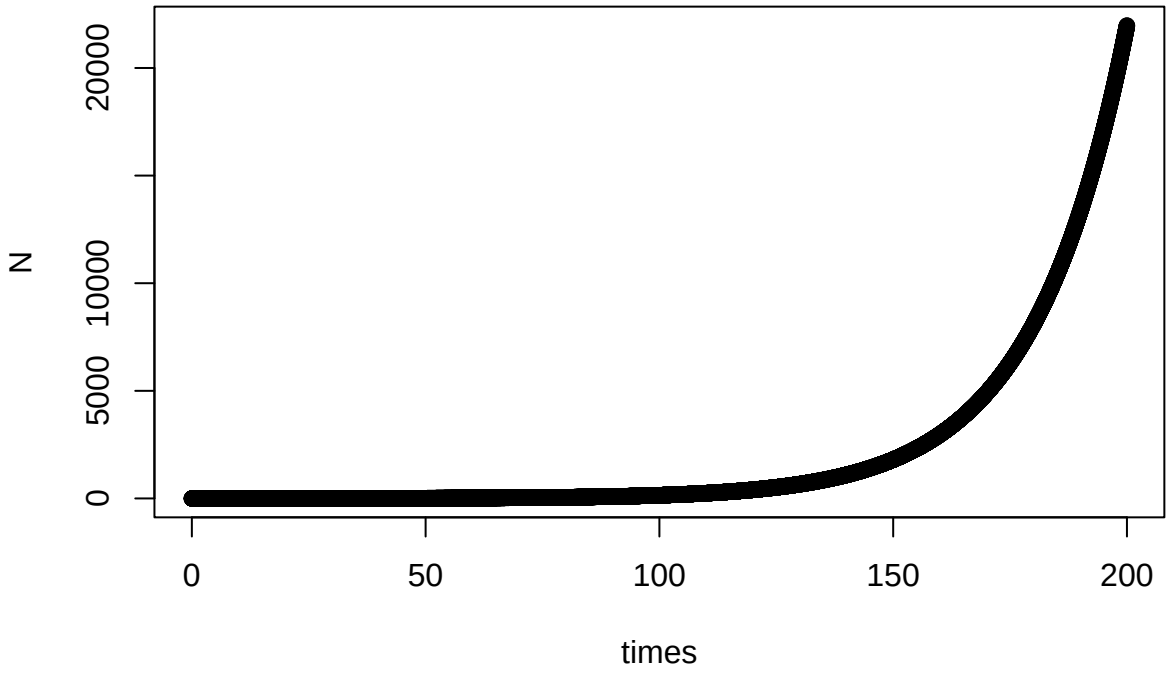


Figure 3.2: Exponential growth

Logistic Population Growth in Continuous Time

Let's then do the same thing for logistic population growth. You can keep the same time periods as before. You will need to reset the birth, death rates, and create a new parameter, K.

```
#Create an array to store the population sizes
N <- array(NA,c(length(times)))

#set the initial number of individuals (i.e. at time t=0)
N[1] <- 1

#set the birth, death rates, and carrying capacity
b <- 0.1
d <- 0.05
K <- 50
```

Then we can run it through a loop. Notice that the only thing that is different here is the population growth rate equation, which has the adjustment for limited growth.

```
for (t in 1:c(length(times)-1)){ # The minus 1 is there so that
                                 #we project only to the last time step.

  dN.dt <- (b-d) * (1 - N[t]/K) * N[t]
  N[t+1] <- dN.dt * (times[2] - times[1]) + N[t]
}
```

Now plot it, and compare the population sizes we got to the analytical solution.

```
plot(times, N)
```

```
N[length(times)] #Our value
```

```
## [1] 49.88908
```

```
K / (1 + (K - N[1])/N[1] * exp(-(b-d)*length(times))) #Actual value
```

```
## [1] 50
```

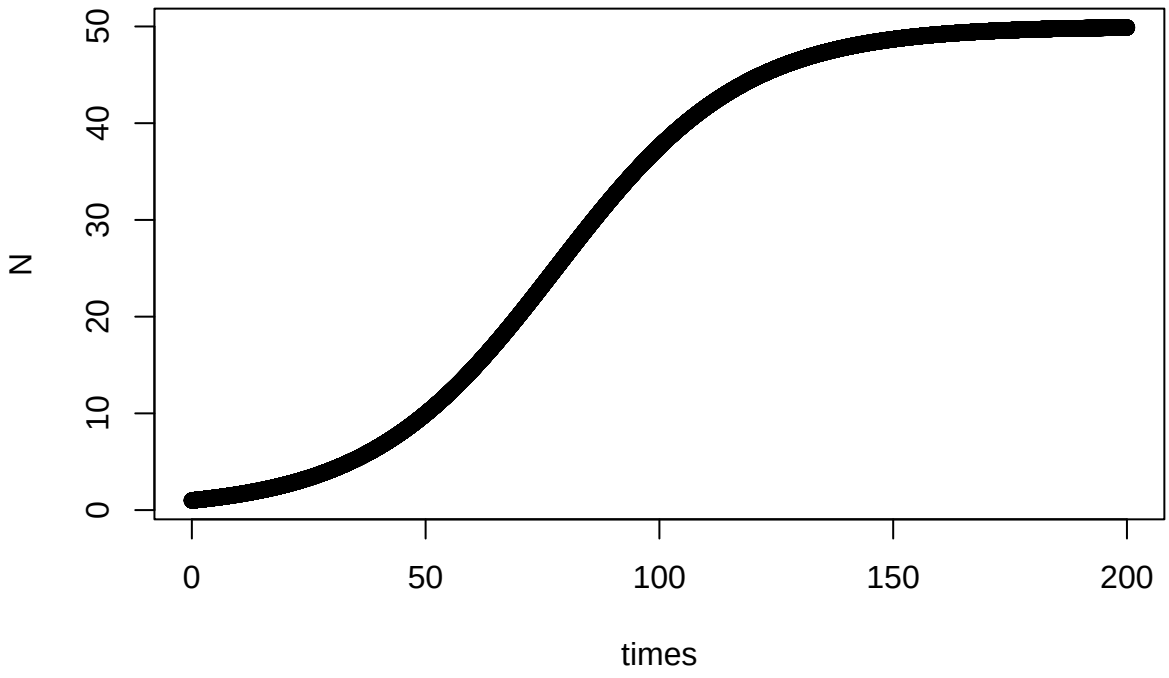


Figure 3.3: Logistic growth

Geometric Population Growth

Because finding the numerical solution to the continuous time models often takes a bit of time, we often will use geometric growth instead when working on the computer. Geometric growth and exponential growth are equivalent given that we use the appropriate time scaling.

Set the times. This time, the interval is 1.

```
times <- seq(0,200,by=1) #now the interval is 1
```

Create an array to store the population sizes and set the initial population size.

```
N <- array(NA,c(length(times)))
N[1] <- 1
```

Set lambda using the instantaneous birth and survival rates.

```
lam <- exp(b - d) * 1
```

Create the loop

```
for (t in 1:c(length(times)-1)){

  N[t+1] <- lam * N[t]

}
```

Plot it and compare it to the analytical solution

```
plot(times, N)
```

```
N[length(times)]
```

```
## [1] 22026.47
```

```
#Compare with the analytical solution
lam^times[length(times)] * N[1]
```

```
## [1] 22026.47
```

Now they are precisely the same. They are also the same as the analytical solution to the continuous model of exponential growth. So geometric growth gives us a quick and efficient way to convert continuous time models into something more manageable.

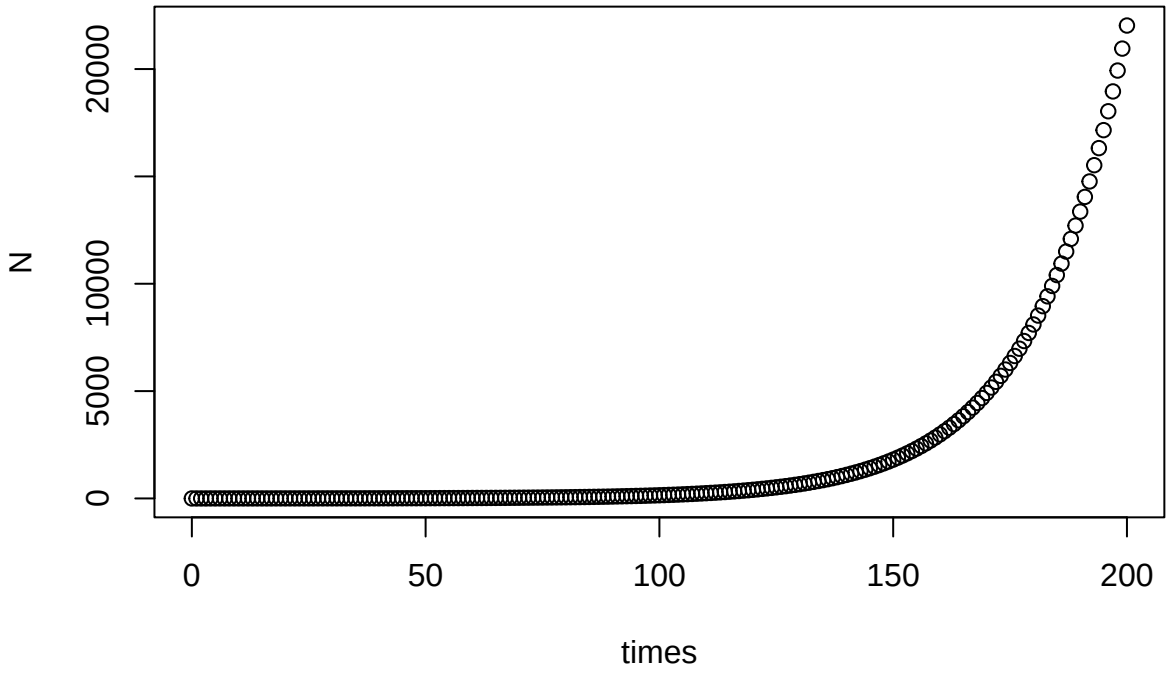


Figure 3.4: Geometric growth

Part I: Exercises

1) For the approximations of exponential and logistic growth models, do it 10 times. Each time you do it, decrease the time interval. Make a figure with the estimated population size on the y-axis and the difference in the time periods on the x-axis.

2) How would you decide which time interval to use?

3.3 Part II. Using real data

We have been looking at some generic growth curves. Here we will be working with some data from a Trinidadian guppy population.

The first thing we need to do is to upload the data into R's brain. The data file is on Canvas. Put it into your data directory for this lab. You can then upload it using:

```
data <- read.csv(file = "./Data/summaryGuppyDataLL.csv")
```

Look at the top few rows

```
head(data)
```

```
##   stream sample N.F N.M
## 1     LL      1  38  38
## 2     LL      2  16  22
## 3     LL      3  22  23
## 4     LL      4  56  42
## 5     LL      5  94  75
## 6     LL      6  89  57
```

There are only four columns that we need to concern ourselves with here. Later, we will be going deeper into these data. The first column is the name of the stream "LL", the second is "sample" which tells us the month of the experiment. The third and fourth are the numbers of males and females respectively.

Let's first combine the male's and females into another variable called "total.N"

```
data$total.Nt <- data$N.F + data$N.M
```

Then, we can plot this.

```
plot(data$sample,data$total.Nt,xlab='Month of experiment',
                ylab='Total population size')
```

So following the introduction into a new habitat, the populations increased over the first two years or so and then leveled off. Following the leveling off, the populations bounced up and down. This is driven by seasonal fluctuations. Guppy populations are high in the tropical dry season and lower in the rainy season.

Does this growth curve resemble the exponential or logistic? A first look says that this looks more like the logistic. Is there an objective way to discriminate between the two hypotheses?

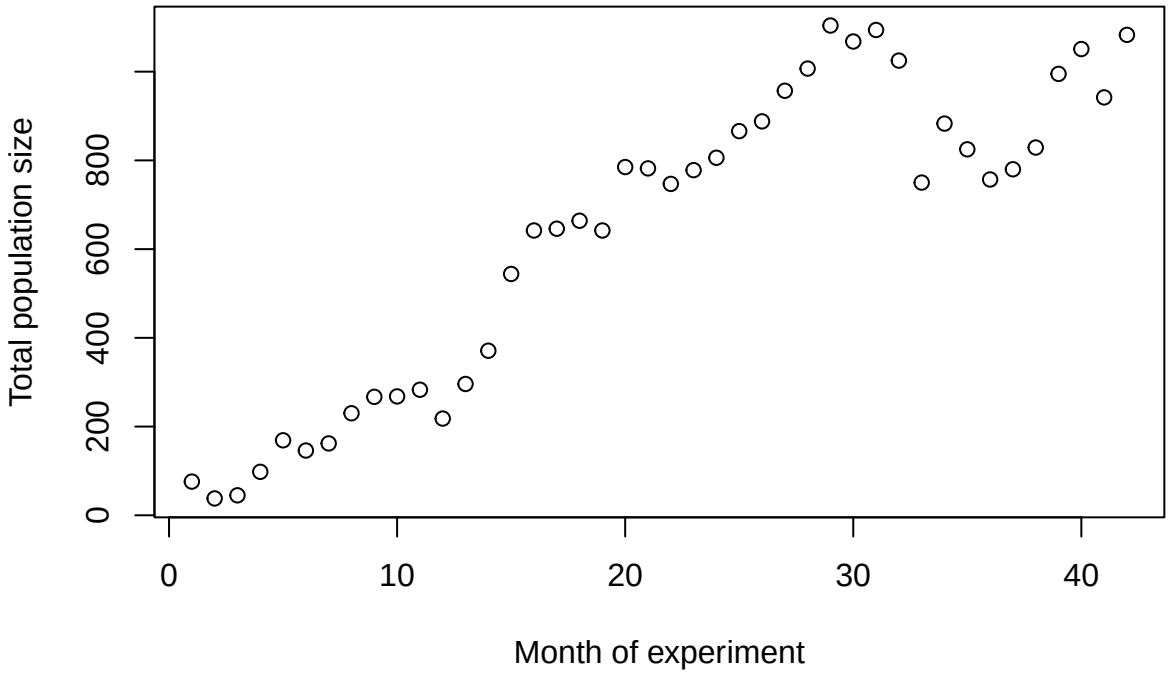


Figure 3.5: Population sizes in the first 42 months of the experiment.

Yes. To do so, we need to convert the population sizes to λ's. Recall that $\lambda(t)$ is just the population size at the next time step divided by the population size at the current time step. We can calculate this using the following code.

One way to do this is to create another variable that is lagged, so that it has the next value on the same row.

```
data$Nt.plus.1 <- c(data$total.Nt[c(2:c(length(data$total.Nt)))],NA)
```

Then λ is just the this new variable divided by the original one for that row.

```
data$lam.t <- data$Nt.plus.1 / data$total.Nt
head(data)
```

```
##   stream sample N.F N.M total.Nt Nt.plus.1     lam.t
## 1     LL      1  38  38       76        38 0.5000000
## 2     LL      2  16  22       38        45 1.1842105
## 3     LL      3  22  23       45        98 2.1777778
## 4     LL      4  56  42       98       169 1.7244898
## 5     LL      5  94  75      169       146 0.8639053
## 6     LL      6  89  57      146       162 1.1095890
```

Now plot it. This is usually done using the natural log of λ.

```
plot(data$total.Nt, log(data$lam.t),xlab='N(t)', ylab=expression(ln~lambda(t)))
```

Now, recall that the exponential growth model has a constant $ln\lambda$. In other words the birth and death rates do not change with increased population size. This forms our null hypothesis: H_0. This means that there should not be a significant relationship between $ln\lambda(t)$ and population size. If the relationship between $ln\lambda$ and population size is significant, then we can reject the null. If it is not significant, then we fail to reject the null. We can easily test this with a linear regression.

```
mod <- glm(log(lam.t) ~ 1 + total.Nt,data=data)
summary(mod)
```

```
##
## Call:
## glm(formula = log(lam.t) ~ 1 + total.Nt, data = data)
##
## Deviance Residuals:
##       Min         1Q     Median         3Q        Max
## -0.86286  -0.07840   0.00040   0.09836   0.60264
```

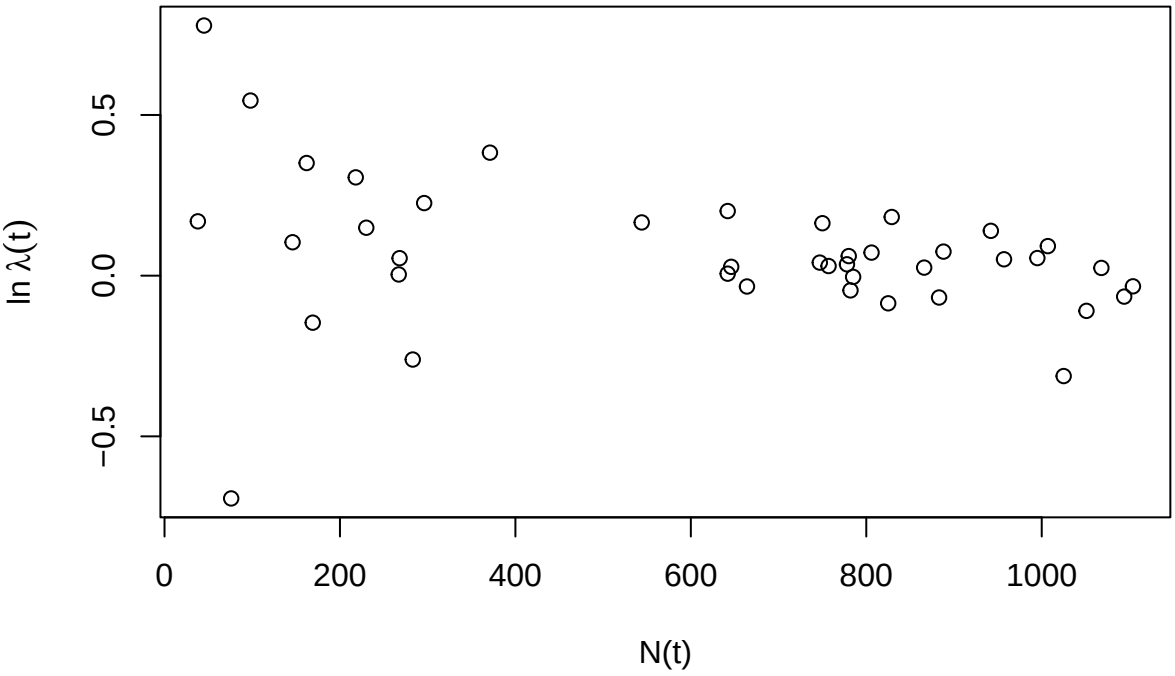


Figure 3.6: The natural log of lambda vs. population size.

```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.184304   0.072335   2.548   0.0149 *
## total.Nt    -0.000192   0.000102  -1.882   0.0674 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.04914869)
##
##     Null deviance: 2.0908  on 40  degrees of freedom
## Residual deviance: 1.9168  on 39  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: -3.2266
##
## Number of Fisher Scoring iterations: 2
```

Here the slope of the line (shown by "total.Nt") is not significant ($p > 0.05$). So we have failed to reject the null hypothesis. Importantly, this does not mean that the null is true, it just means that we do not have enough data to say it is false.

We could have failed to reject the null for many reasons. One is that we did a terrible job finding all the fish in the stream and so our estimates of population size are poor. Accounting for poor detection is a very important component of doing research in evolution and ecology. In the next module, we will learn about how to deal with poor detection probabilities.

Part II: Exercises

1) Male and female are very different from each other and may not compete with each other for resources. Redo the above analyses looking for evidence of logistic growth, except this time, do it for males and females separately.

2) Notice that we are not trying to test whether the logistic is true. Only that the exponential is false. There are other growth curves that lead to constrained growth and falsifying the exponential would not mean that the logistic is true. How do you think you might test whether the data conforms to logistic growth or some other form of constrained growth?

3.4 Part III. Population growth as mean individual fitness

If you took ecology, you may have seen that when there is random variation in time that alters the death and birth rates, environmental stochastic, then the mean growth rate can predict the mean population size at any time in the future.

$$\bar{N} = N_0 e^{\bar{r}t} \tag{3.1}$$

The same is also true with demographic stochasticity, when birth and death rates fluctuate randomly due to chance, usually due to small numbers of individuals in the population. Both of these are stochastic processes. However, natural selection is not a stochastic process and so these equations are not valid for populations experiencing natural selection. However, at any point in time, the population growth rate, r, is the average growth rate of the phenotypes in the population at that time. However, the growth rate at the next time with be larger (usually) than the one before. There are formulas for this, but it is more instructive to use the computer to see how this works. To show this, let's say that individual varying in their birth rate and death rates.

First, let's say there a 1000 different clones in a population.

```
N.clones <- 1000
```

And lets say those 1000 clones have a trait that confers a fitness advantage or disadvantage and that trait ranges from 0 to 100.

```
phenos <- seq(0,100,length.out=N.clones)
```

Each phenotype has a birth rate and a death rate. The nature of these are that clones with larger phenotypic values have higher growth rates.

```
r <- 0.001 + 0.001 * phenos #dnorm(phenos,mean=50,sd=20) * 1
head(r)
```

```
## [1] 0.001000000 0.001100100 0.001200200 0.001300300 0.001400400 0.001500501
```

If each clone (phenotype) is initially represented by 1 individual, then the mean r for the population is

```
N <- rep(1,N.clones)

sum(phenos * N / sum(N))
```

```
## [1] 50
```

And the mean r is:

```
sum(r * N / sum(N))
```

```
## [1] 0.051
```

Let's project the populations in time using the geometric growth. Identify a time frame.

```
time <- 100
```

First, using the mean growth rate at the beginning.

```
N.mn <- exp(mean(r)*time) * sum(array(1,c(N.clones)))
sum(N.mn)
```

```
## [1] 164021.9
```

Then using the whole distribution of clone phenotypes.

```
N.dist <- exp(r*time) * array(1,c(N.clones))
sum(N.dist)
```

```
## [1] 2443949
```

These values are quite distinct. The value using the whole distribution is larger than the one that uses the mean. This is because the clones with low r's are declining in numbers through time and the ones with higher r's are increasing more rapidly through time.

Let's do this again, but keep track of the population sizes of each phenotype at each step.

```
Nt <- array(NA,c(time,N.clones))
Nt[1,] <- 1
for (t in 1:(time-1)){
  Nt[t+1,] <- exp(r) * Nt[t,]
}
```

Now, let's look at the mean phenotype and mean growth rate through time.

```
mn.pheno <- array(NA,c(time))
var.pheno <- array(NA,c(time))
mn.r <- array(NA,c(time))
for (t in 1:time){
  mn.pheno[t] <- sum( phenos * Nt[t,]/sum(Nt[t,]))
  var.pheno[t] <- sum( (phenos - mn.pheno[t])^2 * Nt[t,]/sum(Nt[t,]))

  mn.r[t] <- sum( r * Nt[t,]/sum(Nt[t,]))
}
par(mfrow=c(2,2),mar = c(4,5,3,2),bty='L')
plot(c(1:time),mn.pheno,xlab='time',ylab='mean phenotype')
title(main='A)',adj = 0.1, line = 0)
plot(c(1:time),var.pheno,xlab='time',ylab='var phenotype')
title(main='B)',adj = 0.1, line = 0)
plot(c(1:time),mn.r,xlab='time',ylab='r')
title(main='C)',adj = 0.1, line = 0)
plot(rowSums(Nt),xlab='time',ylab='population size')
title(main='D)',adj = 0.1, line = 0)
```

Notice that the mean phenotypic values increases through time along with the mean intrinsic rate of increase, r. The variance in the phenotypes decreases as clones at the left side of the phenotypic distribution decline relative to those at the right side. To the extent that the variation in the phenotypes represents variation in alleles, the population is evolving.

When there is genetic variation in the population and this genetic variation causes fitness differences, then natural selection will act to increase the mean fitness (population growth rate) to higher levels. This is the basis for Fisher's Fundamental Theorem of Natural Selection. It has wide implications for both ecology and evolutionary biology. For ecology, it means that you cannot predict the future size of the population very easily without knowing something about evolution.

Part III Exercises

1) Redo the analysis above, but this time assume that the initial distribution of phenotypes is skewed towards there being more clones with lower phenotypic values. Plot and discuss the results. You can you the code below to change the initial distribution.

```
Nt <- array(NA,c(time,N.clones))
Nt[1,] <- dnorm(phenos,mean=10,sd=30)
```

2) Now, assume that there is a trade-off between the intrinsic rate of increase, r, and the carrying capacity, K. This would be the case where a clone that is very good at growing under high resources (low density) conditions is very bad at growing at

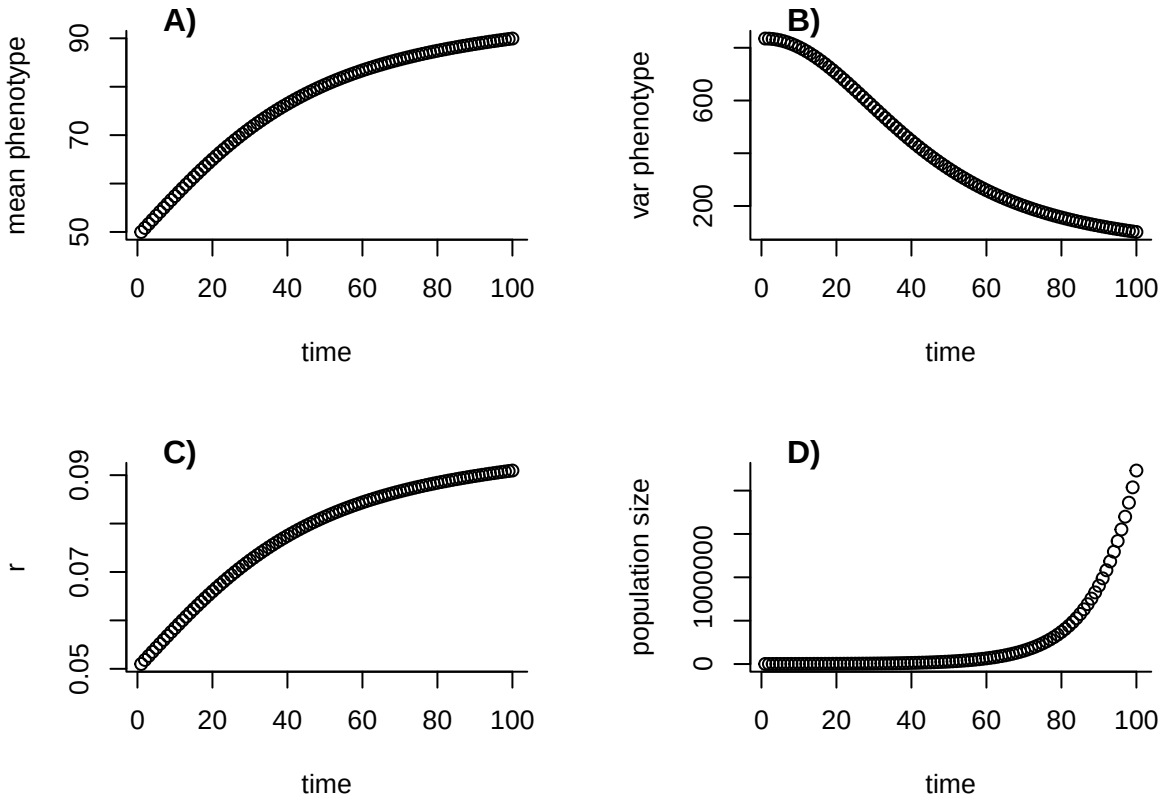


Figure 3.7: A) Mean phenotype, B) variance in the phenotype, C) mean population growth, and D) population size vs. time.

low resource (high density) conditions. Use the code below to run and plot the mean phenotype, mean r, the variance in the phenotypes, and population sizes through time. The code starts the phenotypic distribution with more of the good low density growing clones. Then contrast this with the results from question 1.

```
time <- 600
N.clones <- 100
phenos <- seq(0,100,length.out=N.clones)

r <- 0.01 + 0.001 * phenos

K <- 100 / (1 + 10*r)
# plot(phenos,r)
# plot(phenos,K)

Nt <- array(NA,c(time,N.clones))
Nt[1,] <- dnorm(phenos,mean=80,sd=60) #more higher r clones
growth.rate.t <- array(NA,c(N.clones))

for (t in 1:(time-1)){
  for (p in 1:N.clones){ #This extra loop calculates the growth rate
                         #for each phenotype
    growth.rate.t[p] <- exp(r[p] * (1 - sum(Nt[t,] / K[p])))
  }
  Nt[t+1,] <-  growth.rate.t * Nt[t,]
}

mn.pheno <- array(NA,c(time))
var.pheno <- array(NA,c(time))
mn.r <- array(NA,c(time))
for (t in 1:time){
  mn.pheno[t] <- sum( phenos * Nt[t,]/sum(Nt[t,]))
  var.pheno[t] <- sum( (phenos - mn.pheno[t])^2 * Nt[t,]/sum(Nt[t,]))

  mn.r[t] <- sum( r * Nt[t,]/sum(Nt[t,]))
}
par(mfrow=c(2,2),mar = c(4,5,3,2))
plot(c(1:time),mn.pheno,xlab='time',ylab='mean phenotype')
plot(c(1:time),var.pheno,xlab='time',ylab='var phenotype')
plot(c(1:time),mn.r,xlab='time',ylab='mean r')
plot(rowSums(Nt),xlab='time',ylab='population size')
```

Chapter 4

Age-structure and Fitness

Figure 4.1: A cow elephant with her calf. Elephant populations, like most populations, have an age structure. In these populations, the demographic rates (e.g. survival, reproduction, and growth depend on age.

4.1 Introduction

All of the natural selection we have talked about works well in simple situations. In particular, when the species is semelparous (reproduces once) and reproduction is synchronized. Examples of this include things like annual plants, Pacific salmon, etc. When this is the case, then the generation time is one year and individuals in the next year are the offspring of the parents in the previous year. This is only one type of general life history strategy.

Even in organisms that are semelparous, but are not synchronous (i.e. Pacific Salmon), the generation time is the mean age of the mothers, this complicates the interpretation of a generation because some reproduce in one year and others in other years.

Still, many organisms live over multiple years and reproduce each year. This gives rise to age-structure in the population. Humans are good examples of this, but so are most birds, mammals, fish, reptiles, etc. One important question then is "how is population growth and natural selection different in age-structured populations?" Stated another way, "does age-structure influence how populations grow and the strength of natural selection?"

4.2 Part I. Age-structured Population Growth

Because many populations are structured by age or some other character, we need some methods to allow us to predict population growth that takes this into account. We will begin here by creating a simple model of an age-structured population. We will then calculate lifetime reproductive success (or the per-generation rate of increase: R_0) and also the per time-step rate of population increase (λ).

Per-generation Rate of Increase (R_0)

We will be working with discrete ages and hence in discrete time. First we will be thinking about ages, x. See figure 1 for a refresher on the difference between age classes and ages.

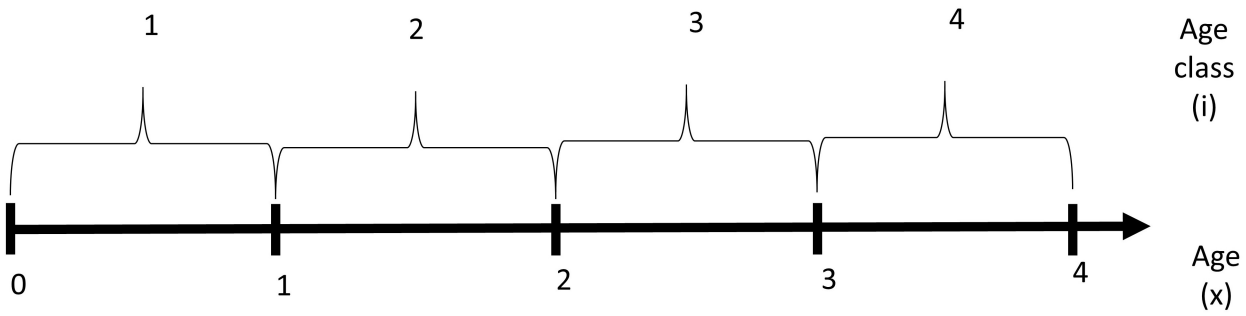


Figure 4.2: Age classes and ages.

First, let's set the number of ages and create a data frame with those ages

```
n.ages <- 4
x <- c(0:n.ages)
data <- as.data.frame(x)
```

Why include 0? 0 is the age at the moment of birth.

Fill in the data with age-specific birth rates $b(x)$ and survival $l(x)$.

```
data$bx <- c(0,0,4.1,2.5,0)
data$lx <- c(1,0.6,0.4,0.1,0)

data$sx <- NA
data$sx[c(1:n.ages)] <- data$lx[c(2:(n.ages+1))] / data$lx[c(1:n.ages)]
```

What type of survivorship curve does this represent? Plotting it helps to see what is going on.

```
plot(c(0:4),log(data$lx),ylab='Survivorship (ln (l(x)))',xlab='Age',type='b')
```

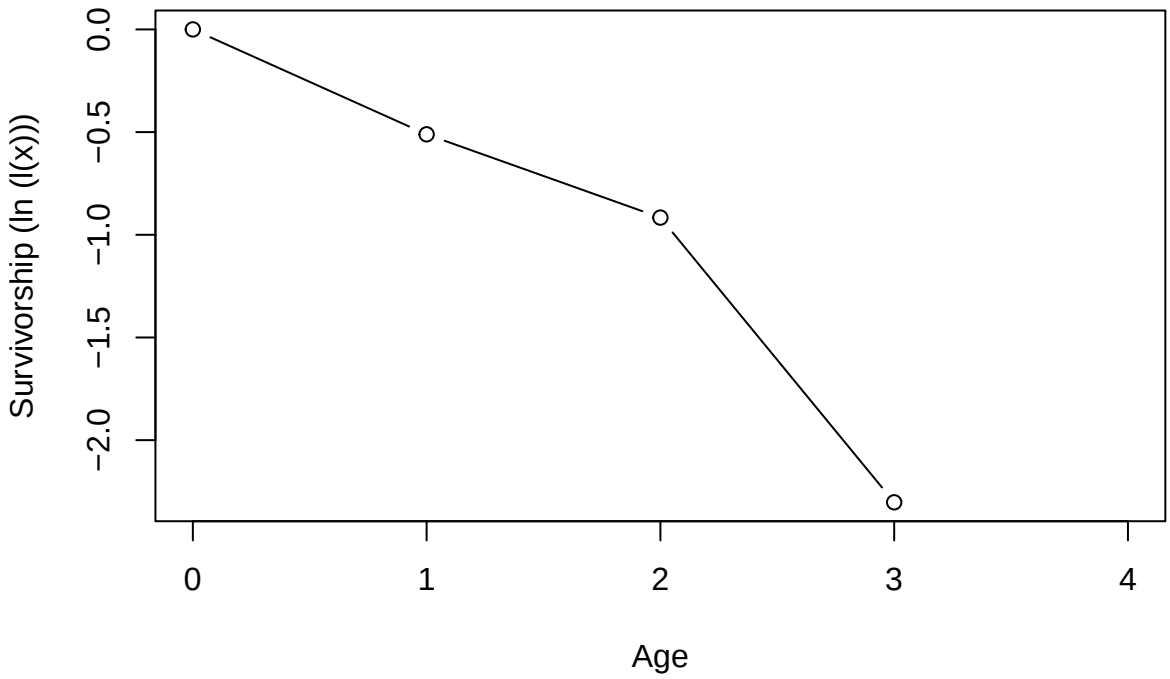


Figure 4.3: Type I survivorship curve.

Then calculate $l(x)b(x)$ and R_0

```
data$lxbx <- data$lx * data$bx
```

Then take a look at the table, which will look similar to what we had in class, just with different numbers.

```
data
```

```
##   x  bx  lx        sx lxbx
## 1 0 0.0 1.0 0.6000000 0.00
## 2 1 0.0 0.6 0.6666667 0.00
## 3 2 4.1 0.4 0.2500000 1.64
## 4 3 2.5 0.1 0.0000000 0.25
## 5 4 0.0 0.0        NA 0.00
```

Then R_0, the per-generation rate of population growth, is the sum of $l(x)b(x)$.

```
R.0 <- sum(data$lxbx)
R.0
```

```
## [1] 1.89
```

This is great, but what is the generation time? In other words, what time period does this population grow at this rate? Recall from lecture that the generation time, T, is defined as:

$$T = \sum x\lambda^{-x} l(x)b(x) \tag{4.1}$$

which gives the average age of mothers of a cohort of offspring. You can think of a cohort as the individuals of a population that are the same age.

But we need the per time step rate of growth, λ, to calculate it.

Per time-step population growth rate, λ

We might think that we could calculate λ from Euler's equation:

$$1 = \sum \lambda^{-x} l(x)b(x) \tag{4.2}$$

But this is not easily done. How can we get close to this without doing the math? One way is by plugging in values until Euler's equation is equal to 1. Here is one way to do this.

Set up some possible λ values

```
lams.pos <- seq(0.8,3,by=0.001)
```

Set up a vector to store the results

```
results <- array(NA,c(length(lams.pos)))
```

Then, using a loop, go through and calculate the left hand side of the equation using each of the possible λ values.

```
for (i in 1:length(lams.pos)){
  results[i] <- lams.pos[i]^-0 * data[1,'lxbx'] +
                lams.pos[i]^-1 * data[2,'lxbx'] +
                lams.pos[i]^-2 * data[3,'lxbx'] +
                lams.pos[i]^-3 * data[4,'lxbx'] +
                lams.pos[i]^-4 * data[5,'lxbx']
}
```

Then plot the results.

```
plot(lams.pos,results)
```

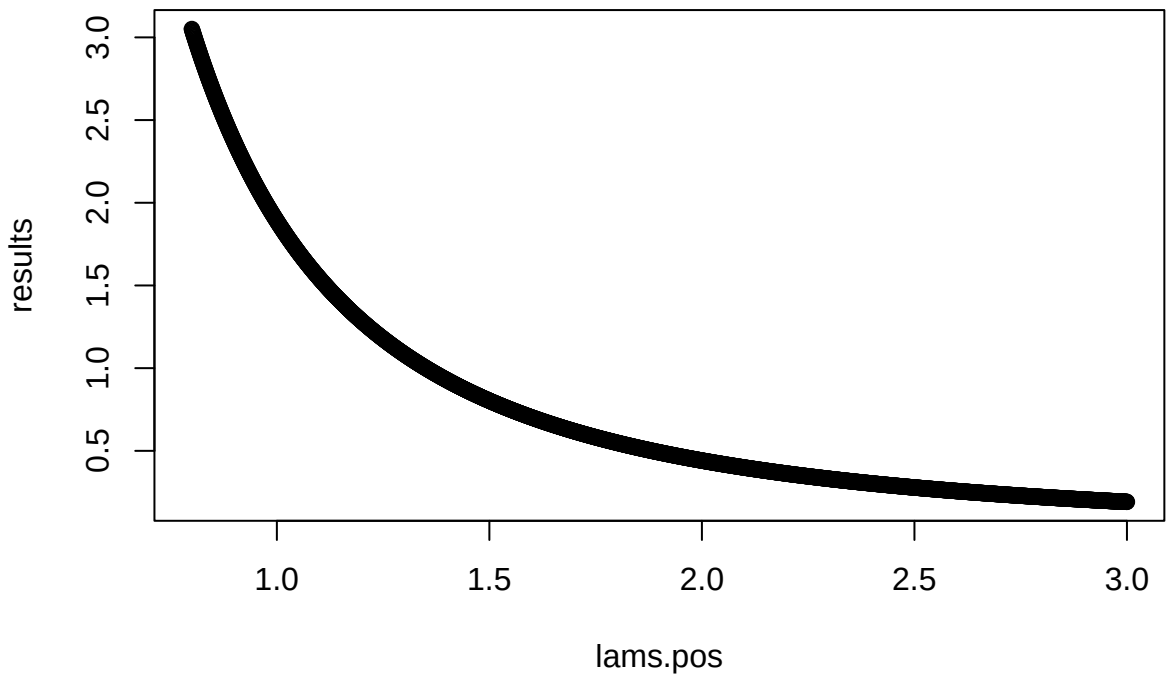


Figure 4.4: The solution of Euler's equation as a function of possible lambdas.

The value of "lams.pos" that makes "results" equal to 1 is λ. We cannot really see it well here, do it again, but limit the range of lambda to one that is nearer to the right one.

Set up some possible lambda values

```
lams.pos <- seq(1.25,1.45,by=0.001)
results <- array(NA,c(length(lams.pos)))
for (i in 1:length(lams.pos)){
  results[i] <- results[i] <- lams.pos[i]^-0 * data[1,'lxbx'] +
                              lams.pos[i]^-1 * data[2,'lxbx'] +
                              lams.pos[i]^-2 * data[3,'lxbx'] +
                              lams.pos[i]^-3 * data[4,'lxbx'] +
                              lams.pos[i]^-4 * data[5,'lxbx']
}
plot(lams.pos,results)
```

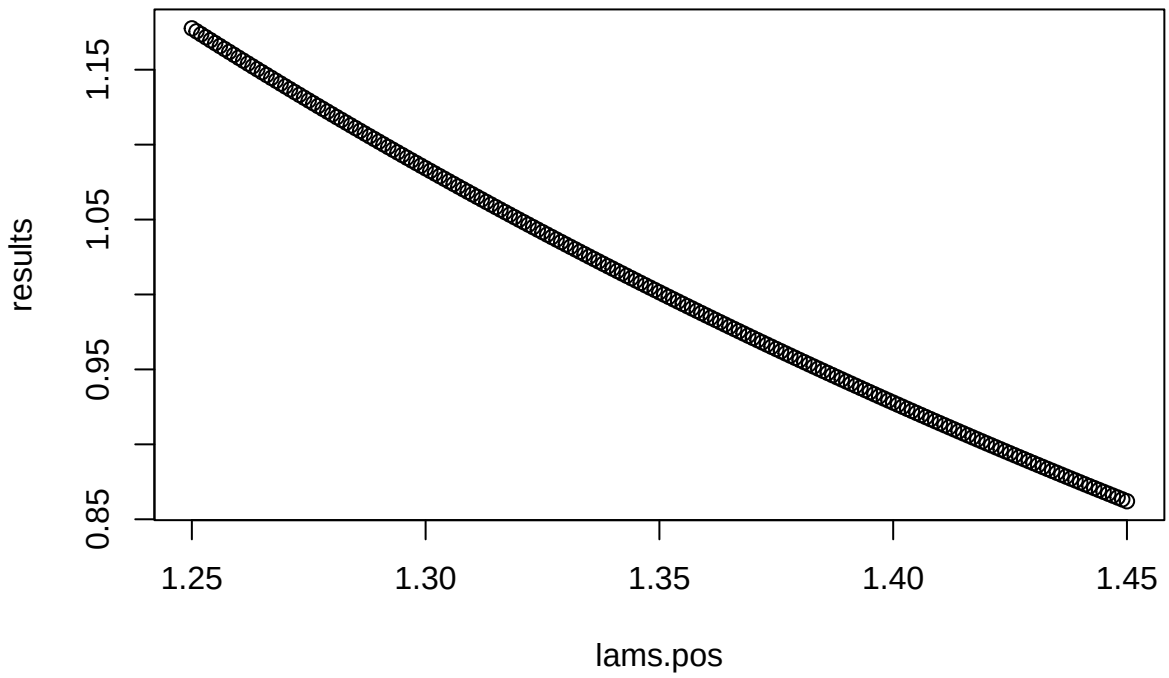


Figure 4.5: The solution of Euler's equation as a function of possible lambdas.

Looks to be about 1.35. Can we do better than that? Yes.

Plotting on the log scale makes the curve a straight line

```
plot(log(lams.pos),log(results))
```

We have already seen how to find the equation for a straight line using regression. In R, you can do this using this bit of code.

```
mod <- glm(log(results)~log(lams.pos))
summary(mod)
```

```
##
## Call:
## glm(formula = log(results) ~ log(lams.pos))
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -8.465e-05  -6.886e-05  -2.134e-05   5.793e-05   1.738e-04
```

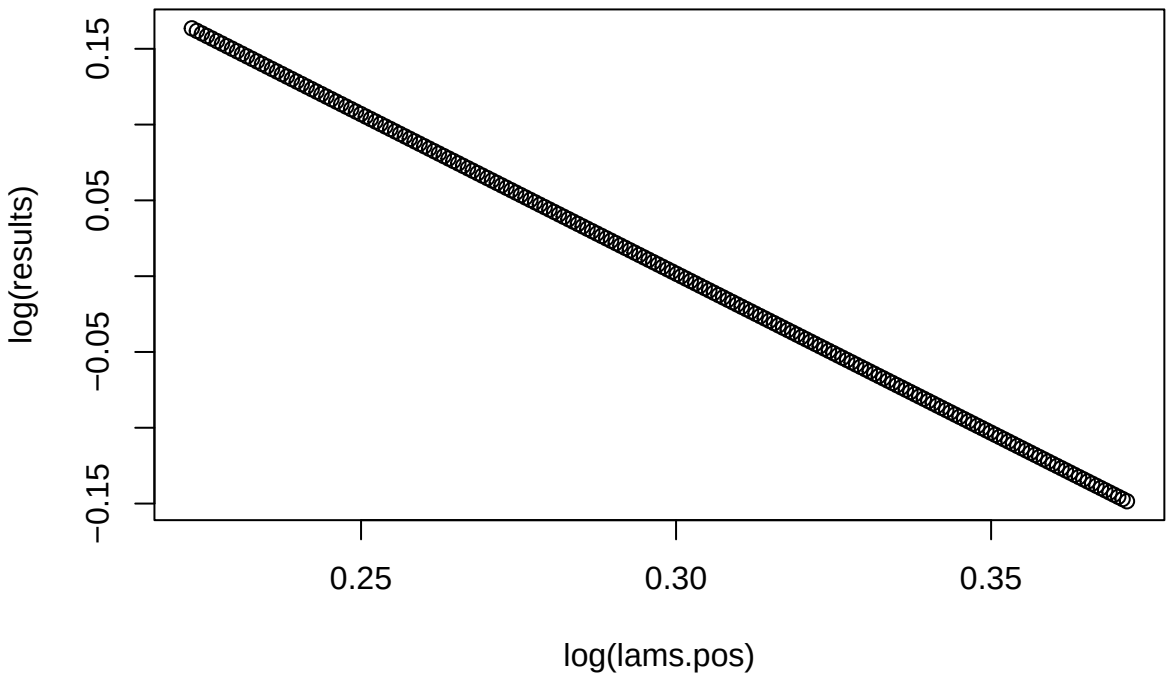


Figure 4.6: The natural log of the solution of Euler's equation as a function of possible natural logs of lambda.

```
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.6322827  0.0000377   16773   <2e-16 ***
## log(lams.pos) -2.1016874  0.0001247  -16852   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 5.789931e-09)
##
##     Null deviance: 1.6443e+00  on 200  degrees of freedom
## Residual deviance: 1.1522e-06  on 199  degrees of freedom
## AIC: -3238
##
## Number of Fisher Scoring iterations: 2
```

The numbers we are looking for are under "Coefficients". You can get them by typing:

```
coef(mod)["(Intercept)"]
```

```
## (Intercept)
##   0.6322827
```

and

```
coef(mod)["log(lams.pos)"]
```

```
## log(lams.pos)
##     -2.101687
```

which are the intercept and the slope of the line, respectively.

The equation we fit was the following:

$$log(result) = coef(mod)["(Intercept)"] + coef(mod)["log(lams.pos)"] * log(lam.pos) \quad (4.3)$$

What we want is the value of log(lam.pos) that makes the left hand side 0 ($ln(1) = 0$). So set the left hand side to 0 and solving for log(lam.pos) gives you the value of lambda.

```
lam <- exp( (0 - coef(mod)["(Intercept)"]) /  coef(mod)["log(lams.pos)"])
lam
```

```
## (Intercept)
##       1.351
```

Now that we have an estimate of lambda, we can find the generation time.

```
T <- 0 * lam^-0 * data[1,'lxbx'] +
     1 * lam^-1 * data[2,'lxbx'] +
     2 * lam^-2 * data[3,'lxbx'] +
     3 * lam^-3 * data[4,'lxbx'] +
     4 * lam^-4 * data[5,'lxbx']
T
```

```
## (Intercept)
##    2.101217
```

Our value for the generation time is a little more than two. This should make sense as the reproductive classes are 2 and 3 year olds. So the average age of mothers of a cohort should be between 2 and 3 years old too. Because we are defining the generation time as an average age of mothers of a cohort (newborns), the fact that this average is closer to 2 than 3 should tell you something about which ages contribute more to the future generation.

Geometric rate of increase, λ: brute force method 2

The example of how to calculate λ is not the only one. Let's go through another brute force method, then we will use the linear algebra to calculate it exactly. Before getting there, this method requires that we work with age classes instead of ages. In that way, we work with survival between the ages.

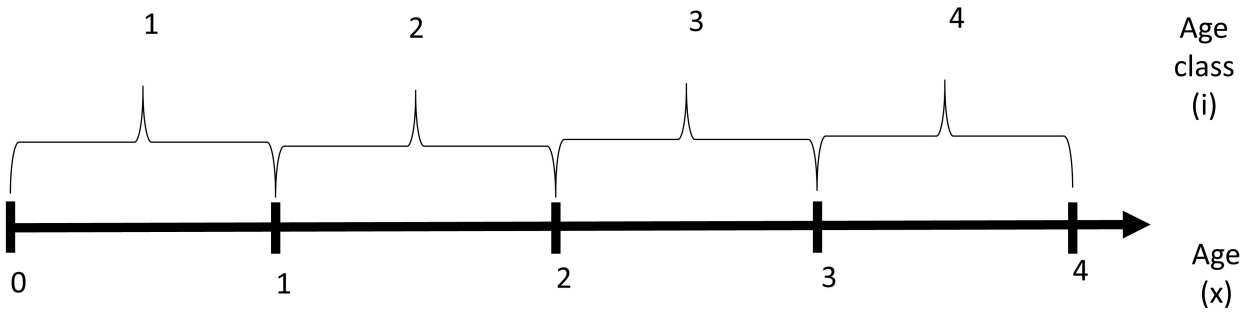


Figure 4.7: Age classes and ages.

First, we need to set up a matrix to store the age classes, the total population size, and λ. Let's also set the number of time steps to 100.

```
times <- seq(0,100, by=1)
n <- array(0,c(4,length(times))) # holds the number in each
                                 #age class at each time.
N <- array(NA,c(length(times)))  # holds the total population size at each time.
lam <- array(NA,c(length(times))) #holds the estimated lambda at each time.
```

Start off with a single individual in the first age class

```
n[1,1] <- 1
N[1] <- sum(n[,1])
```

Now we need the equations that tell us the demographic rates. Remember from lecture, these are:

$$n_1(t+1) = f_1 n_1(t) + f_2 n_2(t) + f_3 n_3(t) + f_4 n_4(t); \tag{4.4}$$

$$n_2(t+1) = s_1 n_1(t); \tag{4.5}$$

$$n_3(t+1) = s_2 n_2(t); \tag{4.6}$$

$$n_4(t+1) = s_3 n_3(t) \tag{4.7}$$

We can set these up in a loop, just as we have done before and iterate them through time. At each time step, we will get a λ value. It will change through time and eventually settle down as the population approaches the stable age distribution.

```
for (t in 1:c(length(times)-1)){

  n[1,t+1] <- data[which(data$x==1),'bx'] * data[which(data$x==1),'lx']/
                  data[which(data$x==0),'lx'] * n[1,t] +
              data[which(data$x==2),'bx'] * data[which(data$x==2),'lx']/
                  data[which(data$x==1),'lx'] * n[2,t] +
              data[which(data$x==3),'bx'] * data[which(data$x==3),'lx']/
                  data[which(data$x==2),'lx'] * n[3,t]
  n[2,t+1] <- data[which(data$x==1),'lx']/data[which(data$x==0),'lx'] * n[1,t]
  n[3,t+1] <- data[which(data$x==2),'lx']/data[which(data$x==1),'lx'] * n[2,t]

  n[4,t+1] <- data[which(data$x==3),'lx']/data[which(data$x==2),'lx'] * n[3,t]

  N[t+1] <- sum(n[,t+1])

  lam[t] <- N[t+1]/N[t]

}
```

And plot it

```
plot(times,lam,ylab=expression(lambda))
```

Notice how it settles down through time. The initial jumps are due to the population not be in the stable age distribution. Eventually, it reaches a stable age distribution. We can see what it settles down to with:

```
lam[length(times)-1]
```

```
## [1] 1.350953
```

We can also plot the population sizes through time with:

```
plot(times,N)
```

It seems ridiculous that the population grows so much. However, we are not trying to really project population size through time. Rather we are just trying to find the population growth rate. With this life history, it takes the population a long time to reach the stable age distribution.

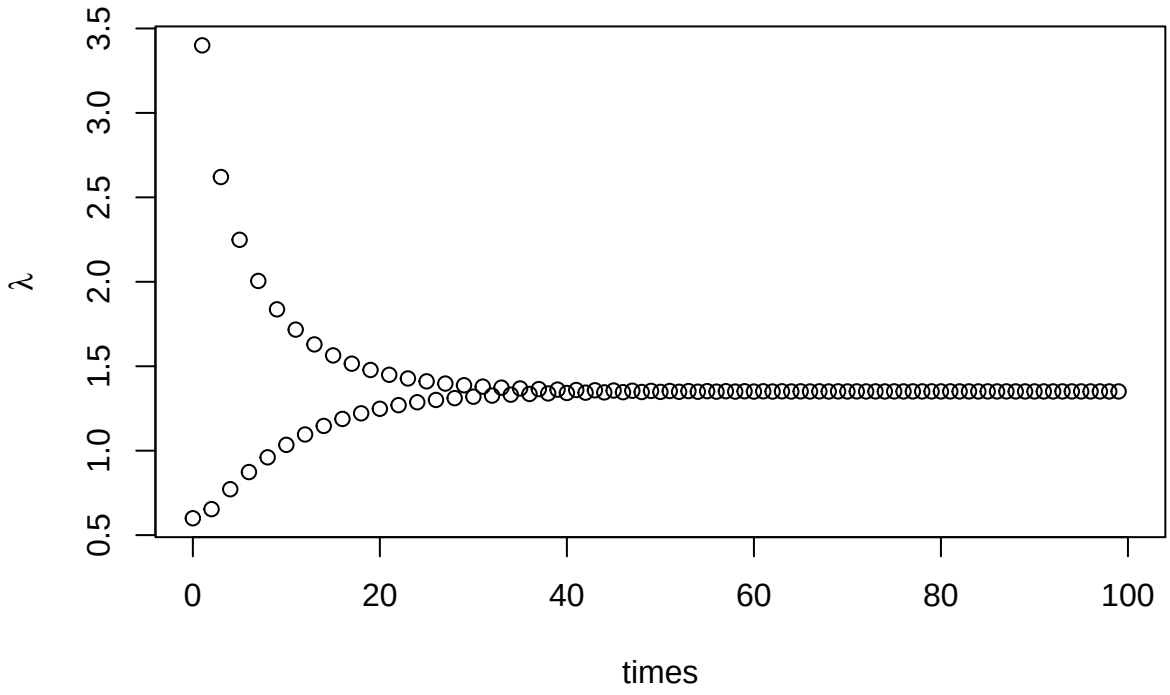


Figure 4.8: Lambda versus time.

Geometric rate of increase, λ: Analytical method

It turns out that there is a solution to Euler's equation. It is not a unique solution, meaning there is more than one, but we need to use linear algebra to find it. Will not learn the details of how to do this as it is not pertinent to the course. We will let the computer calculate it for us.

First, we need to make a projection matrix. A simple projection matrix has rows that are equivalent to the equations for population size at time t + 1. The entries in the rows are the parameters in the equations. Notice there are no variables in the matrix, only parameters.

$$A = \begin{bmatrix} b_1 s_0 & b_2 s_1 & b_3 s_3 & b_4 s_4 \\ s_0 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & 0 & s_3 & 0 \end{bmatrix},$$

```
A <- array(0,c(4,4))

A[1,] <- c(data[which(data$x==1),'bx'] * data[which(data$x==1),'lx']
              /data[which(data$x==0),'lx'],
```

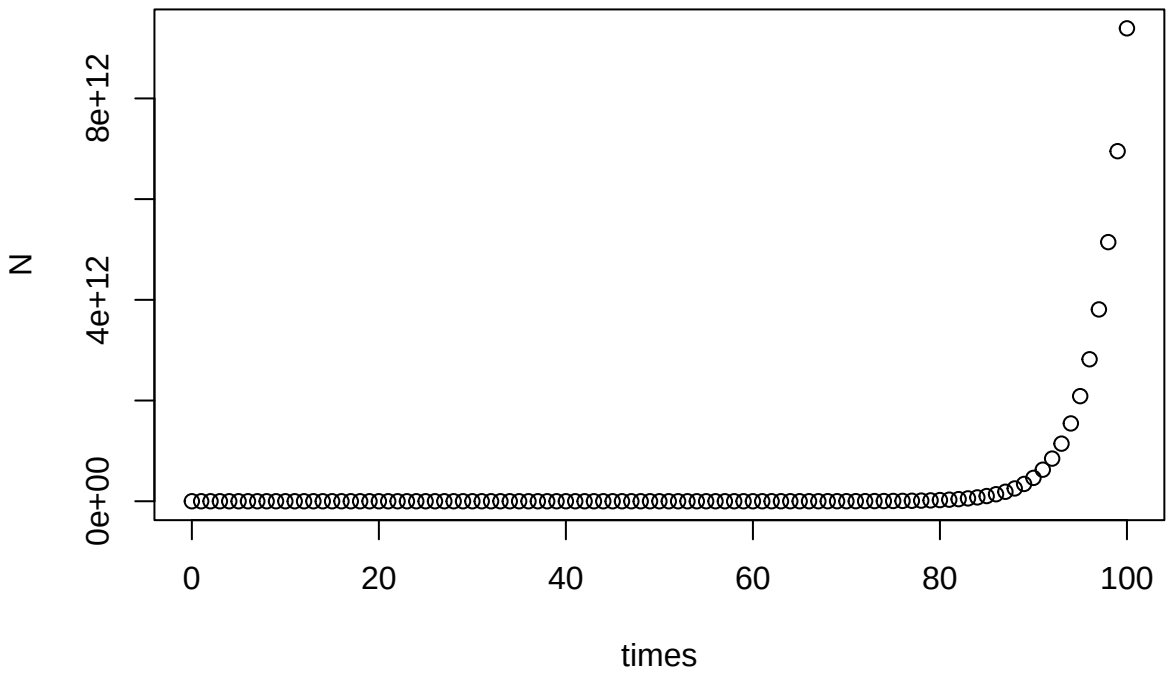


Figure 4.9: Population size as a function of time.

```
           data[which(data$x==2),'bx'] * data[which(data$x==2),'lx']
              /data[which(data$x==1),'lx'],
           data[which(data$x==3),'bx'] * data[which(data$x==3),'lx']
              /data[which(data$x==2),'lx'],0)
A[2,1] <- data[which(data$x==1),'lx']/data[which(data$x==0),'lx']
A[3,2] <- data[which(data$x==2),'lx']/data[which(data$x==1),'lx']
A[4,3] <- data[which(data$x==3),'lx']/data[which(data$x==2),'lx']
```

Print it out using:

```
A
```

```
##      [,1]      [,2]  [,3] [,4]
## [1,]  0.0 2.7333333 0.625    0
## [2,]  0.6 0.0000000 0.000    0
## [3,]  0.0 0.6666667 0.000    0
## [4,]  0.0 0.0000000 0.250    0
```

Each row in the matrix is an equation that says how many individuals will be in that age class in the next time point. Each column in each row says from which age class they are coming from.

To get the full set of equations, we then define the vector that holds the sizes of each age class at time t as:

$$n(t) = \begin{bmatrix} n_1(t) \\ n_2(t) \\ n_3(t) \\ n_4(t) \end{bmatrix}.$$

The projection matrix premultiplied by the population size vector gives the population size in each class in the next time step:

$$\begin{bmatrix} n_1(t+1) \\ n_2(t+1) \\ n_3(t+1) \\ n_4(t+1) \end{bmatrix} = \begin{bmatrix} b_1 s_0 & b_2 s_1 & b_3 s_3 & b_4 s_4 \\ s_0 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 \\ 0 & 0 & s_3 & 0 \end{bmatrix} \begin{bmatrix} n_1(t) \\ n_2(t) \\ n_3(t) \\ n_4(t) \end{bmatrix}.$$

Matrices are powerful ways to represent several simultaneous equations. Under many circumstances, we can use them to solve the equations or to calculate quantities such as λ. For example, from the matrix, λ is the leading (first) eigenvalue of the matrix, A. You can get this, simply by typing:

This is lambda.

```
lambda <- Re(eigen(A)$values[1])
lambda
```

```
## [1] 1.350946
```

Unlike our previous approximations, this value of λ is exact. Because in ecology and evolution we are often interested in several components of populations and communities changing simultaneously and as a function of each other, it is not surprising that they are used extensively in Ecology, Evolution, and Conservation Science.

4.3 Part II. Beyond Population Growth: Stable Age Distributions and Reproductive Value

The population growth rate is not the only quantity that is important in age structured populations. Both the stable age distribution and reproductive value influence the action of natural selection in age structured populations. Recall that the stable age distribution is the distribution of ages in the population where all age classes grow at the same rate, λ. We can get these values from both the life table methods and the matrix methods.

Stable Age Distribution and Reproductive Value From the Life Table

We will deal with the stable age distribution first. The stable age distribution from the life table is:

$$W(x) = \frac{\lambda^{-x} l(x)}{\sum_{y=x}^{d} \lambda^{-y} l(y)} \tag{4.8}$$

```
denom <-0 #create a placeholder for the denomenator that has a zero in it.
for (x in 1:n.ages){
  denom <- denom + ( lambda^(-x) * data[(x),'lx'] )
}

data$stable.age <- NA
data$stable.age[c(1:4)] <- (lambda^(-c(1:4)) * data[c(1:4),'lx']) / denom

data$stable.age
```

```
## [1] 0.58690149 0.26066242 0.12863203 0.02380406         NA
```

You can compare this to what we got by iterating the population through time using:

```
n[,t]/sum(n[,t])
```

```
## [1] 0.58689922 0.26066513 0.12863130 0.02380436
```

These should be approximately equal. The one from the iteration would be closer if we allowed it to run longer.

Next is the reproductive value. Recall from lecture that reproductive value is the contribution of individuals from a given age group to the ancestry of the next generation. One definition of it is:

$$V(x) = \frac{\lambda^{x}}{l(x)} \sum_{y=x}^{d} \lambda^{-y} b(y) l(y) \tag{4.9}$$

We can calculate this from the data in the life table using:

```
data$rv <- NA
data$rv[1] <- (lambda^1 / data[(1),'lx'] * (lambda^-1 * data[c(1+1),'lxbx']) ) +
              (lambda^1 / data[(1),'lx'] * (lambda^-2 * data[c(2+1),'lxbx']) ) +
              (lambda^1 / data[(1),'lx'] * (lambda^-3 * data[c(3+1),'lxbx']) ) +
              (lambda^1 / data[(1),'lx'] * (lambda^-4 * data[c(4+1),'lxbx']) )

data$rv[2] <- (lambda^2 / data[(2),'lx'] * (lambda^-2 * data[c(2+1),'lxbx']) ) +
              (lambda^2 / data[(2),'lx'] * (lambda^-3 * data[c(3+1),'lxbx']) ) +
              (lambda^2 / data[(2),'lx'] * (lambda^-4 * data[c(4+1),'lxbx']) )

data$rv[3] <- (lambda^3 / data[(3),'lx'] * (lambda^-3 * data[c(3+1),'lxbx']) ) +
              (lambda^3 / data[(3),'lx'] * (lambda^-4 * data[c(4+1),'lxbx']) )

data$rv[4] <- (lambda^4 / data[(4),'lx'] * (lambda^-4 * data[c(4+1),'lxbx']) )
```

Another way is to use two loops.

```
rv <- array(0,c(n.ages))
for (x in 1:n.ages){
  for (y in x:n.ages){
    rv[x] <- rv[x] + ( lambda^(x) / data[(x),'lx'] *
                       (lambda^-y * data[c(y+1),'lxbx']) )
  }
}
```

Reproductive value is often scaled so that RV for the first age is 1.

```
rv <- rv/rv[1]

data$rv[c(1:4)] <- rv
```

We can plot the stable age distribution and reproductive value side by side using:

```
par(mfrow=c(1,2),mar = c(5,6,1,2))

plot(c(0:4),data$stable.age,ylab='frequency',xlab='age (x)')
title(main='A)',adj = 0.1, line = 0)
plot(c(0:4),data$rv,ylab='Reproductive value',xlab='age (x)')
title(main='B)',adj = 0.1, line = 0)
```

Reproductive value is important because it, in part, defines how sensitive population growth is to changes in things like survival and reproduction at each age class. For example, for survival,

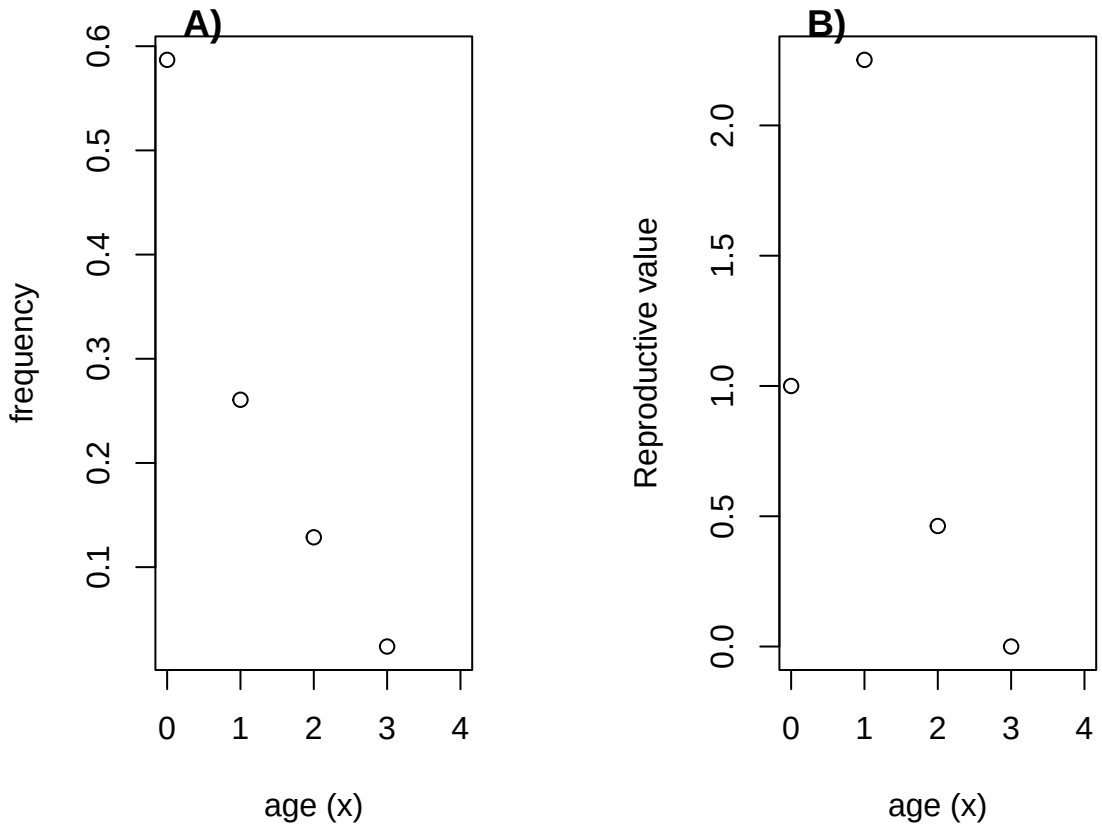


Figure 4.10: A) Stable age distribution and B) reproductive value as a function of age.

$$\frac{dr}{dln(s(x))} = \frac{l(x)}{\lambda^x}\frac{V(x)}{T} \tag{4.10}$$

```
dr.dlogS <- array(0,c(4))
for (x in 1:4){
dr.dlogS[x] <- (data$rv[x] * lambda^(-x) * data[x,'lx']) / T
}

plot(c(1:4),dr.dlogS,ylab=expression(dr/dln(s(x))),xlab=c("age(x)"))
```

This sensitivity, and others like it, are used extensively in ecology and evolution. They are important because they tell us changing which age-specific will change the population growth rate the most. Or which it is most sensitive to. This can do things like inform management decisions and also tells us where in the life cycle evolutionary changes will have the biggest impact. For our example, increasing the survival rate of the two youngest age classes will have the largest impact on the population growth rate.

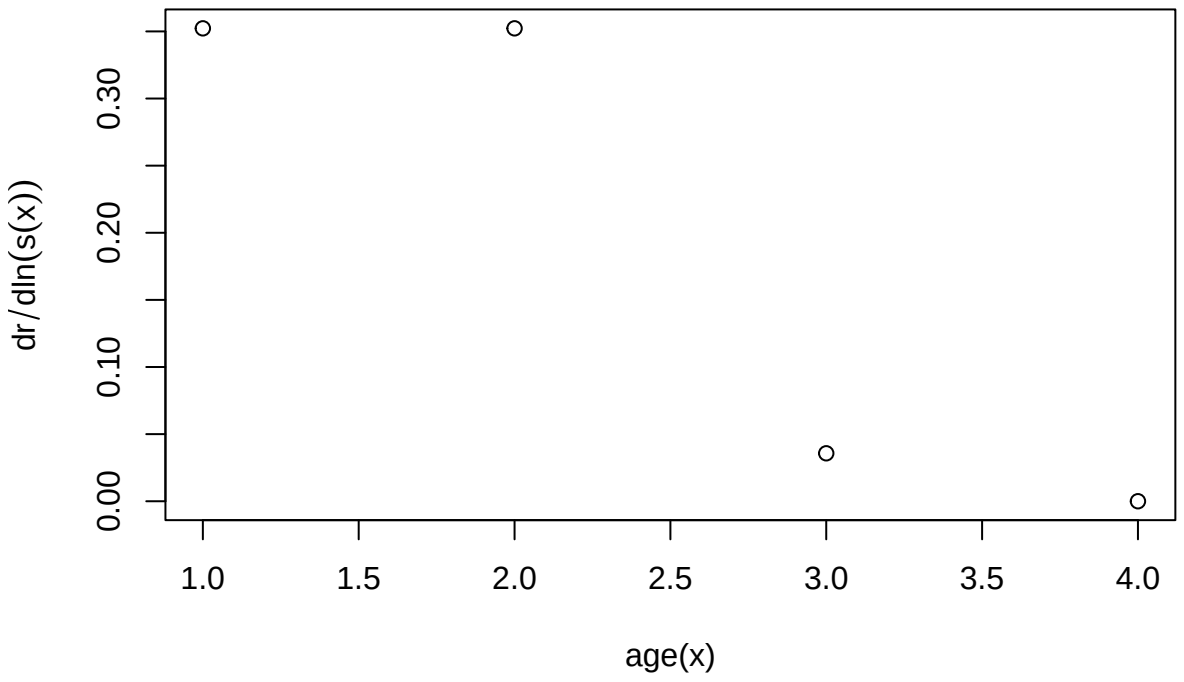


Figure 4.11: The sensitivity of fitness (ln (lambda)) to survival at age (x)

Stable Age Distribution and Reproductive Value From the Matrix Model

Calculations of the stable age distribution and reproductive value from the matrix model are very straightforward. Again, this relies upon an eigen analysis of the matrix A. The stable age distribution is calculated as the right eigenvector associated with the leading eigenvalue. That is a mouth full, but all you really need at this point is:

```
w <- Re(eigen(A)$vectors[,1]) / sum(Re(eigen(A)$vectors[,1]) )
w
```

```
## [1] 0.58690149 0.26066242 0.12863203 0.02380406
```

Similarly, the reproductive value is the left eigen vector that is associated with the leading eigen value, λ.

```
v <- Re(eigen(t(A))$vectors)[,1]
```

Reproductive value is then scaled by sum of the products of the left and right eigenvectors using.

```
v <- v / as.vector(t(v)%*%w)
```

The "%*%" in this equation means matrix multiplication. The "t" means to take the transpose. This makes the sum of the products of the reproductive value and the stable age distribution equal to 1.

```
t(v)%*%w
```

```
##      [,1]
## [1,]    1
```

We can plot the stable age distribution and reproductive value side by side using:

```
par(mfrow=c(1,2),mar = c(5,6,1,2))

plot(c(1:4),w,ylab='frequency',xlab='age class (i)')
title(main='A)',adj = 0.1, line = 0)
plot(c(1:4),v,ylab='Reproductive value',xlab='age class (i)')
title(main='B)',adj = 0.1, line = 0)
```

As we saw in lecture, reproductive value increases up to the age at first reproduction and then declines. Meaning the contribution of the age classes to future generations peaks at the age of first reproduction and then declines throughout life. This means that the strength of selection is often weak early in life, increases up to the age of first reproduction, and then declines as an organism ages. Note that this initially seems at odds with what our sensitivities told us. That is the youngest groups would change the population growth rate the most. The difference however, is that while the sensitivities tell us how much changing one thing will change something else (e.g. survival and population growth), the reproductive value tells us what is currently the most important ages to population growth.

You can compare these values to those that you got from the life table approach. Note that the reproductive value will differ because the values are scaled differently. To make the one from the matrix approach comparable to the values from the life table approach, divide all the values by the first one using:

```
v/v[1]
```

```
## [1] 1.0000000 2.2515769 0.4626387 0.0000000
```

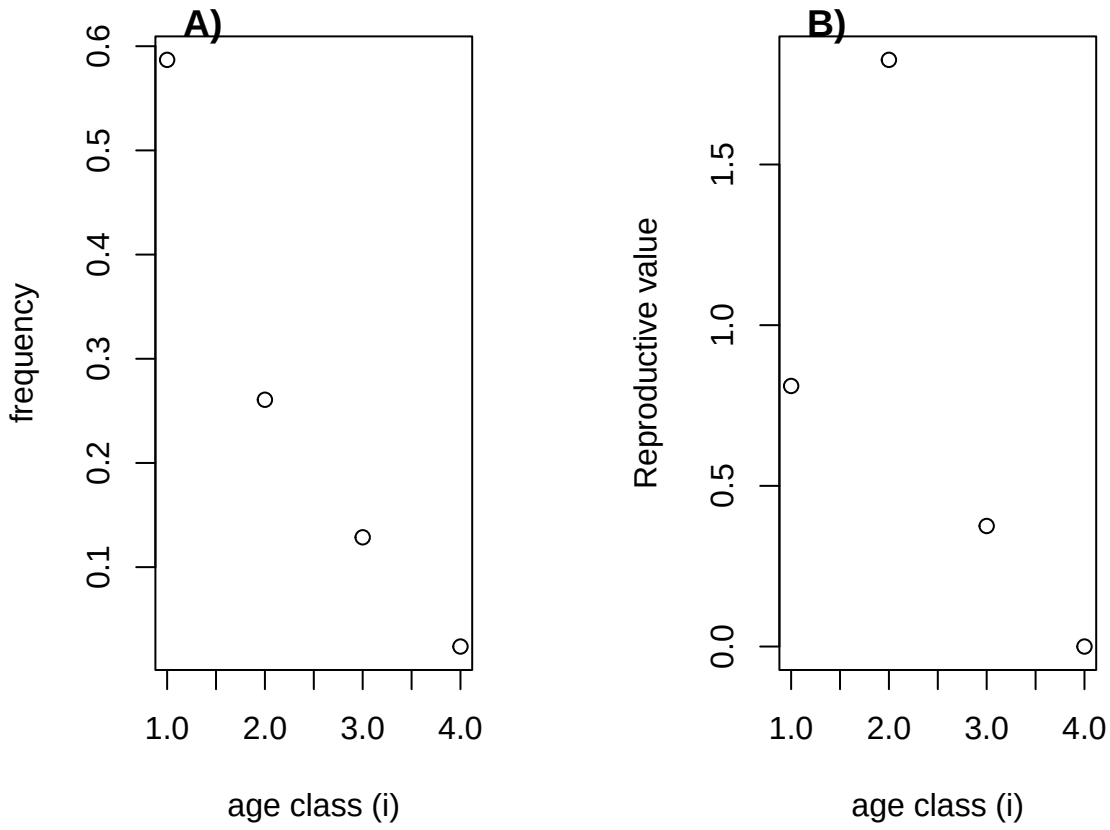


Figure 4.12: A) Stable age distribution and B) reproductive value as a function of age class.

```
data$rv
```

```
## [1] 1.0000000 2.2515769 0.4626387 0.0000000        NA
```

What does it mean to be a left and right eigenvector of A? In the simplest terms, the left and right eigenvectors are those that A is pre- and post- multiplied by to get λ. In other words:

$$\lambda = v^T A w \tag{4.11}$$

You can try this using:

```
t(v)%*%A%*%w
```

```
##          [,1]
## [1,] 1.350946
```

should be equal to λ.

```
Re(eigen(A)$values)[1]
```

```
## [1] 1.350946
```

Part II Exercises

1) In our examples, we have been working with an organism that has a Type I like survivorship curve. Go back and change the survivorship curve so that it is more like a type III survivorship curve. Use either the life table or matrix approach to then recalculate the λ, R_0, the stable age distribution, reproductive value, and the strength of selection at each age. Compare these to the original analyses. What changes do you observe? Pay particular attention to the sensitivities, reproductive value, and the stable age distribution.

Chapter 5

The Analysis of Marked Individuals

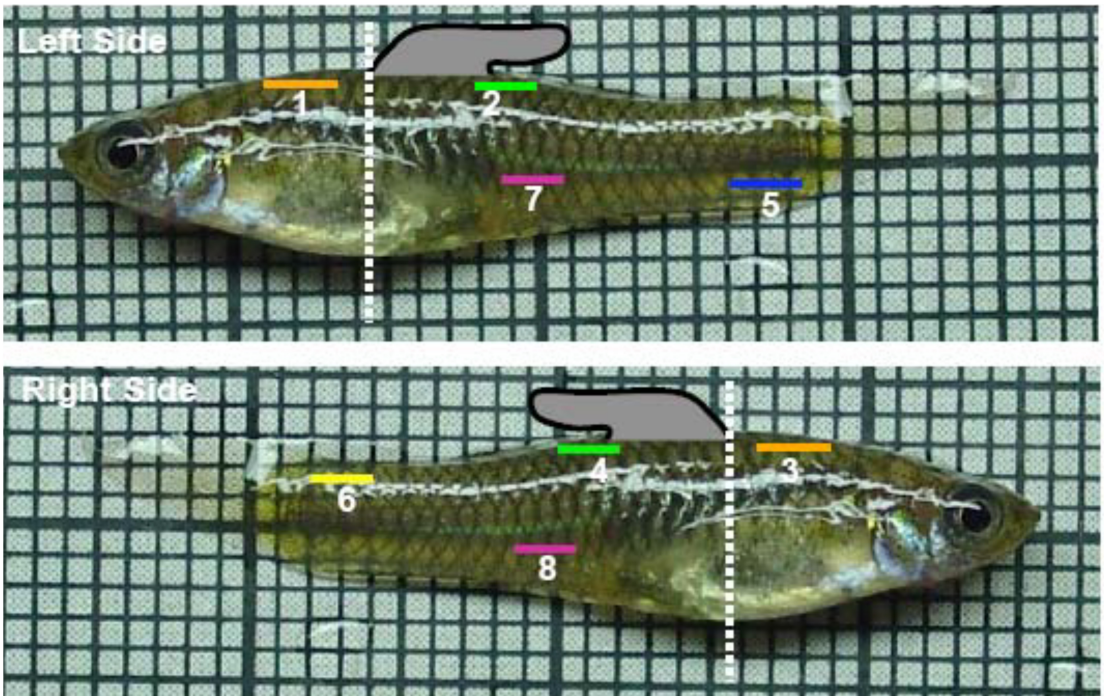


Figure 5.1: Positions on guppies used to mark the fish with colored elastomer implants

5.1 Introduction

Testing hypotheses in evolutionary ecology often requires the measurement of survival, growth, reproduction, and accurate estimates of population size or density. We are often interested in these quantities for the population as a whole, but also of subgroups that may represent individuals in different treatment groups. In order to be able to ask how experimental treatments affect individuals, we need a way to be able to identify each individual

through time. One way to do this is to mark individuals with unique identifiable tags or marks.

In this lab we will be working through how to conduct a mark-recapture analyses and along the way learn why it is important. *For reading, you should read the The Mark Book Chapter 1*, which is up on Canvas and included as an appendix to this lab. This is an excellent reference for these types of studies. Chapter 1 gives some background and justification. It has a fair amount of math in it. Interested readers can go through the math, but I am not expecting you to understand it all. The main concepts that you should be familiar with by the end of the chapter are: Return rates, how survival and probability of capture can be separated, and Maximum Likelihood, particularly how it is used to estimate the demographic quantities.

You will also need to download and install Program Mark, which can be found here: http://www.phidot.org/software/mark/downloads/index.html. It works well on PC's. There are instructions on the page for installing to various operating systems.

5.2 Part I: A simple mark-recapture analysis–the Cormack-Jolly-Seber (CJS) model for survival and probability of capture

Clear R's brain and set the working directory.

```
rm(list=ls())
setwd("~/Evolutionary Ecology/Lab 4")
```

```
survival <- read.csv(file = "./Data/survival.csv",colClasses = "character")
```

Then take a look at the file.

```
head(survival)
```

```
##                                                ch stream sex
## 1 1111111111100000000000000000000000000000000     LL   F
## 2 1000000000000000000000000000000000000000000     LL   F
## 3 1000000000000000000000000000000000000000000     LL   F
## 4 1111111110001011111111101011000000000000000     LL   F
## 5 1010101111100000000000000000000000000000000     LL   F
## 6 1111111111111111000001000000000000000000000     LL   F
```

This has the capture history for all the individuals in the population during the first 42 months.

You will need to install a package called "RMark", which let's us run program Mark using R. You can use this code to do this. Only run it one time.

```
install.packages("RMark")
```

Then you can load the package using this.

```
require(RMark)
```

This runs the model and shows the results.

```
dp <- process.data(survival,model="CJS",groups=('sex'))
ddl <- make.design.data(dp)

Phi <- list(formula=~1)
p <- list(formula=~1)

surv.mod <- mark(dp,ddl,model.parameters=list(Phi=Phi, p=p),
                 invisible=TRUE,silent=TRUE,output=FALSE)

surv.mod$results$real
```

```
##                   estimate        se       lcl       ucl fixed note
## Phi gF c1 a0 t1 0.8693292 0.0020422 0.8652741 0.8732800
## p gF c1 a1 t2   0.9023848 0.0020909 0.8982092 0.9064069
```

This tells us that our estimate for survival is 86.9% (Phi:ϕ) each month and our probability of capturing an individual (Rho:ρ) is 90.2%. It also gives us some standard errors and confidence intervals, which are pretty small. This is not surprising as there is a lot of data to work with.

One powerful thing about being able to do such analyses is that we can test hypotheses in the context of these types of analyses. Let's say we wanted to know if males and females have different survival probabilities. Then we can run a model that has sex as a factor. Let's do this first with both the survival and the probability of capture.

```
Phi <- list(formula=~sex)
p <- list(formula=~sex)

surv.mod <- mark(dp,ddl,model.parameters=list(Phi=Phi, p=p),
                 invisible=TRUE,silent=TRUE,output=FALSE)

surv.mod$results$real
```

```
##                   estimate        se       lcl       ucl fixed note
## Phi gF c1 a0 t1 0.9086900 0.0021138 0.9044616 0.9127493
## Phi gM c1 a0 t1 0.7845322 0.0044259 0.7757309 0.7930803
## p gF c1 a1 t2   0.9050433 0.0023853 0.9002640 0.9096166
## p gM c1 a1 t2   0.8968443 0.0042347 0.8882439 0.9048537
```

Now there are two lines of results for ϕ and two lines for ρ. The first line is for females and the second line is for males. The 'g' before the F or M is short for group. These results show that the survival probability for females is much higher than for males and the probability of capture if alive is about the same. We can plot these results using the estimates and the confidence intervals.

To do so, you must first install the package "Hmisc".

```
install.packages("Hmisc") #Only run this one time, then not again.
```

Then you can load the package using this.

```
require("Hmisc")
```

Then make the plots.

```
toplot <- surv.mod$results$real[c(1,2),c('estimate','lcl','ucl')]

errbar(x=c(1,2),y=toplot$estimate,yminus=toplot$lcl,yplus=toplot$ucl,
       xlab='Sex',ylab='Survival',xaxt='n',xlim=c(0.5,2.5),ylim=c(0.7,1))
axis(1,at=c(1,2),c('F','M'))
```

We can also plot the capture probabilities.

```
toplot <- surv.mod$results$real[c(3,4),c('estimate','lcl','ucl')]

errbar(x=c(1,2),y=toplot$estimate,yminus=toplot$lcl,yplus=toplot$ucl,
       xlab='Sex',ylab='Probability of capture',xaxt='n',
       xlim=c(0.5,2.5),ylim=c(0.85,0.95))
axis(1,at=c(1,2),c('F','M'))
```

We also want to have a formal test of the differences between the groups (sex). In essence we want to ask whether a model with sex as a factor fits the data better than one without it. For this we need to run two models. One with sex and one without sex.

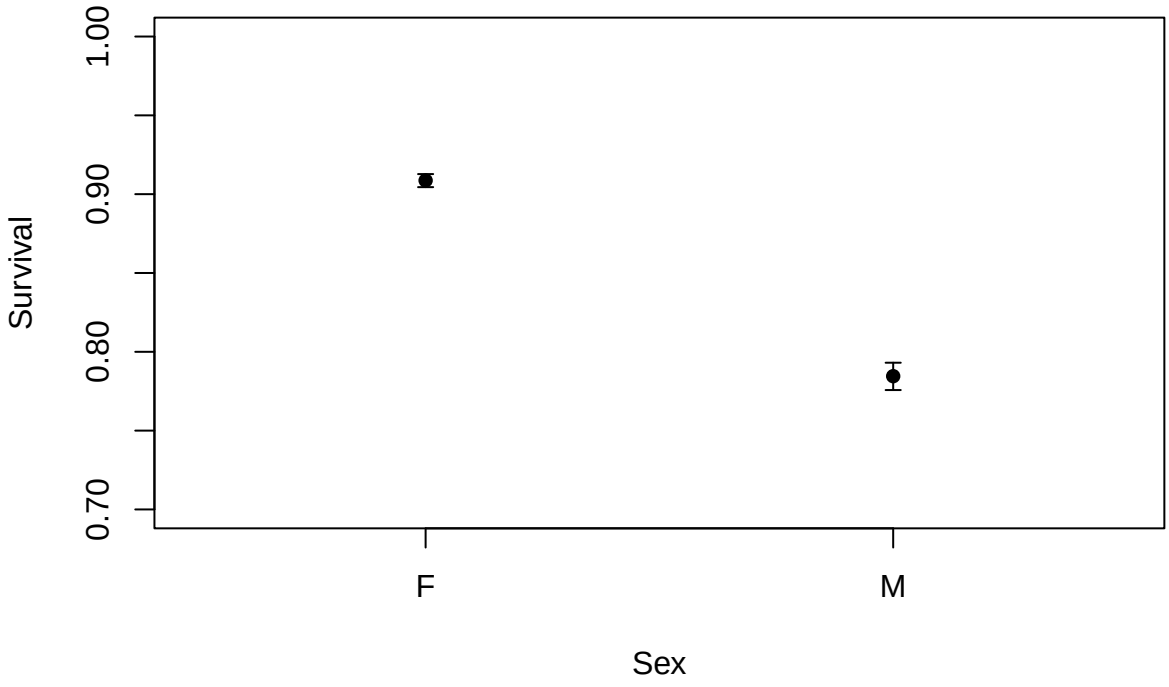


Figure 5.2: Survival vs. sex from the mark-recapture analysis.

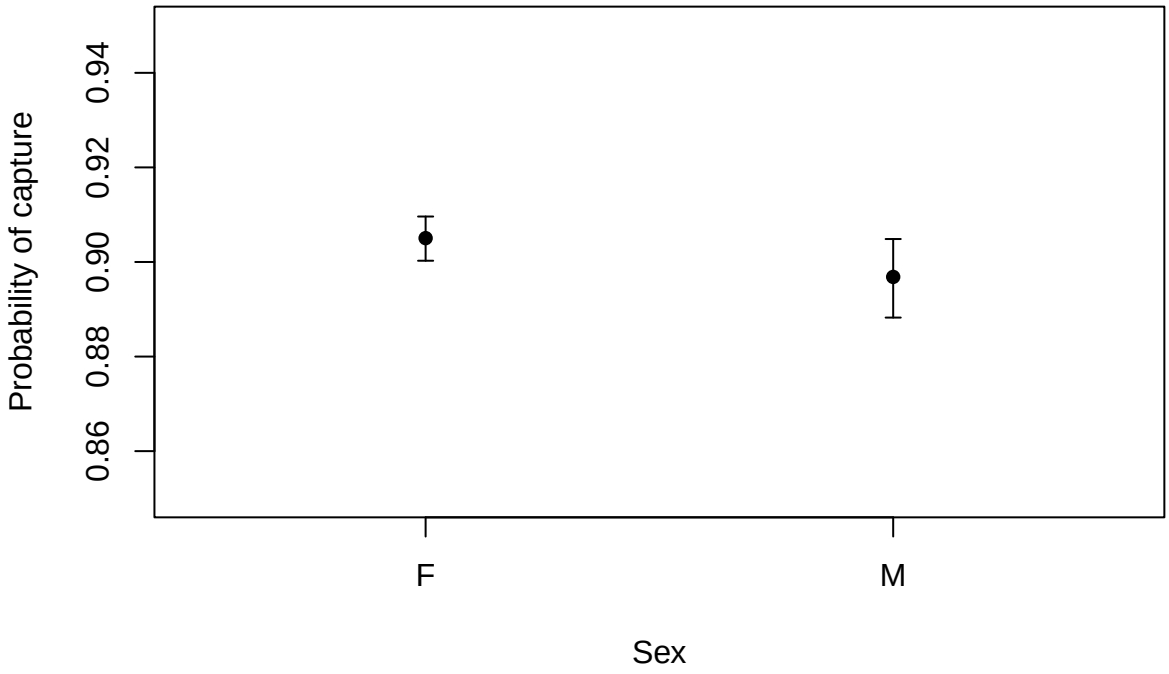


Figure 5.3: Probability of capture vs. sex from the mark-recapture analysis.

```
dp <- process.data(survival,model="CJS",groups=('sex'))
ddl <- make.design.data(dp)

#first with sex
Phi <- list(formula=~sex)
p <- list(formula=~sex)

surv.mod.sex <- mark(dp,ddl,model.parameters=list(Phi=Phi, p=p),
                     invisible=TRUE,silent=TRUE,output=FALSE)

#then without sex
Phi <- list(formula=~1)
p <- list(formula=~sex)

surv.mod <- mark(dp,ddl,model.parameters=list(Phi=Phi, p=p),
                 invisible=TRUE,silent=TRUE,output=FALSE)
```

Now each model has its own likelihood. Remember the likelihood tells us something about how likely the model is to be true. The higher the likelihood, the more probable the model is a good model. Likelihoods that are calculated on different models, but with the same data, can be compared to each other to ask whether the better fit model fits significantly better. If so, the the factor that is in one and not the other, sex in this case, can be said to explain a significant amount of variation in the model.

The log-likelihoods for each model can be found using the following code. The test is actually based on the - 2 x natural log of the likelihood for various reasons and R returns these to us as part of the model output.

```
surv.mod.sex$results$lnl
```

```
## [1] 33723.76
```

```
surv.mod$results$lnl
```

```
## [1] 34472.39
```

To test whether they are different from each other, we subtract the smaller from the larger one and save the results as 'test'.

```
test <- surv.mod$results$lnl - surv.mod.sex$results$lnl
test
```

```
## [1] 748.636
```

Test, is now our test statistic. If this value is small then there is little difference in how much variation in the data is explained by sex. If it is large, then sex explains a significant amount of the variation. How large is large enough? The test that is used here is the χ^2-test (pronounced chi-square). The degrees of freedom for this test is 1. We can get a p-value for the test using the following code.

```
p <- pchisq(test,df=1,lower.tail=FALSE)
p
```

```
## [1] 7.943001e-165
```

This says that our p-value is much much less than 0.05, so we can be fairly confident that there is a significant effect of sex on the survival probabilities. Sorry males.

Including other factors is certainly possible. For example, survival may not be constant across different time periods. We could just ask about the effect of time (month of capture) on survival and probability of capture.

Part I Exercises

1) The capture probability can have a large impact on your estimates of survival, particularly the standard errors. Upload the data set called 'survivallimited.csv', which has half of the captures for individuals randomly removed (half the 1's changed to 0's). Rerun the sex model using this data. How less reliable are the survival results? Importantly, we did not change the number of individuals in the dataset, only the number of times they were captured.

5.3 Part II: POPAN analysis for survival, probability of capture, recruitment, and population size

Often we are interested in more than just survival and probability of capture. We also want to know how many new individuals are entering the population (through new births or migration) and we also want estimates of population size. Each of these quantities are influenced by our probability of capture.

In last week's lab, we tested whether there was evidence for logistic growth in the guppy populations. These analyses used only the number of individuals captured as our data. However, these results may give us the wrong answer if our capture probabilities depend on the number of individuals in the population. Let's redo that analysis after correcting for the capture probabilities in our estimates of population size.

```
dp <- process.data(survival,model="POPAN")
ddl <- make.design.data(dp)

Phi <- list(formula=~time)
p <- list(formula=~time)
pent <- list(formula=~time)


surv.mod <- mark(dp,ddl,model.parameters=list(Phi=Phi, p=p, pent=pent),
                 invisible=TRUE,silent=TRUE,output=FALSE)
```

Let's not worry too much about the estimates for survival and probability of capture. Instead, let's just focus on the estimates of population size. These are called 'derived' parameters because R uses several pieces of information to come up with these numbers. Because there are 42 rows of these, I am just going to print the top 6 using the head function.

```
head(surv.mod$results$derived$`N Population Size`)
```

```
##    estimate        se       lcl       ucl
## 1  75.97519  8.667833  60.79560  94.94484
## 2  44.44027  6.693502  33.13496  59.60284
## 3  50.95475  7.355067  38.45462  67.51819
## 4 109.72976 11.556381  89.31494 134.81082
## 5 173.73360 13.235656 149.66794 201.66885
## 6 175.24597 13.711884 150.36518 204.24375
```

The estimate column gives R's estimate of the population size at each capture, taking into account the probability of capture. Save the estimates as 'popsize'.

```
N.hat <- surv.mod$results$derived$`N Population Size`[,'estimate']
```

Then, let's get the data from last week's lab and see how the numbers compare.

```
data <- read.csv(file = "./Data/summaryGuppyDataLL.csv")
```

Sum the male and female numbers together.

```
data$total.Nt <- data$N.F + data$N.M
```

Look at the top few rows.

```
head(data)
```

```
##   stream sample N.F N.M total.Nt
## 1     LL      1  38  38       76
## 2     LL      2  16  22       38
## 3     LL      3  22  23       45
## 4     LL      4  56  42       98
## 5     LL      5  94  75      169
## 6     LL      6  89  57      146
```

Let's put these together into the same dataframe. To do this, run the following code.

```
data <- cbind(data,N.hat)
head(data)
```

```
##   stream sample N.F N.M total.Nt     N.hat
## 1     LL      1  38  38       76  75.97519
## 2     LL      2  16  22       38  44.44027
## 3     LL      3  22  23       45  50.95475
## 4     LL      4  56  42       98 109.72976
## 5     LL      5  94  75      169 173.73360
## 6     LL      6  89  57      146 175.24597
```

The first thing you should notice is that the estimated population sizes should always be larger than the captured individuals. Let's plot these values.

```
plot(data$sample,data$N.hat,pch=16,xlab='Sample',ylab='N.hat')
points(data$sample,data$total.Nt,pch=1)
legend(c(5,1000),pch=c(16,1),c('N.hat','N.captured'),bty='n')
```

Now let's do our test using lambda. Remember we need to create another variable that is lagged, so that it has the next value on the same row.

```
data$N.hat.plus.1 <- c(data$N.hat[c(2:c(length(data$N.hat)))],NA)
```

Then λ is just the this new variable divided by the original one for that row.

```
data$lam.t <- data$N.hat.plus.1 / data$N.hat
head(data)
```

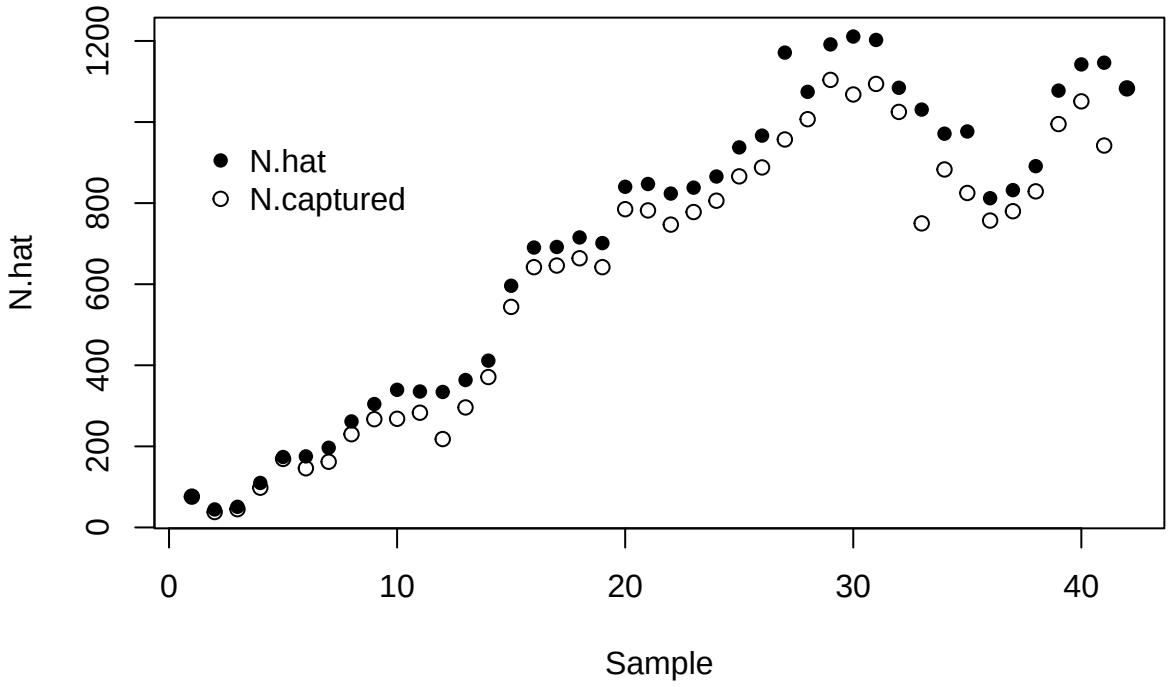


Figure 5.4: Estimated population size and number captured for each sampling period.

```
##   stream sample N.F N.M total.Nt     N.hat N.hat.plus.1     lam.t
## 1     LL      1  38  38       76  75.97519     44.44027 0.5849314
## 2     LL      2  16  22       38  44.44027     50.95475 1.1465895
## 3     LL      3  22  23       45  50.95475    109.72976 2.1534746
## 4     LL      4  56  42       98 109.72976    173.73360 1.5832860
## 5     LL      5  94  75      169 173.73360    175.24597 1.0087051
## 6     LL      6  89  57      146 175.24597    196.56472 1.1216505
```

Now plot it. This is usually done using the natural log of λ.

```
plot(data$N.hat, log(data$lam.t),xlab='N.hat', ylab=expression(ln~lambda(t)))
```

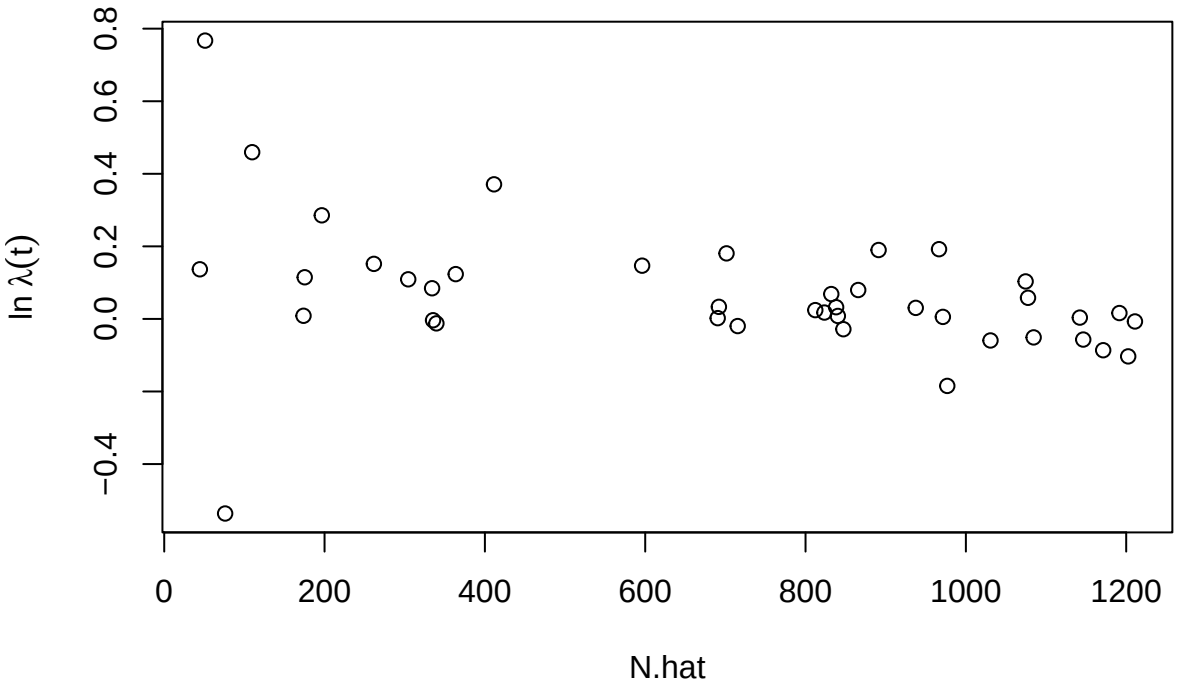


Figure 5.5: Natural log of lambda vs. the estimated population size.

Now, recall that the exponential growth model has a constant $ln\lambda$. In other words the birth and death rates do not change with increased population size. This forms our null hypothesis: H_0. This means that there should not be a significant relationship between $ln\lambda(t)$ and population size. If the relationship between $ln\lambda$ and population size is significant, then we can reject the null. If it is not significant, then we fail to reject the null. We can easily test this with a linear regression.

```r
mod <- glm(log(lam.t) ~ 1 + N.hat,data=data)
summary(mod)
```

```
##
## Call:
## glm(formula = log(lam.t) ~ 1 + N.hat, data = data)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -0.71349  -0.06326  -0.01935   0.04582   0.58531
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.910e-01  5.925e-02   3.224  0.00256 **
## N.hat       -1.815e-04  7.509e-05  -2.417  0.02041 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.03217832)
##
##     Null deviance: 1.443  on 40  degrees of freedom
## Residual deviance: 1.255  on 39  degrees of freedom
##   (1 observation deleted due to missingness)
## AIC: -20.592
##
## Number of Fisher Scoring iterations: 2
```

Wow, now the slope of the line (shown by "N.hat") is significant ($p < 0.05$). So we reject the null hypothesis that the growth is exponential. So, taking into account the probability of capture into our estimates of population size improved our ability to reject the null hypothesis. Now, the estimates that we had for population size based on the raw capture data here were not too bad. Our probabilities of capture were quite high. In most studies, they are never really that high. Probability of capture is usually somewhere south of 50%. In those instances, correcting for it will make a big difference.

Part II Exercises

1) Recall the the logistic growth model arises because the per-capita birth and death rates are linearly declining functions of population size. Plot the survival estimates from the POPAN model against the estimated population size estimates. Then do the test to see if there is a significant slope, just as we did for *lambda*. You can get the survival estimates and merge them into the data file using:

```
surv.hat <- surv.mod$results$real[grep("Phi",
                        dimnames(surv.mod$results$real)[[1]]),'estimate']

data$surv.hat <- c(surv.hat,NA) #The NA is there because there is
                                #one less estimate for
                                #survival than there are captures.
```

Is there evidence to suggest that survival decreases with increasing population size?

2) Do the same for the birth rates. You can get the birth rates from the output of the POPAN model using:

```
births.hat <- surv.mod$results$derived$`B* Gross Births`[,'estimate']
data$births.hat <- c(births.hat,NA)
#Also need this on a per-capita basis so divide through by the
#number of individuals in the population
data$births.per.hat <- data$births.hat / data$N.hat
```

Is there evidence to suggest that the birth rate decreases with increasing population size?

3) Put both 1 and 2 together to construct a statement about whether the declining population growth rate is due to density-dependence in survival or the birth rate.

Chapter 6

Pedigrees, Breeding Values, and Heritability

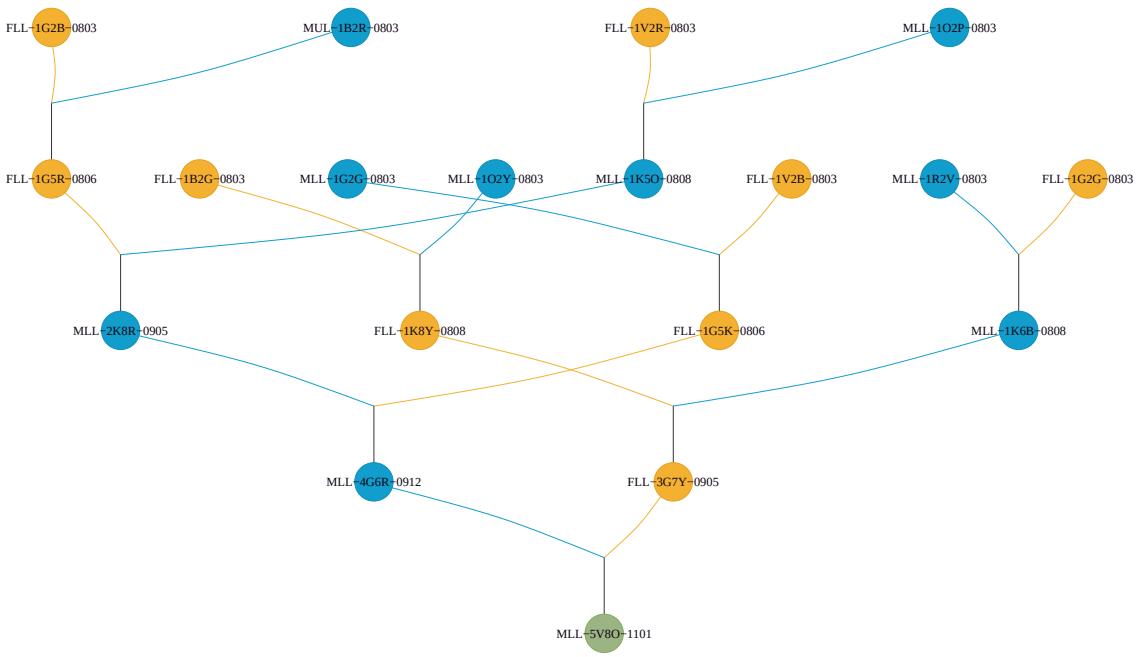


Figure 6.1: A pedigree from a population of guppies including all the ancestors of the individual in green.

6.1 Introduction

As we have discussed in lecture and discussion, the breeding values of an individual tell us the value of an individual as judged by the mean value of their offspring. These values are often represented as the deviation of that individual's from the mean trait value in the population. The variance in the breeding values is the additive genetic variance. The total variance in the trait is the phenotypic variance. So if we know both of these then we have an estimate of the heritability, h^2. Of course, one way to do this is to use the mean offspring mid-parent regression. This works very well for populations that are in the lab, animal breeding programs, or when males and females mate for life. We have also seen that heritability is only really useful if selection on the phenotype is the same as the selection on the genotype.

Yet, there is another way to figure out the breeding values of an individual. This is with something called the animal model. Unlike the mean offspring mid-parent regression used for heritability, the animal model calculates both the additive genetic variance and the phenotypic variance. To use the animal model, we need a pedigree, or a description of the all the relationships between the different individuals in the population and their descendants. Figure 1 shows a pedigree for a single individual in a guppy population.

If we were to look at all the individuals and their relationships, it looks like what we would get in figure 2.

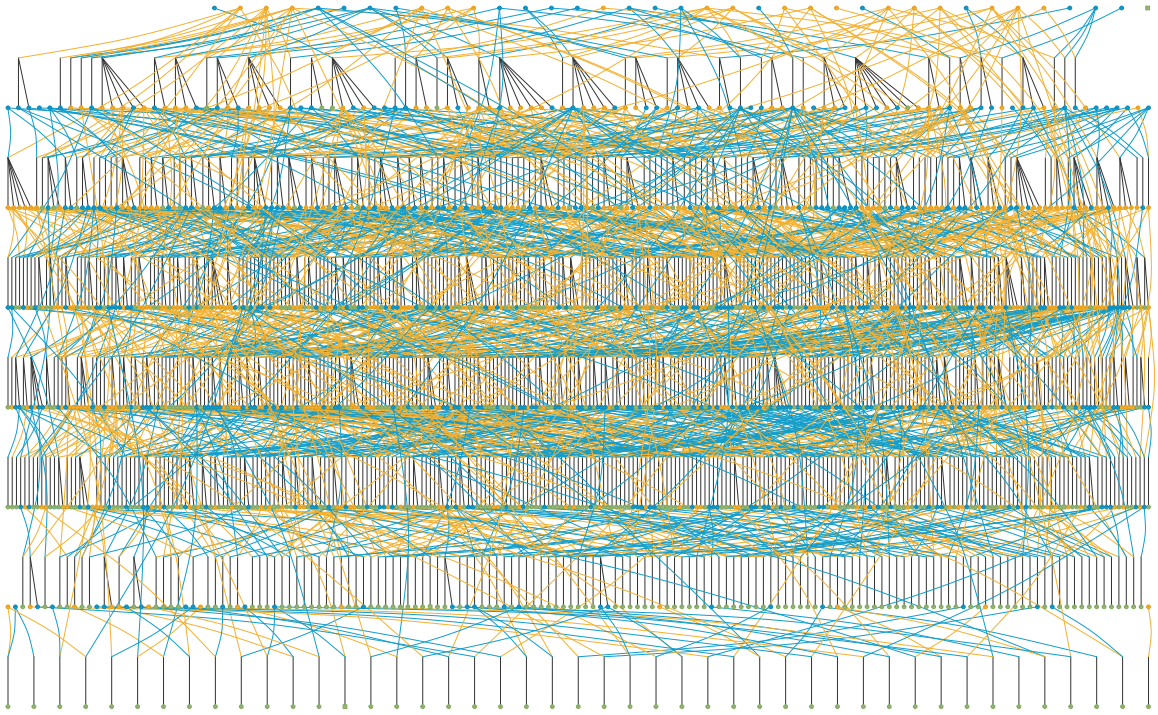


Figure 6.2: A pedigree from a population of guppies.

There are many ways to create a pedigree. For many organisms, pedigrees can be inferred simply through observing who mates with whom and keep track of their descendants. In other organisms, we cannot observe these pairings. In these cases, we have to rely on other methods. One method that has become increasingly popular is the use of microsatelites. This is what companies like 23 and Me and Ancestry.com use to find relatives of individuals. This is also how we keep track of ancestry in the guppy populations. Here we use 43 microsattelite markers to identify relatives. In this lab, we will not be concerned with how this is done, rather we will focus on using this the pedigree to estimate the breeding values and heritability of a trait.

Before beginning, we need to learn a little bit about how this is done. Relationships between individuals in natural populations are complex, hence we need an approach that can allow us to deal with these complexities. The animal model is this approach. It is a mixed model that allows us to separate the variance in traits across individuals into several different components. Among them, the additive genetic variance and the environmental variance (there are also others that come into play). Unlike parent offspring regressions, it uses information from the whole pedigree, and thus uses the maximum amount of information possible. This is the approach we will take below.

6.2 Part I: The Pedigree

First, let's set up our working file and take a look at the pedigree. Clear R's brain and set the working directory.

```
rm(list=ls())
```

```
setwd("~/Evolutionary Ecology/Lab 5")
```

Then, we need a couple of packages. Run these, but only once and then comment them out.

```
install.packages("tidyverse")
install.packages("MCMCglmm")
install.packages('parallel')
install.packages('coda')
```

Then load the packages. Run these each time you start RStudio.

```
require("tidyverse")
require("MCMCglmm")
require('parallel')
require('coda')
```

You will then need to upload these two files.

```
data <- read.csv(file = "./Data/data.csv",colClasses = "character")
ped <- read.csv(file = "./Data/ped.csv",colClasses = "character")
data$SLAM <- as.numeric(data$SLAM)
```

Take a look at the pedigree.

```
head(ped)
```

```
##          animal sire  dam
## 1 MUL-1R3O-0803 <NA> <NA>
## 2 MUL-1B2R-0803 <NA> <NA>
## 3 MUL-1B2Y-0803 <NA> <NA>
## 4 MUL-1V2G-0803 <NA> <NA>
## 5 MUL-1V2O-0803 <NA> <NA>
## 6 MUL-1V2R-0803 <NA> <NA>
```

The column animal gives the ID of the fish. These are unique to each fish. There are three parts to it. The first part gives the sex and stream, the second part is the unique mark given to the fish, and the third part is the year and month the fish was first captured. For example, MUL-1R3O-0803, is a male in the stream named UL, it has a red mark in the 1 position and an orange in the 3 position. It was first marked in March of 2008.

The second (sire) and third (dam) columns for this fish are blank (have NA's). Sire is the father and dam is the mother. This is rather odd language, but it is the language used in breeding programs, so we are stuck with it. The parents of this first fish is blank because it was one of the fish that was introduced–it is a founder. Let's look at the end of the file.

```
tail(ped)
```

```
##            animal          sire           dam
## 866 MLL-6V7B-0911 MLL-1Y7K-0905 FLL-1Y5Y-0810
## 867 MLL-6Y8R-0912          <NA> FLL-2R6G-0904
## 868 MLL-7R8G-0912 MLL-3G5R-0904 FLL-2P6Y-0905
## 869 MLL-7R8Y-0912          <NA> FLL-1K7Y-0808
## 870 MLL-7W8O-0908 MLL-2V8Y-0812 FLL-1P6R-0811
## 871 MLL-7Y8R-0912 MLL-3Y5O-0906 FLL-2R6B-0905
```

Great, now there is the ID of the parents in some of the sire and dam columns. Some are missing, meaning we do not know who the mother or father is. That is okay, the animal model can handle missing parents.

Now, we need a trait of the fish. We are going to use a trait that we can measure in the field. This trait is the standard length at maturity or "SLAM" for short. Standard length is

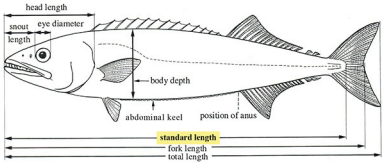


Figure 6.3: Measurements of fish

one type of measurement for the size of a fish. It goes from the tip of the snout to the place on the tail where the fish rays are inserted into the muscle (the hypural plate)(Figure 3).

Now take a look at the guppy data. It has animal in it just like the pedigree. It also has SLAM and some other information that will be used later.

```
head(data)
```

```
##          animal  SLAM LRS sampling year    pop.dens season         dadID
## 1 MLL-1B3W-0910 21.05   0       20    2 2.062550591    wet MLL-3G8K-0904
## 2 MLL-1B5B-0805 18.72   5        3    1 0.119502889    dry MUL-1B2R-0803
## 3 MLL-1B6W-0908 16.65   9       18    2 1.747805738    wet MLL-1P8Y-0902
## 4 MLL-1B7W-0910 16.81   0       20    2 2.062550591    wet          <NA>
## 5 MLL-1G2P-0803 18.56   7        1    1 0.215915196    dry          <NA>
## 6 MLL-1G2W-0909 16.18   0       19    2 1.667643089    wet MLL-2R8P-0905
##           mumID
## 1 FLL-3K7P-0905
## 2 FLL-1G2B-0803
## 3 FLL-3K6V-0812
## 4          <NA>
## 5          <NA>
## 6 FLL-3G8O-0810
```

6.3 Part II. The Animal Model

In order to understand the animal model, we need to learn a little bit about linear models. SLAM is a phenotypic measurement and there is one per individual. These measurements are in a vector and there are n individuals. For simplicity, we will replace SLAM with y in the notation below:

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \tag{6.1}$$

The purpose of the animal model is to separate the variance in Y into the a variance that that is due to the breeding values and the rest. To do that, consider the the phenotype

of y_i of individual i in the population. We can express this phenotypic value as the mean phenotype in the population $\bar{y}$ plus the difference between the phenotype of individual i and the mean, d_i.

$$y_i = \mu + d_i \tag{6.2}$$

This difference is due to both the individual i's breeding value a_i (due to the additive effect of its alleles) and an effect of the environment, e_i.

$$d_i = a_i + e_i \tag{6.3}$$

We can insert equation 3 into equation 2 to get:

$$y_i = \mu + a_i + e_i \tag{6.4}$$

Now we are interested in the variance in breeding values across all individuals in the population. The breeding values across all individuals in the population are described by:

$$\begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \sim N(0, \mathbf{A}\sigma_A^2) \tag{6.5}$$

This is a statistical description of the variance, which is opposed to the description we saw in lecture that arises from the underlying mechanisms of the interaction between alleles at a locus. This also includes any number of potential loci that are involved in the trait, which is unknown. The part on the right, $N(0, \mathbf{A}\sigma_A^2)$ is shorthand for saying that we assume the values on the left come from a normal (bell-shaped) distribution with a mean of 0 and a variance that is equal to σ_A^2, which is what we want to estimate. The matrix $\mathbf{A}$ comes from the pedigree and expresses how each individual is related to every other individual that is current, or ever has been, in the population.

Likewise, we need something similar for the variation in the traits that is due to everything else.

$$\begin{pmatrix} e_1 \\ \vdots \\ e_n \end{pmatrix} \sim N(0, \mathbf{I}\sigma_E^2) \tag{6.6}$$

The $\mathbf{I}$ in this case is a matrix with 1's on the diagonal and 0's everywhere else.

From these equations we can see that there are 3 key assumptions. 1) The trait is normally distributed (has a bell-shaped distribution across individuals). 2) The breeding values are normally distributed and there is a genetic correlation among related individuals, given by $\mathbf{A}$. 3) The residuals, e_i's are normally distributed, uncorrelated between individuals, and are independent of the breeding values. In other words, there is not a genotype x environment interaction.

Thus the primary outputs of the animal model are $\bar{y}$, σ^2_A, and σ^2_E.

6.4 Part III. Calculating additive genetic variance (i.e. variance in breeding values) and heritability

The type of statistics that are used to actually fit the models are called Bayesian statistics. There is literally whole classes on this and we will not spend any time on this. But one thing we need to do is to specify something called a prior distribution. That is a distribution that is a reasonable start for what we might get. The priors we will use are "uninformative", that is they in effect say that we know nothing about the distributions of the $\bar{y}$, σ^2_A, and σ^2_E.

This is the priors for the animal model.

```
prior_guppy <- list(R = list(V=1, nu=0.002), G = list(G1 = list(V=1, nu=0.002)))
```

Run the model. This is just a fancy linear model.

```
model <- MCMCglmm(SLAM ~ 1  ,# this is the model statement that says what
                             #the relationship between SLAM and
                             #the non-random effects are.
                             # The 1 means to find the mean.
                  random = ~ animal , # estimate the G-matrix.
                                      #This says that the information
                                      #for the A matrix is in the 'animal' column.
                  family = c("gaussian"), # specify trait distribution.
                                          #Gaussian means a
                                          #normal distribution
                  pr = TRUE, # saves estimated breeding values
                  pedigree = ped, # specify pedigree data
                  data = data,  # specify phenotypic data
                  prior = prior_guppy, # specify the priors to use
                  nitt = 44000, thin = 50, burnin = 4000)
```

```
##
##                       MCMC iteration = 0
##
##                       MCMC iteration = 1000
##
##                       MCMC iteration = 2000
##
##                       MCMC iteration = 3000
##
```

```
##                         MCMC iteration = 4000
##
##                         MCMC iteration = 5000
##
##                         MCMC iteration = 6000
##
##                         MCMC iteration = 7000
##
##                         MCMC iteration = 8000
##
##                         MCMC iteration = 9000
##
##                         MCMC iteration = 10000
##
##                         MCMC iteration = 11000
##
##                         MCMC iteration = 12000
##
##                         MCMC iteration = 13000
##
##                         MCMC iteration = 14000
##
##                         MCMC iteration = 15000
##
##                         MCMC iteration = 16000
##
##                         MCMC iteration = 17000
##
##                         MCMC iteration = 18000
##
##                         MCMC iteration = 19000
##
##                         MCMC iteration = 20000
##
##                         MCMC iteration = 21000
##
##                         MCMC iteration = 22000
##
##                         MCMC iteration = 23000
##
##                         MCMC iteration = 24000
##
##                         MCMC iteration = 25000
##
##                         MCMC iteration = 26000
```

```
##
##                          MCMC iteration = 27000
##
##                          MCMC iteration = 28000
##
##                          MCMC iteration = 29000
##
##                          MCMC iteration = 30000
##
##                          MCMC iteration = 31000
##
##                          MCMC iteration = 32000
##
##                          MCMC iteration = 33000
##
##                          MCMC iteration = 34000
##
##                          MCMC iteration = 35000
##
##                          MCMC iteration = 36000
##
##                          MCMC iteration = 37000
##
##                          MCMC iteration = 38000
##
##                          MCMC iteration = 39000
##
##                          MCMC iteration = 40000
##
##                          MCMC iteration = 41000
##
##                          MCMC iteration = 42000
##
##                          MCMC iteration = 43000
##
##                          MCMC iteration = 44000
```

The model will chug away 44000 times. When it is done, you can find the breeding values and create a histogram of the estimated breeding values using:

```
means <- colSums(model$Sol) / dim(model$Sol)[1]
breed <- means[2:c(length(means)-1)]
hist(breed,xlab='breeding values')
```

This distribution has a mean of zero and a variance, the additive genetic variance. You can

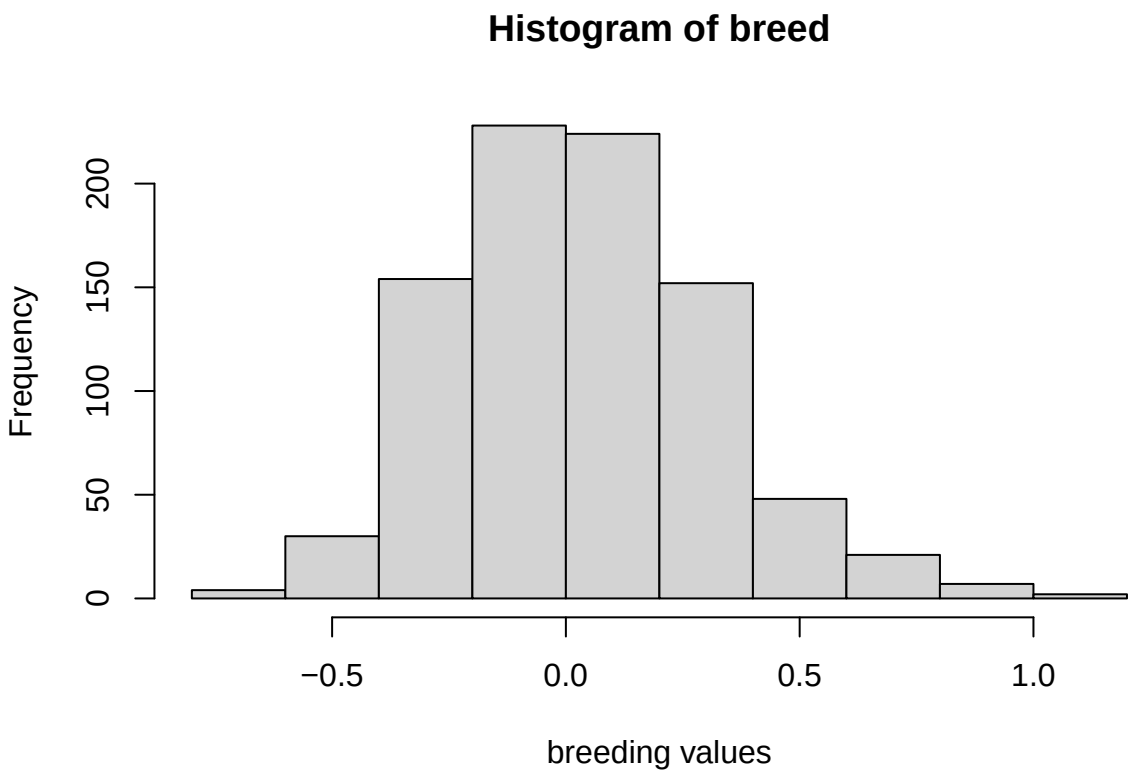


Figure 6.4: Histogram of breeding values.

get a visual of the estimate of this variance using:

These are the estimates for the additive genetic variance (animal) and the variance not accounted for by the additive genetic variance (unit).

```
plot(model$VCV)
```

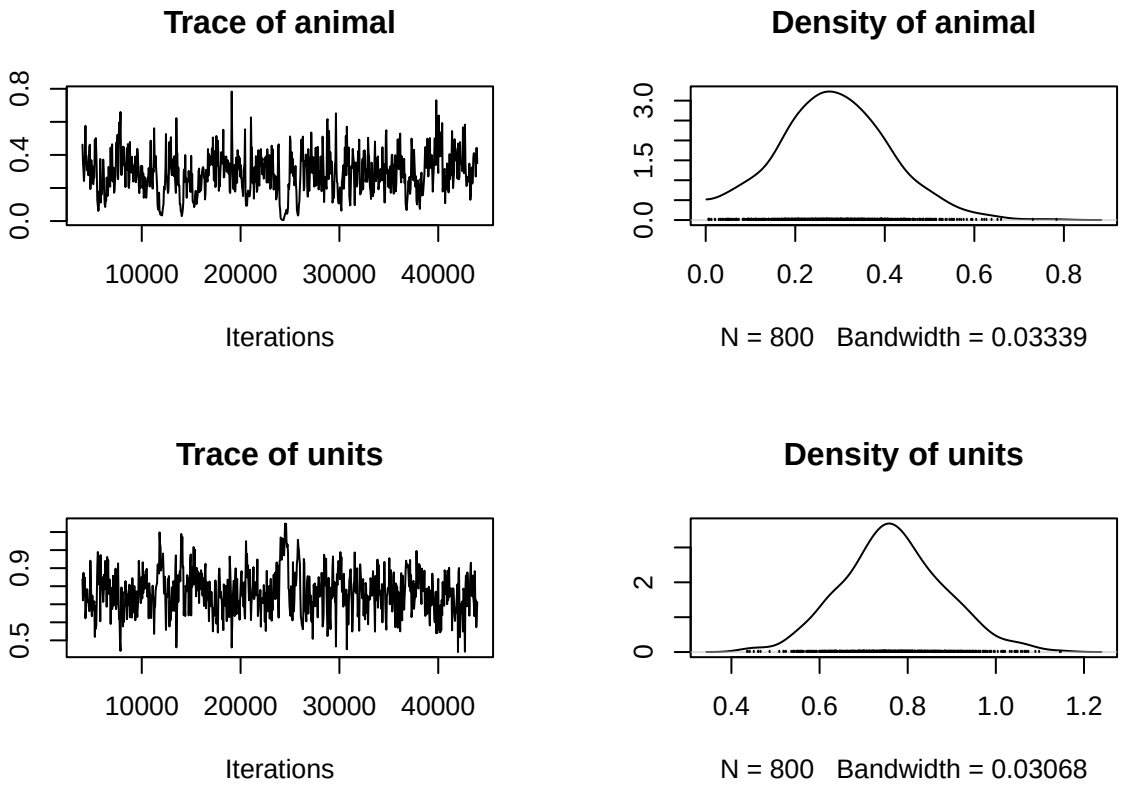


Figure 6.5: Output from the animal model.

The left columns are the 'traces' of the additive genetic variance (labelled 'animal') and the environmental plus any nonadditive genetic variance (labelled 'units'). The traces are the values of the variances for each as the program "walks" over the distribution of estimates for each variance. The plots on the right are smoothed probability distributions of the two variances. Careful not to confuse this with the breeding values themselves. The plot in the upper right is the estimate of the variance in breeding values. Likewise, the lower right is the distribution of the residual, environmental, variance.

Our best estimate of σ^2_A is the mean of the distribution in the upper right. Likewise, our best estimate of the environmental variance is the mean of the distribution in the lower right. We can use this to calculate the heritability. We can also use this to give us some 95% confidence intervals on this number.

```r
V.A <- mean(model$VCV[, "animal"])


V.P <- model$VCV[, "animal"] + model$VCV[, "units"]

herit <- V.A / V.P

mean(herit)
```

```
## [1] 0.2740956
```

```r
HPDinterval(herit)
```

```
##          lower    upper
## var1 0.2419684 0.310195
## attr(,"Probability")
## [1] 0.95
```

```r
hist(herit)
```

Part III Exercises

1) Redo this analysis, but use three years of data instead of the two years of data. You will need the files "data3.csv" and "ped3.csv". Calculate the additive genetic variance (σ^2_A), Phenotypic variance (σ^2_P), and the heritability (h^2). Did the heritability change? Was this due to changes in σ^2_A, σ^2_P, or both?

2) If the additive genetic variance increased or decreased, why might you expect this to happen?

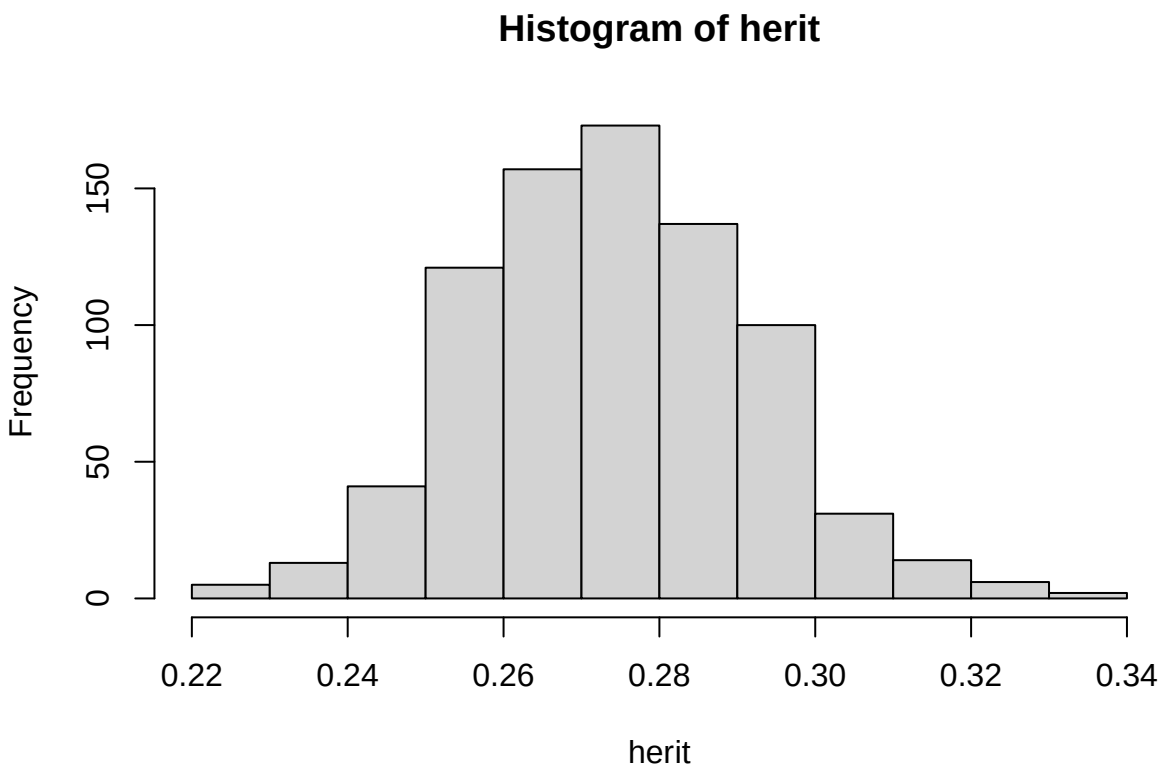


Figure 6.6: Histogram of the posterior distribution for heritability.

Chapter 7

Natural Selection on SLAM

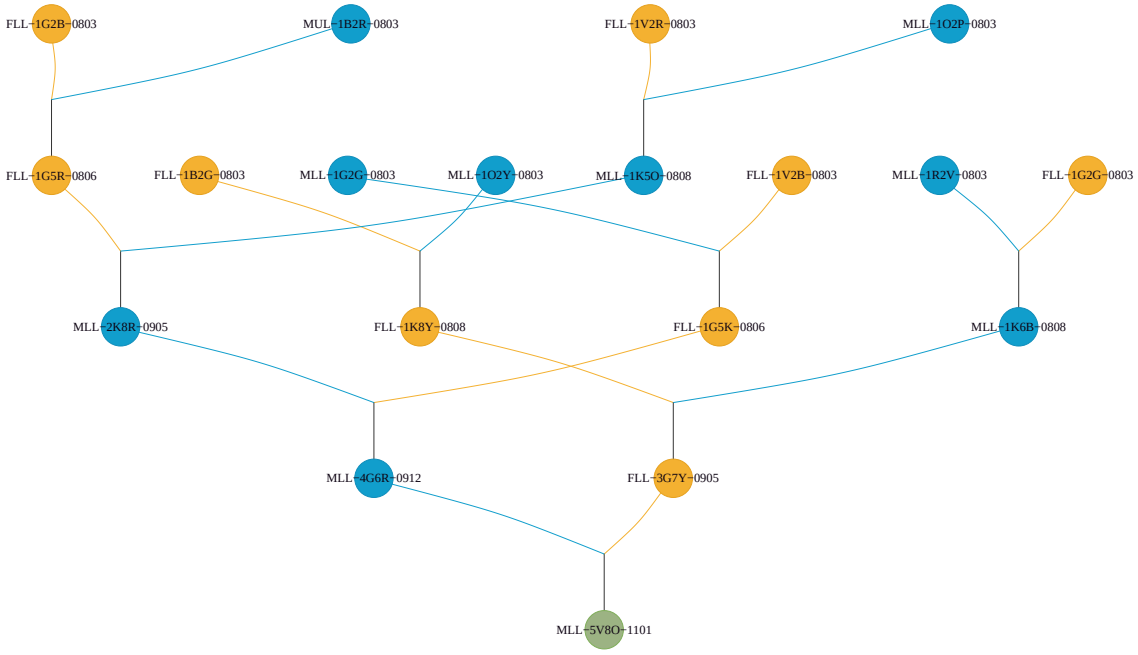


Figure 7.1: A pedigree from a population of guppies including all the ancestors of the individual in green.

7.1 Introduction

In the last lab, we learned how to estimate the additive genetic variance of a trait σ^2_A using a pedigree and the animal model. This week we will take this one step further and use the data to estimate the selection gradient (β). We will then combine these two measures to

predict the direction and magnitude of evolution on a trait. We will then use phenotypic and breeding values measures of the trait over time to see if change in the trait conforms to what we predicted. Finally, we will use a multivariate version of the animal model to estimate the covariance between the additive genetic variance in the trait and fitness. All together we will find that the Breeder's equation does a poor job in predicting the direction and magnitude of evolutionary change in the population over time, but the Robertson-Price equation does a reasonable job at predicting both the direction and the magnitude of trait change.

7.2 Part I. Natural Selection on SLAM Using the Breeder's Equation

First, let's set up our working file and take a look at the pedigree. Clear R's brain and set the working directory.

```
rm(list=ls())
```

```
setwd("~/Evolutionary Ecology/Lab 6")
```

Then, we need a couple of packages. Run these, but only once and then comment them out.

```
install.packages("plyr")
install.packages("dplyr")
install.packages("tidyverse")
install.packages("MCMCglmm")
install.packages('parallel')
```

Then load the package. Run this each time you start RStudio.

```
require("plyr")
```

Then load the packages. Run these each time you start RStudio.

```
require("plyr")
require("dplyr")
require("tidyverse")
require("MCMCglmm")
require('parallel')
```

Upload the three year dataset.

```
data3 <- read.csv(file = "./Data/data3.csv")
ped3 <- read.csv(file = "./Data/ped3.csv")
```

Take a look at the data.

```
head(data3)
```

```
##          animal  SLAM LRS sampling year  pop.dens season         dadID
## 1 MLL-1B3W-0910 21.05   0       20    2 2.0625506    wet MLL-3G8K-0904
## 2 MLL-1B5B-0805 18.72   5        3    1 0.1195029    dry MUL-1B2R-0803
## 3 MLL-1B6W-0908 16.65   9       18    2 1.7478057    wet MLL-1P8Y-0902
## 4 MLL-1B7W-0910 16.81   0       20    2 2.0625506    wet          <NA>
## 5 MLL-1F2B-1010 16.80   1       32    3 3.4067891    wet MLL-4P7K-1003
## 6 MLL-1F3O-1012 16.04   4       34    3 2.3335318    dry MLL-2W8G-1006
##           mumID
## 1 FLL-3K7P-0905
## 2 FLL-1G2B-0803
## 3 FLL-3K6V-0812
## 4          <NA>
## 5 FLL-2O8W-0908
## 6          <NA>
```

With the pedigree we not only have the ability to estimate σ^2_A and h^2, but we also have a measure of fitness for each individual. Because we know who is related to the whom and how, we can calculate the life time reproductive success (R_0) of individuals, which is one measure of fitness. It is coded as "LRS" in the data file. Here we will do so using the three year data set. Hence, because we have a phenotypic measure of a trait, SLAM, and fitness, we can estimate natural selection acting on the trait.

First, let's plot the data.

```
plot(data3$SLAM,data3$LRS,xlab='SLAM (mm)',ylab='Lifetime Reproductive Success')
```

There are lots of zeros (no offspring) and some with lots of offspring. There looks to be positive selection on SLAM at the level of the phenotype. Let's see if it is a significant positive trend. Here, we are doing just a linear regression. Remember from Morrissey et al 2010 and lecture that the selection gradient is the slope of the line between fitness and the phenotype.

```
mod <- glm(LRS ~ 1 + SLAM,data=data3)
summary(mod)
```

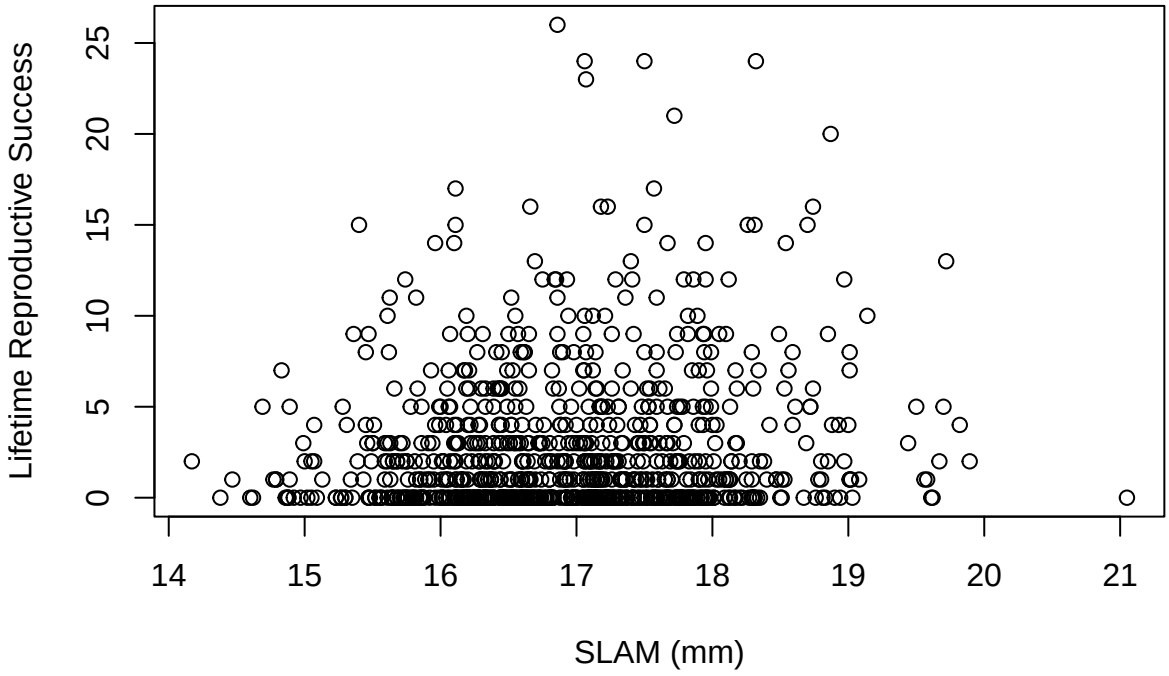


Figure 7.2: Fitness (LRS) versus body size (standard length at maturity).

```
##
## Call:
## glm(formula = LRS ~ 1 + SLAM, data = data3)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -4.818  -2.546  -1.489   1.218  23.267
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.6580     2.3170  -2.442 0.014812 *
## SLAM          0.4977     0.1365   3.645 0.000283 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 15.23466)
##
##     Null deviance: 13137  on 850  degrees of freedom
## Residual deviance: 12934  on 849  degrees of freedom
## AIC: 4736.8
##
## Number of Fisher Scoring iterations: 2
```

```
#Save the slope as beta
beta <- coef(mod)['SLAM']
```

Looks like there is indeed positive selection on SLAM. The coefficient telling us the slope (SLAM) is positive and significant. Let's put the fitted line on our plot.

```
plot(data3$SLAM,data3$LRS,xlab='SLAM (mm)',ylab='Lifetime Reproductive Success')
lines(c(14:21),(coef(mod)['(Intercept)'] + coef(mod)['SLAM']*c(14:21)))
```

Okay, let's think about what this means. In the last lab, we generated values for σ_A^2 and h^2 of SLAM. We now have an estimate of the selection gradient on SLAM, β, which is the slope of the line between fitness and the trait value.

In the breeder's equation, the change in the mean trait value is then $\Delta\bar{z} = \sigma_A^2\beta$. All we should need to do is plug in our estimates of σ_A^2 and β and we should have an estimate of how much SLAM should change. Let's try it.

Use this code to recalculate σ_A^2 for the 3 year data set. Notice the names of the two datasets end in a '3'.

This the priors for the animal model.

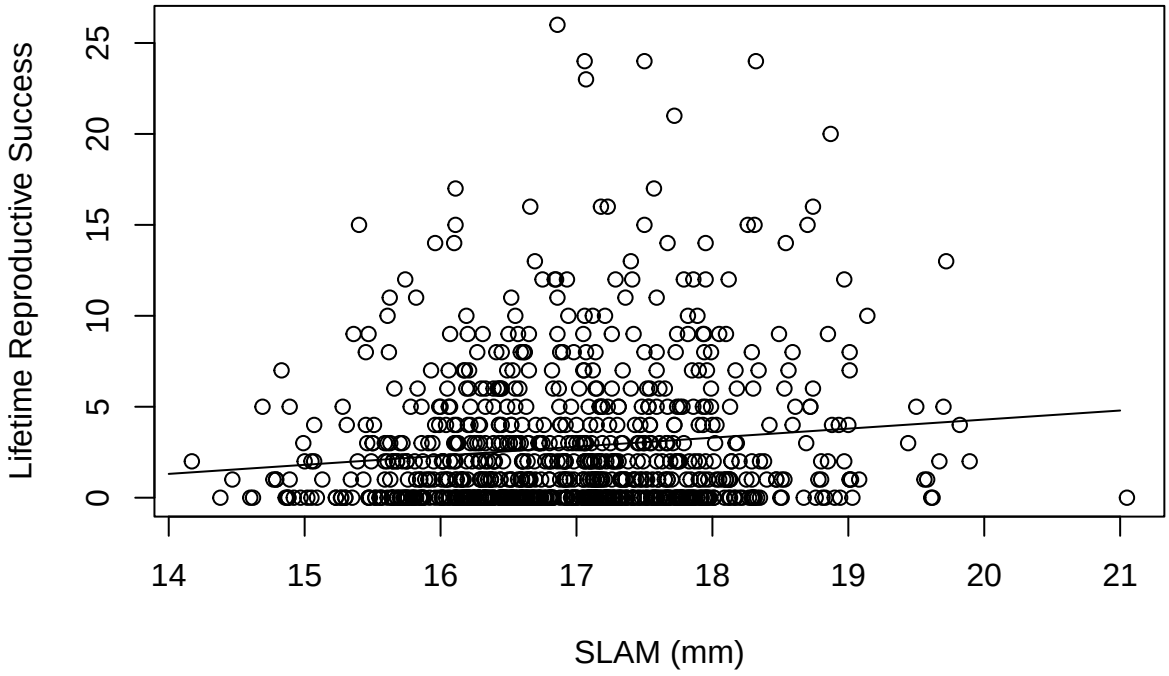


Figure 7.3: Fitness (LRS) versus body size (standard length at maturity) with fit line.

```
prior_guppy <- list(R = list(V=1, nu=0.002), G = list(G1 = list(V=1, nu=0.002)))
```

```
model <- MCMCglmm(SLAM ~ 1  ,# this is the model statement that says what
                             #the relationship between SLAM and the
                              #non-random effects are.
                             # The one means to find the mean.
                  random = ~ animal , # estimate the G-matrix.
                                      #This says that the information
                                      #for the A matrix is in the 'animal' column.
                  family = c("gaussian"), # specify trait distribution.
                                          #Gaussian means a
                                          #normal distribution
                  pr = TRUE, # saves estimated breeding values
                  pedigree = ped3, # specify pedigree data
                  data = data3,  # specify phenotypic data
                  prior = prior_guppy, # specify the priors to use
                  nitt = 44000, thin = 50, burnin = 4000)
```

```
V.A <- mean(model$VCV[, "animal"])
V.A
```

```
## [1] 0.3712059
```

To predict how much SLAM should change. All we need to do based on the breeder's equation is multiply σ^2_A by β. We have both these.

```
delta.z.bar <- V.A * beta
delta.z.bar
```

```
##      SLAM
## 0.1847347
```

This says we expect to see SLAM increase 0.1847 mm. Do we?

7.3 Part II. Change in the Phenotypic and Breeding Values

In this section we will look at how the phenotypic values of SLAM have changed over time. Our prediction from the Breeder's equation above says that it should have increased over time. Let's see if that is the case.

This bit of code will tell you how the mean of the breeding values has changed through time.

```
#Breeding values
means <- colSums(model$Sol) / dim(model$Sol)[1]
breed <- means[2:c(length(means)-1)]


animal <- unlist(strsplit(names(breed),"[.]"))[c(F,T)]
breeding <- as.data.frame(cbind(animal,breed))

newdata <- merge(x=data3,y=breeding,by='animal',all.x=TRUE)
newdata$breed <- as.numeric(newdata$breed)

breed.yr <- ddply(newdata,c('year'),summarise,mn.breed = mean(breed,na.rm=TRUE))
breed.yr
```

```
##   year    mn.breed
## 1    1 0.341153107
## 2    2 0.020734249
## 3    3 0.001882641
```

The average change in the breeding values over the years is:

```
mean(c((0.0207-0.34),c(0.00188-0.0207)))
```

```
## [1] -0.16906
```

```
plot(breed.yr$year,breed.yr$mn.breed,ylab='mean breeding value',xlab='year')
```

This tells us that the mean breeding values were the largest in the first year and then declined across years, meaning SLAM should be getting smaller, not larger as predicted!

What about the change in the phenotypic values over time?

```
phenos <- ddply(newdata,c('year'),summarise,mn.breed = mean(SLAM,na.rm=TRUE))


plot(phenos$year,phenos$mn.breed,ylab='mean phenotypic value',xlab='year')
```

Well, these are declining also. So, the change in the breeding values and the change in the phenotypic values agree with each other. Nevertheless, both are going in the opposite direction than predicted based on the selection analysis at the phenotypic level.

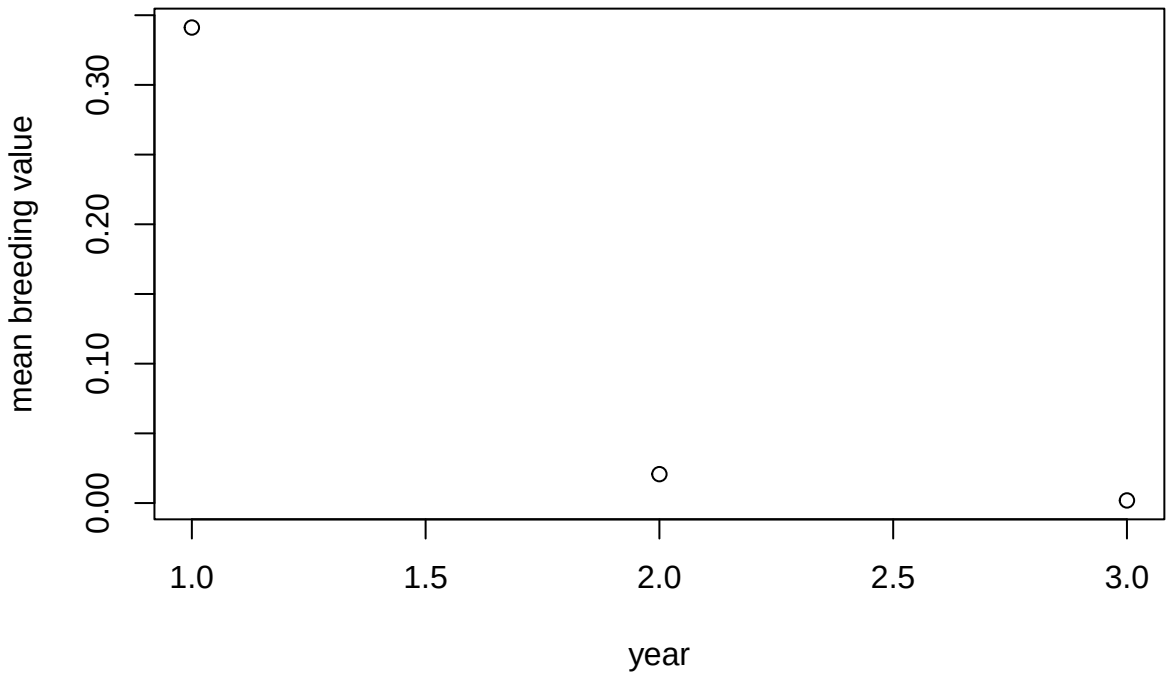


Figure 7.4: Mean breeding values in the first three years.

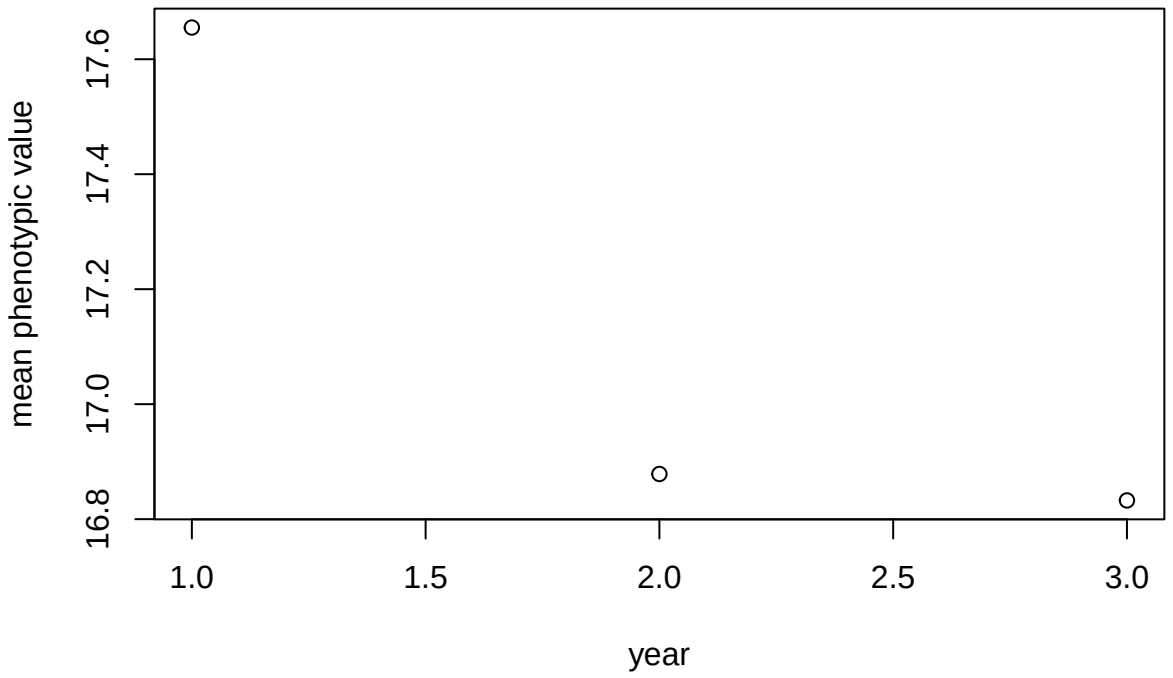


Figure 7.5: Mean phenotypic values in the first three years.

Part II Exercises

1) The predicted direction of evolutionary change in our example was opposite to the change that was actually observed in both the mean breeding values and phenotype. Why might this be the case? There are actually several possibilities. One possibility is that our selection gradient may not be linear (directional selection) and may be curved (stabilizing selection). Come up with another possibility based on our previous lab work and discussions.

7.4 Part III. The Robertson-Price Identity

One reason that we failed to predict even the correct direction of evolution lies in the fact that the Breeder's equation is poorly suited to study evolution in natural populations (Morrissey et al 2010). In particular, the Breeder's equation will only predict the correct direction of evolutionary change if selection at the phenotypic level is the same as selection at the level of the breeding values. In other words: $\frac{\sigma_A(z,w)}{\sigma^2_A(z)} = \frac{\sigma_P(z,w)}{\sigma^2_P(z)}$. This does not appear to be the case in our data.

Remember that the $\sigma_A(z,w)$ is the covariance between fitness and the breeding values of the trait, SLAM. A covariance is a measure of how related to variables are to each other and is closely related to a correlation. Also, a covariance divided by a variance is a regression slope. So, the right side of the equation above is actually just the regression parameter we measured using linear regression in Part I, β.

Here we will use the Robertson-Price Identity to calculate the covariance between the breeding values and fitness and the variance in the breeding values. This is the left hand side of the equality above. Both the Breeder's equation and Robertson-Price Identity predict the change in a trait over a generation. That is the left hand side of the equation is the same. But how they do this is quite different. The Robertson-Price Identity relates the change to the relationship (covariance) between the breeding values of the trait and fitness, or the additive genetic covariance between the trait and fitness.

$$\Delta z = \sigma_a(z,w) \tag{7.1}$$

It turns out that we can use the animal model to estimate this covariance. The key thing to realize is that we must now estimate three variances and one covariance. We now are looking for σ^2_a and now also σ^2_w and $\sigma_a(z,w)$. This is now a multivariate animal model. Multivariate because we have will have two variables on the left hand side of the equals sign. We use the same pedigree information and the same data. This is just a different model fit to the data.

We can run the multivariate model using:

```
prior_guppy <- list(G=list(G1=list(V=diag(2)/2, nu=1.002)),
                    R=list(V=diag(2),nu=1.002))
```

```
  # The model
model <-  MCMCglmm(cbind(SLAM, LRS) # bivariate response: SLAM (z) and LRS (w)
           ~ trait-1 , # allow separate intercepts for z and w
           random = ~ us(trait):animal,  # estimate the G-matrix
           # + us(trait):mumID, # estimate the M-matrix
           rcov = ~ us(trait):units, # estimate the E-matrix
           family = c("gaussian","poisson"), # specify trait distributions
           pr = TRUE, # saves estimated breeding values
           pedigree = ped3, # specify (trimmed) pedigree data
           data = data3,  # specify phenotypic data
           prior = prior_guppy, # specify the priors to use
           nitt = 60000, thin = 500, burnin = 6000)
```

We can plot the traces of the covariance between fitness and SLAM and also the variance in SLAM using:

```
plot(model$VCV[,c('traitLRS:traitSLAM.animal','traitSLAM:traitSLAM.animal')])
```

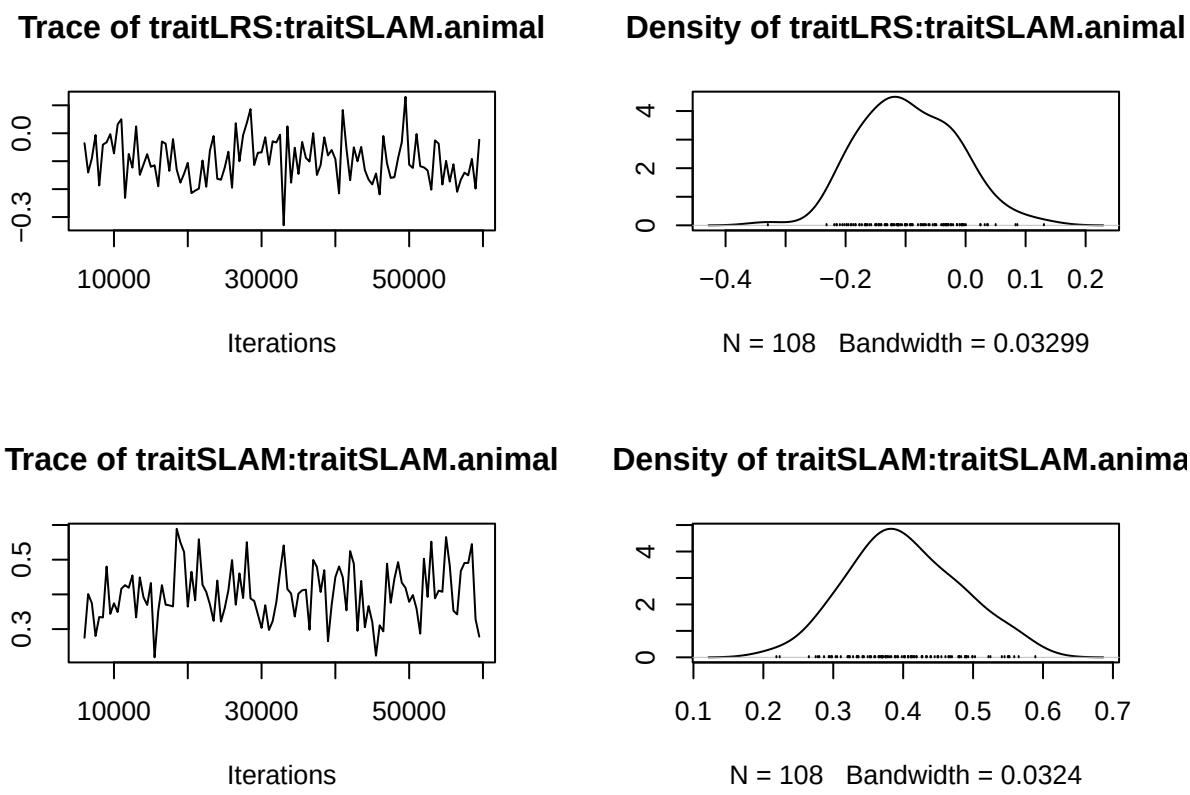


Figure 7.6: Output from the animal model.

The upper row is the traces for the covariance and the lower row is the trace for the variance in the breeding values of SLAM. Both of these look pretty good.

We can calculate the genetic covariance between the additive genetic components and fitness using:

```
#Additive genetic covariance
cov.w.SLAM <- mean(model$VCV[,'traitLRS:traitSLAM.animal'])
cov.w.SLAM
```

```
## [1] -0.09461829
```

This is the prediction of change from the Robertson-Price Identity. Compare this with the observed change in the breeding values over time from above. You can see that they have the same sign, but the prediction from the Robertson-Price Identity is a little less than what is observed.

This is the additive genetic variance in SLAM

```
#Additive variance in SLAM
var.SLAM <- mean(model$VCV[,'traitSLAM:traitSLAM.animal'])
var.SLAM
```

```
## [1] 0.4022239
```

We divide the covariance by the variance to get the slope based on the additive genetic variances and covariances

```
cov.w.SLAM / var.SLAM
```

```
## [1] -0.2352379
```

Thus, our "slope" as estimated from the Robertson-Price Identity is negative. Meaning that selection on the additive genetic variance (i.e. breeding values) is actually negative, not positive as estimated from the application of the Breeder's equation and selection at the phenotypic level.

We can plot these values to have a look at what this looks like using:

```
SLAM <-colSums(model$Sol[,grep("traitSLAM.animal.",dimnames(model$Sol)[[2]])]) /
  dim(model$Sol[,grep("traitSLAM.animal.",dimnames(model$Sol)[[2]])])[1]

LRS <- colSums(model$Sol[,grep("traitLRS.animal.",dimnames(model$Sol)[[2]])]) /
  dim(model$Sol[,grep("traitLRS.animal.",dimnames(model$Sol)[[2]])])[1]

plot(SLAM,LRS,ylab='Fitness (LRS)',xlab='Breeding values for SLAM')
```

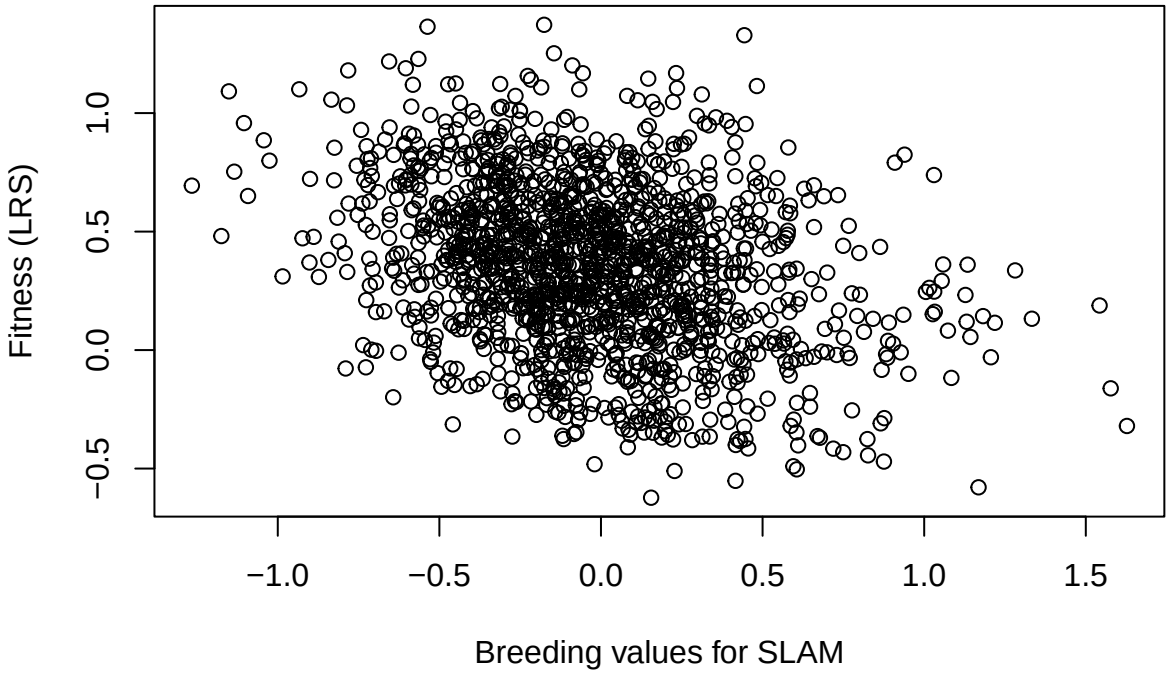


Figure 7.7: Fitness (LRS) versus breeding values for SLAM.

This clearly looks negative. One last thing we can do here is to ask whether there is evidence that the "slope" is significant. You can do this using the following code:

```
cov.gen <- model$VCV[,'traitLRS:traitSLAM.animal'] /
           model$VCV[,'traitSLAM:traitSLAM.animal']
mean(cov.gen)
```

```
## [1] -0.2391591
```

```
HPDinterval(cov.gen)
```

```
##           lower     upper
## var1 -0.6453877 0.1203478
## attr(,"Probability")
## [1] 0.9537037
```

We cannot really say that it is negative because the 95% CI overlaps zero. But this is better than before. At the very least, the mean direction of change we predict is the same as that which is observed at both the phenotypic and breeding value levels.

Part III Exercises

1) Our analysis shows that the selection at the level of the breeding values differs in direction from the direction of selection at the phenotypic level. Go back to Morrissey et al (2010). Which of the four scenarios in Figure 1 do we find ourselves in? To help sort that out, you can use the following code to calculate the environmental covariance in the trait with fitness. There is not really a good way to plot this.

```
#Environmental covariance
mean(model$VCV[,'traitLRS:traitSLAM.units'])
```

2) There are literally thousands of phenotypic measures of selection in the literature. For years, it was everyone's favorite way to get a PhD dissertation. One result of these analyses is that selection is often strong and directional. Yet, in cases where this is found, we often see no evolution or evolution in a direction that is different than that which is predicted based on the selection gradient. For a long time, this presented a paradox. How can you have strong selection without evolutionary change or evolutionary change in the opposite direction? What do our results say about this paradox? What types of information do our results say one needs to have to be able to make inferences about evolutionary change?

Chapter 8

Species Coexistence

8.1 Part I: Visualizing Competition and Coexistence

In this section, we will be working with to create phase planes where we can visualize the necessary conditions for species to coexist in constant, fluctuation-independent, environments. Recall that phase planes have the population sizes of the species as axes and lines that represent the zero-growth isoclines for each species. Along the way, we will also learn something new about Program R. That is how to build your very own function. Recall that functions in R are things like "log(x)" that returns the natural log of x. One nice thing about R is that we can fairly easily build our own functions. This allows us to do things several times, without the annoyance of having to type all the necessary code each time you want to run a series of commands.

Species coexistence in discrete time

To start, we will be working with the a two species version of the Ricker Model. The Ricker model is convinient for this because it is linear on the natural log scale. Recall that the Ricker model for one species is:

$$N(t+1) = N(t)e^{r-r/KN(t)} \tag{8.1}$$

We can modify this in a manner similar to what we did for the continuous logistic equation by:

$$N_1(t+1) = N_1(t)e^{r_1 - \frac{r_1}{K_1}N_1(t) - \alpha_{12}\frac{r_1}{K_1}N_2(t)} \tag{8.2}$$

and a similar equation for the species 2:

$$N_2(t+1) = N_2(t)e^{r_2 - \frac{r_2}{K_2}N_2(t) - \alpha_{21}\frac{r_2}{K_2}N_1(t)} \tag{8.3}$$

One way to visualize the conditions for when two species can coexist is to simply put in

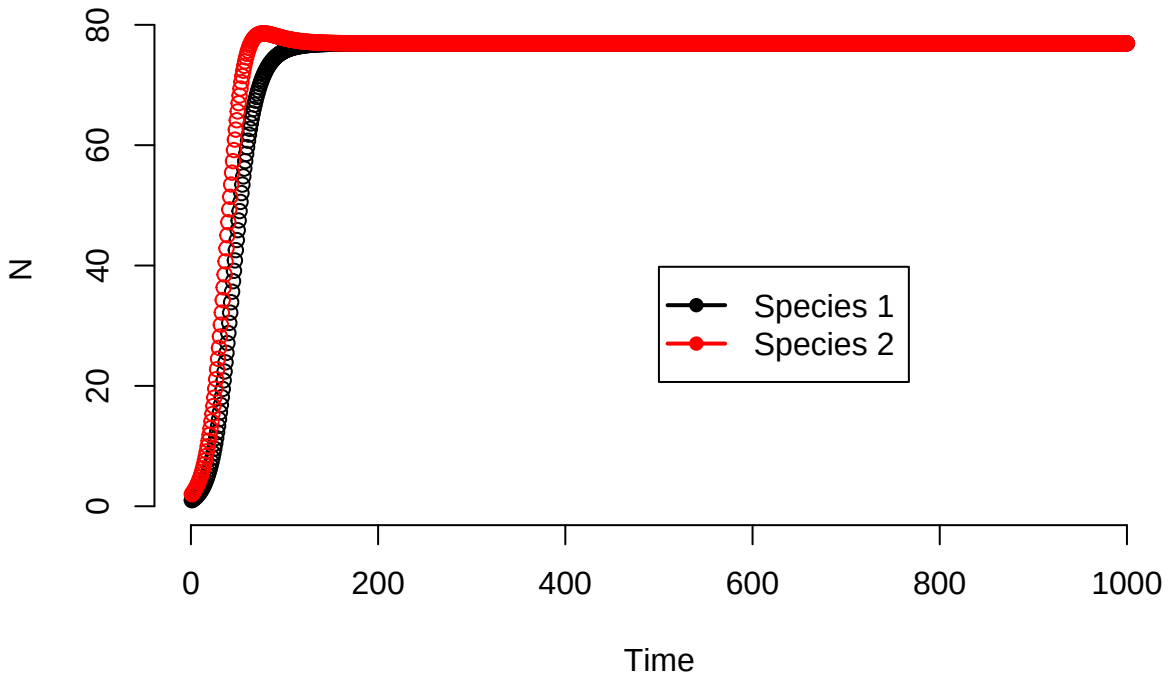
values for parameters and see which combinations lead to cases where both species have positive equilibrium population sizes.

You can do this with the following code that uses a loop to iterate the populations through time.

Remember to clear R's brain and set the working directory.

```
rm(list=ls())
setwd("~/Evolutionary Ecology/Lab 7")
```

```
#Set the parameters
r1=0.1
r2=0.1
K1=100
K2=100
alpha12=0.3
alpha21=0.3
N1.init=1
N2.init=2

#Set the number of time steps
time <- 1000

#Create an array to hold results; species x time
N <- array(NA,c(2,time))

#Put the initial values in the first slot of the array for each species
N[1,1] <- N1.init
N[2,1] <- N2.init

#Loop through different time points
for (t in 1:(time-1)){

  N[1,t+1] <- exp(r1 - r1/K1 * N[1,t] - r1/K1 * alpha12 * N[2,t]  ) * N[1,t]
  N[2,t+1] <- exp(r2 - r2/K2 * N[2,t] - r2/K2 * alpha21 * N[1,t]  ) * N[2,t]

}

#plot the results
par(mfrow = c(1, 1),bty='n')
plot(c(1:time),N[1,],ylim=c(0,max(N)),xlab='Time',ylab='N')
points(c(1:time),N[2,],col='red')
legend(time/2,mean(range(N)),c("Species 1","Species 2"),
       pch=c(16,16),col=c('black','red'),lwd=2)
```



That is one way to look at it. Remember that we also looked at this using phase planes, which have the species numbers on the y and x axes and lines that are the zero-growth isoclines. For the Ricker equation with two species, these two zero growth isoclines are

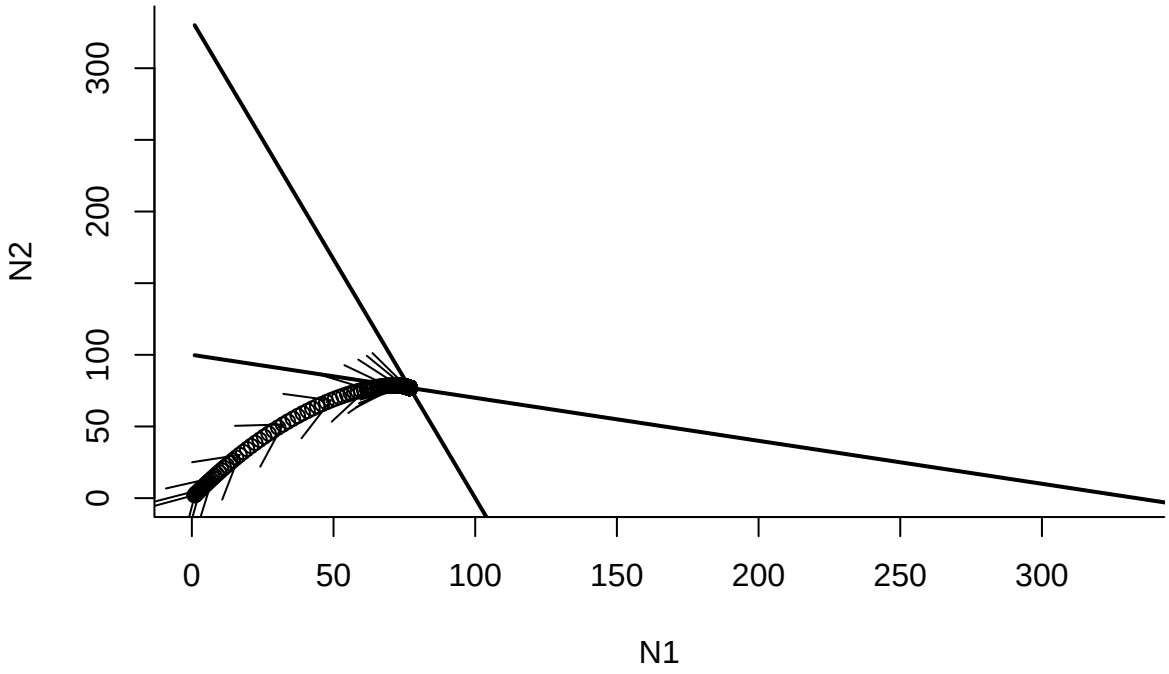
$$N_2 = \frac{K_1}{\alpha_{12}} - \frac{1}{\alpha_{12}} N_1; \; N_2 = K_2 - \alpha_{21} N_1 \tag{8.4}$$

when they are written with species 2 as the y-axis and species 1 as the x-axis.

You can then create the phase plane for the two species using this code.

```
#set the points to create the zero growth isoclines
N1 <- seq(1,1000,length.out=100)
N2 <- seq(1,1000,length.out=100)

#Equations for the zero-growth isoclines
sp1.null <- K1/alpha12 - 1/alpha12 * N1
sp2.null <- K2 - alpha21 * N1

#plot them
par(mfrow = c(1, 1),bty='L')
plot(N1,sp1.null,type='l',lwd=2,ylab='N2',xlab='N1',
```

```
     ylim=c(0,max(sp1.null,sp2.null)),xlim=c(0,max(sp1.null,sp2.null)))
lines(N1,sp2.null,type='l',lwd=2)

#This adds in the population sizes for each time step.
points(N[1,],N[2,],ylim=c(0,1.1*K2),xlim=c(0,1.1*K1))

#This makes arrows on the points to show which direction the populations
#are going in the phase plane.
s <- seq(1,c(time-10),10)
suppressWarnings( arrows(N[1,s], N[2,s], N[1,s + 1], N[2,s + 1]) )
```



Now in this case the arrows on the points are going towards the point where the lines cross, the two species equilibrium. Does that happen for all initial values of population sizes? One way you could do this is to copy and paste all of this code in successive iterations and build new plots each time. An easier way is to create a function that will do all of this for you.

Creating a function to explore different scenarios

Creating functions is fairly easy and it will save you lots of time and anguish with your code. A simple function can be written as:

```
myfunc <- function(x,z){
  y <- x + z
  return(y)
}
```

This creates a function called "myfunc". The function has arguments x and z that give the values to be used in the equation or code that is inside the function. To run the function and get an output, simply type:

```
myfunc(x=1, z=3)
```

```
## [1] 4
```

Now if you want to run the same set of code with different values of x and y, you can just type in different values for the arguments and do not need to rewrite or copy the code inside the function.

```
myfunc(x=3, z=10)
```

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

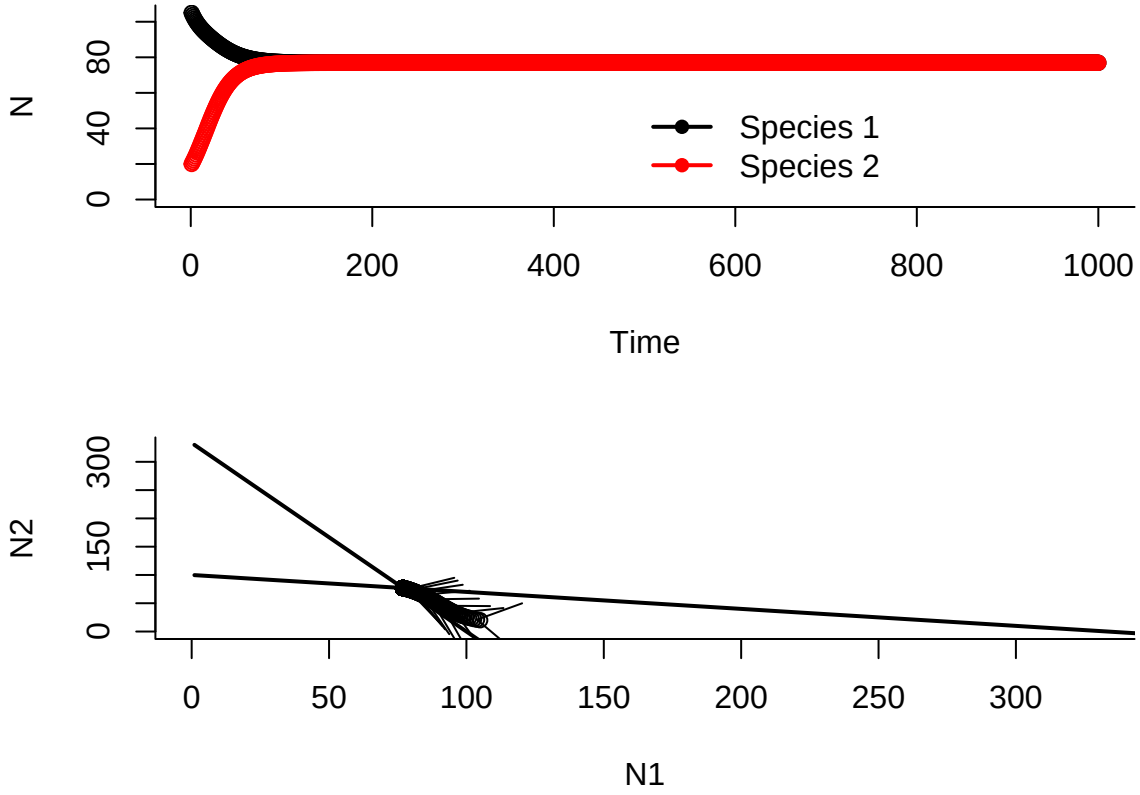
Let's do the same with the coexistence equations. The code below is the same as above with the values of the parameters and initial conditions removed.

```
#Call the function ricker or whatever you like
ricker <- function(r1,r2,K1,K2,alpha12,alpha21,N1.init,N2.init){

  time <- 1000
  N <- array(NA,c(2,time))

  N[1,1] <- N1.init
  N[2,1] <- N2.init

  for (t in 1:(time-1)){

    N[1,t+1] <- exp(r1 - r1/K1 * N[1,t] - r1/K1 * alpha12 * N[2,t]  ) * N[1,t]
    N[2,t+1] <- exp(r2 - r2/K2 * N[2,t] - r2/K2 * alpha21 * N[1,t]  ) * N[2,t]

  }

  par(mfrow = c(2, 1),bty='L',mar = c(5,6,1,1))
```

```
  plot(c(1:time),N[1,],ylim=c(0,max(N)),xlab='Time',ylab='N')
  points(c(1:time),N[2,],col='red')
  legend(time/2,mean(range(N)),c("Species 1","Species 2"),
         pch=c(16,16),col=c('black','red'),lwd=2,bty='n')


  N1 <- seq(1,1000,length.out=100)
  N2 <- seq(1,1000,length.out=100)

  sp1.null <- K1/alpha12 - 1/alpha12 * N1
  sp2.null <- K2 - alpha21 * N1

  plot(N1,sp1.null,type='l',lwd=2,ylab='N2',xlab='N1',
       ylim=c(0,max(sp1.null,sp2.null)),xlim=c(0,max(sp1.null,sp2.null)))

  lines(N1,sp2.null,type='l',lwd=2)

  points(N[1,],N[2,],ylim=c(0,1.1*K2),xlim=c(0,1.1*K1))

  s <- seq(1,c(time-10),10)
  suppressWarnings( arrows(N[1,s], N[2,s], N[1,s + 1], N[2,s + 1]) )


}
```

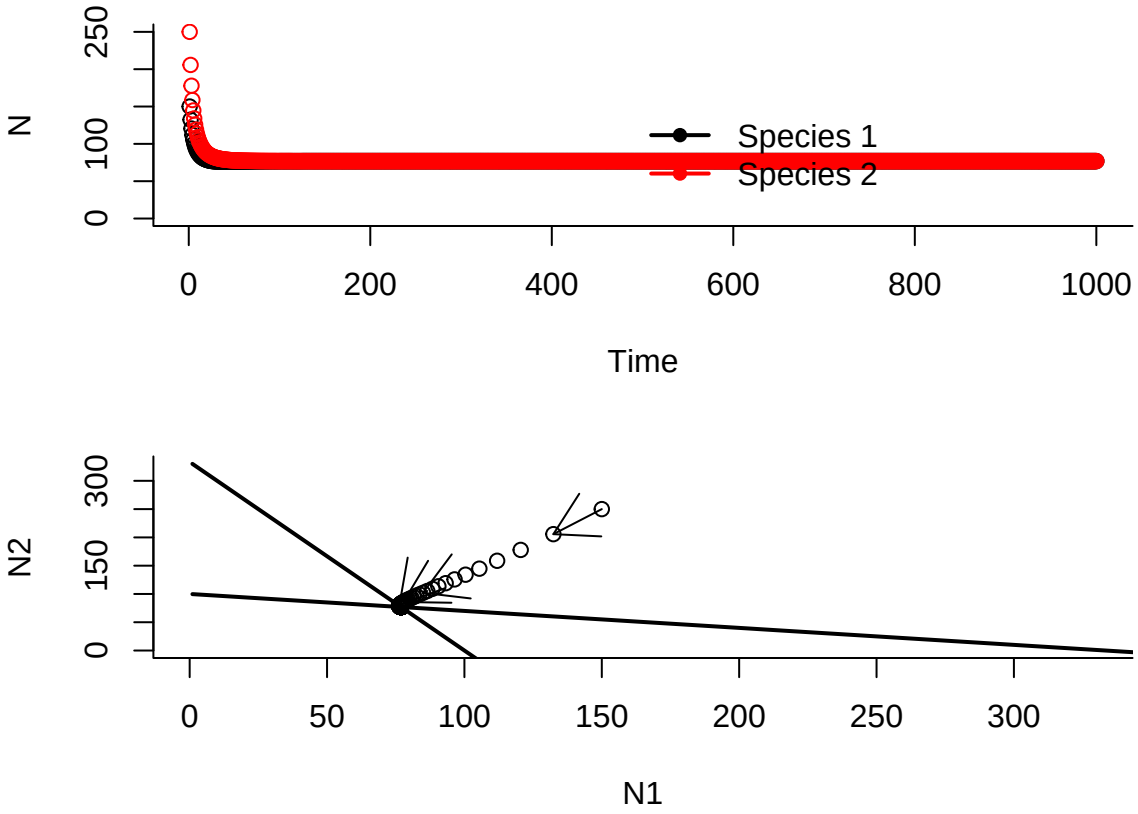
Now take it for a test drive to see what you get. Now all you need to do it to type the following:

```
ricker(r1=0.1,r2=0.1,K1=100,K2=100,alpha12=0.3,alpha21=0.3,
       N1.init=105,N2.init=20)
```



If you then want to try it with different parameters, then all you need to do is run this again. This one has the initial values of N1 and N2 much higher than before.

```
ricker(r1=0.1,r2=0.1,K1=100,K2=100,alpha12=0.3,alpha21=0.3,
       N1.init=150,N2.init=250)
```



Part I Exercises

1) Try running the function several more times. Each time change some other aspect of the parameters. For example, you can change the r's, the K's, the alpha's and the initial population sizes. What do you notice from this? Are there particular patterns that emerge?

8.2 Part II: Invasion Analysis

Sometimes it is not possible to derive the zero growth isoclines or they are not practical. This is especially true in populations with age or size structure. In these cases, there is an alternative based on the invasion criterion. We have seen this in lecture and in the reading. The essence of this approach is that if two species can coexist, then each species should be able to invade a community containing only the competitor at its single species equilibrium.

One advantage to this approach is that we can easily figure out many different values of the parameters that can lead to coexistence, competitive exclusion, or priority effects.

The first step in doing this is to find the single species equilibria, or the equilibrium population size of each species in a community without the competitor. You can do this by creating

the following function. Note that although this function runs the populations at the same time, the two species are non interacting.

```
ricker.eq <- function(r1,r2,K1,K2,N1.init,N2.init){

  #Number of time steps
  time <- 1000

  #Preallocate an array to store results
  N <- array(NA,c(2,time))

  #Set the initial number of each species in the community
  N[1,1] <- N1.init
  N[2,1] <- N2.init

  #Loop through times to get the equilibrium. In this case it is just K.
  for (t in 1:(time-1)){

    N[1,t+1] <- exp(r1 - r1/K1 * N[1,t] ) * N[1,t]
    N[2,t+1] <- exp(r2 - r2/K2 * N[2,t] ) * N[2,t]

  }

  N.eq <- N[,time]

  return(N.eq)
}
```

Now you can run the function using the following.

```
N.eq <- ricker.eq(r1=0.1,r2=0.1,K1=100,K2=100,N1.init=150,N2.init=250)
```

Now we need to make another function that asks a different question. In this case, we want to ask whether λ is greater than one for each species when it competes only with the other species. That is there are no individuals of the same species in the community. This may sound weird, but functionally we are asking whether the zero equilibrium of each species is stable or unstable. If it is unstable, λ is greater than 1 and the invader can grow. If λ is less than 1, it cannot grow.

In our example, we will examine how the values of the interaction coefficients α's influence the outcome. Remember that the interaction coefficients tell us something about how the two species are competing for resources. Therefore by examining different values of the interaction coefficients, we are asking which types of competition can lead to coexistence, competitive exclusion, and priority effects.

```
ricker.inv <- function(r1,r2,alpha12,alpha21,N.eq){

  #Preallocate an array to hold the results
  inv <- array(NA,c(2,length(alpha12),length(alpha21)))

  #Loop through alpha values for each species. Careful to get the 12's or
  #21's in the correct order.
  for (a12 in 1:length(alpha12)){
    for (a21 in 1:length(alpha21)){
      inv[1,a12,a21] <- exp(r1 - r1/N.eq[1] * alpha12[a12] * N.eq[2]  )
      inv[2,a12,a21] <- exp(r2 - r2/N.eq[2] * alpha21[a21] * N.eq[1]  )
    }
  }

  return(inv)
}
```

Then you can run this with the following code to produce a figure. To run this, you will first need to install the package "reshape2".

```
require(reshape2)
```

```
## Loading required package: reshape2
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

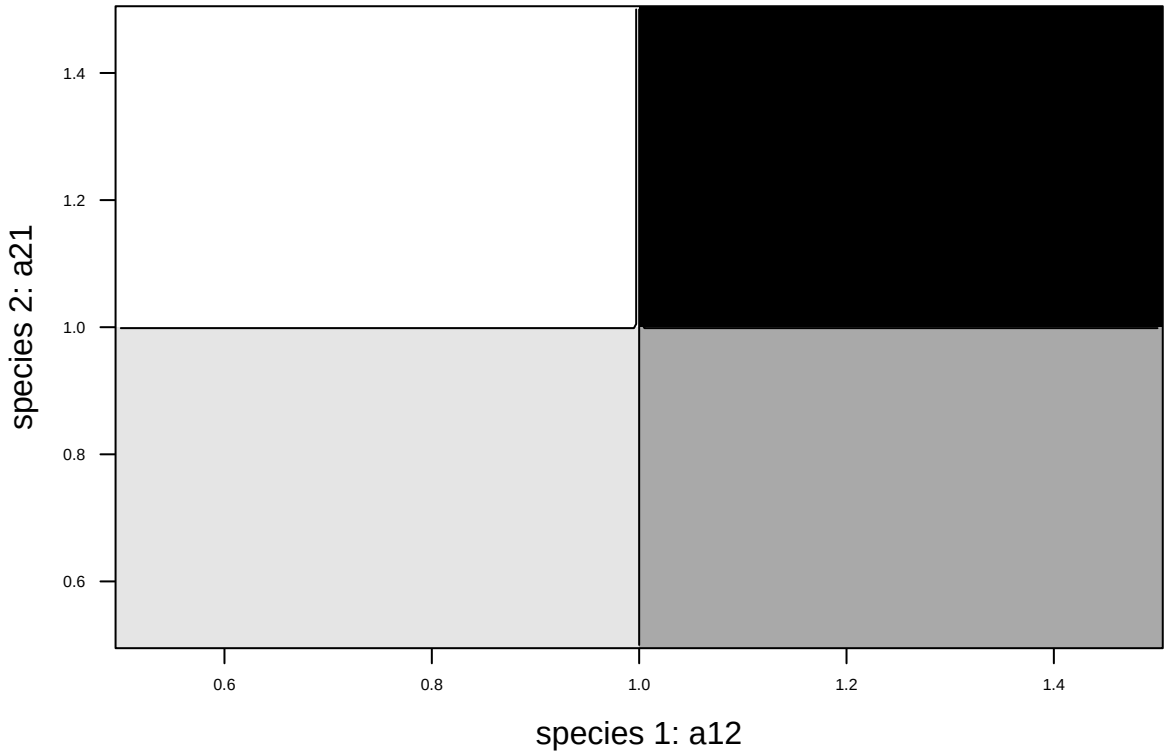
```
#This sets the values of the alphas you want to use
alphas <- seq(0.5,1.5,length.out=100)

#Run the function with those alpha values
inv <- ricker.inv(r1=0.1,r2=0.1,alpha12=alphas,alpha21=alphas,N.eq=N.eq)

#dimnames gives values to the dimensions of inv.
dimnames(inv) <- list(c(1,2),alphas,alphas)

#The function melt converts the array into a dateframe.
inv.df <- reshape2::melt(inv)
```

```r
#Names the columns of the data frame
names(inv.df) <- c('invader','a12','a21','value')

inv.df[which(inv.df$value>1),'lambda'] <- 10
inv.df[which(inv.df$value<1),'lambda'] <- -10
inv.df[which(round(inv.df$value,10)==1),'lambda'] <- 0

inv.1 <- inv.df[which(inv.df$invader=='1'),]
inv.2 <- inv.df[which(inv.df$invader=='2'),]

inv.1$lambda1 <- inv.1$lambda
inv.2$lambda2 <- inv.2$lambda

reinvade <- merge(x=inv.1[,c('a12','a21','lambda1')],
                  y=inv.2[,c('a12','a21','lambda2')],by=c('a12','a21'))

#This sets the conditions for each outcome. Ultimately, they will get different
#colors in the figure.
reinvade[which(reinvade$lambda1>0 & reinvade$lambda2>0),'coexist'] <- 0
reinvade[which(reinvade$lambda1>=0 & reinvade$lambda2<0),'coexist'] <- -1
reinvade[which(reinvade$lambda1<0 & reinvade$lambda2>=0),'coexist'] <- 1

#if both are negative then it is the species that is there first
#that has priority
reinvade[which(reinvade$lambda1<0 & reinvade$lambda2<0),'coexist'] <- 2
```

```r
require(lattice)
```

```
## Loading required package: lattice
```

```r
levelplot(reinvade[,c('coexist')]~reinvade[,c('a12')]*reinvade[,c('a21')],
          colorkey=FALSE,
          contour=TRUE,
          col.regions=c('white','grey90','darkgrey','black'),cuts=3,
          scales = list(tck = c(1,0),cex=0.5),
          xlab='species 1: a12',ylab='species 2: a21')
```



Each color in the figure represents a different outcome of competition. Areas of the plot that are black mean that those combinations of α's (or competitive interactions) lead to priority effects, where each species can resist invasion by the other species. The medium grey areas of the plot (lower left) are combinations of *alpha*'s that lead to species coexistence. For this example, the values are less than one, which occurs when there is separation of resource use among the two species.

The dark grey section are then competitive interactions (α's) where species 2 outcompetes species 1. Likewise the light grey areas are competitive interactions (α's) where species 1 outcompetes species 2.

Part II Exercises

1) Try going through and changing the parameters to see how this changes your predictions. Note that you do not need to change the alphas any more. This is done as part of the analysis. But you can change the K's and r's.

2) You likely noticed that changing the K's leads to different values of the alpha's that lead to coexistence, competitive exclusion, and priority effects. What might be a biological explanation for this?

8.3 Part III: When Invasion Analysis Fails

Invasion analysis works about 95% of the time. There are situations where it does not work. One important situation is when one of the species exhibits an Allee effect. Remember that Allee effects are positive density-dependence over some range of densities. Let's take a look at how this influences the how coexistence operates.

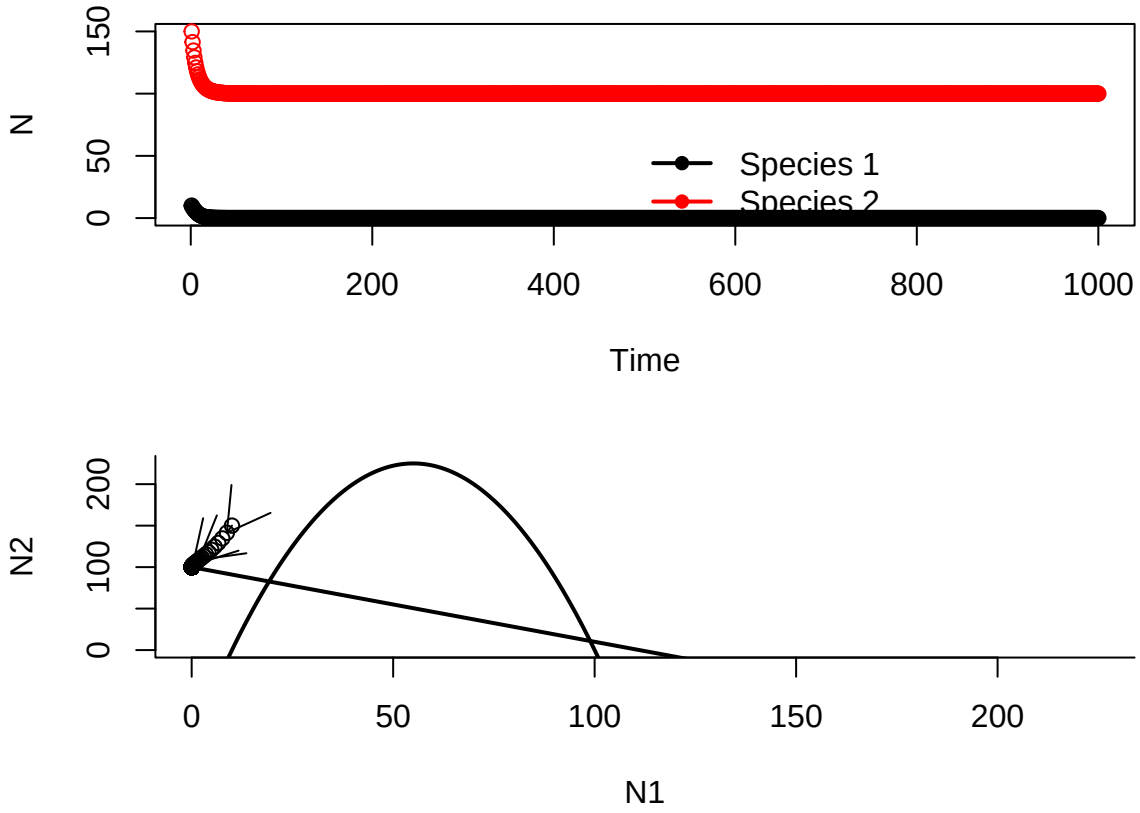
First, let's create a function called ricker.allee. This function is similar to the ricker function you created in part I, so you can copy that code and make a few small changes. Namely, you need to change the equation for species 1 in the time loop so that it includes the Allee effect and you need to change the equation for sp1.null to the values below.

```
ricker.allee <- function(r1,r2,K1,K2,A,alpha12,alpha21,N1.init,N2.init){
  time <- 1000
  N <- array(NA,c(2,time))

  N[1,1] <- N1.init
  N[2,1] <- N2.init

  for (t in 1:(time-1)){

    N[1,t+1] <- exp(r1 * (1 - N[1,t] / K1) * (N[1,t] / A - 1) -
                      r1/K1 * alpha12 * N[2,t] ) * N[1,t]
    N[2,t+1] <- exp(r2 - r2/K2 * N[2,t] - r2/K2 * alpha21 * N[1,t]  ) * N[2,t]

  }

  par(mfrow = c(2, 1),mar = c(5,6,1,1))

  plot(c(1:time),N[1,],ylim=range(N),xlab='Time',ylab='N')
  points(c(1:time),N[2,],col='red')
  legend(time/2,mean(range(N)),c("Species 1","Species 2"), pch=c(16,16),
         col=c('black','red'),lwd=2,bty='n')


  N1 <- seq(1,1000,length.out=1000)
  N2 <- seq(1,1000,length.out=1000)

  sp1.null <- K1/alpha12 * (N1/A - 1) -
              (N1^2) / (A * alpha12) + 1 / alpha12 * N1
  sp2.null <- K2 - alpha21 * N1

  plot(N1,sp1.null,type='l',bty='L',lwd=2,ylab='N2',xlab='N1',
       ylim=c(0,max(sp1.null,sp2.null)),xlim=c(0,max(sp1.null,sp2.null)))
```

```
  lines(N1,sp2.null,type='l',bty='L',lwd=2,ylab='N2',xlab='N1')

  points(N[1,],N[2,],ylim=c(0,1.1*K2),xlim=c(0,1.1*K1))

  s <- seq(1,c(time-10),10)
  suppressWarnings(arrows(N[1,s], N[2,s], N[1,s + 1], N[2,s + 1]) )

}
```

Then we can run it with the following parameters.

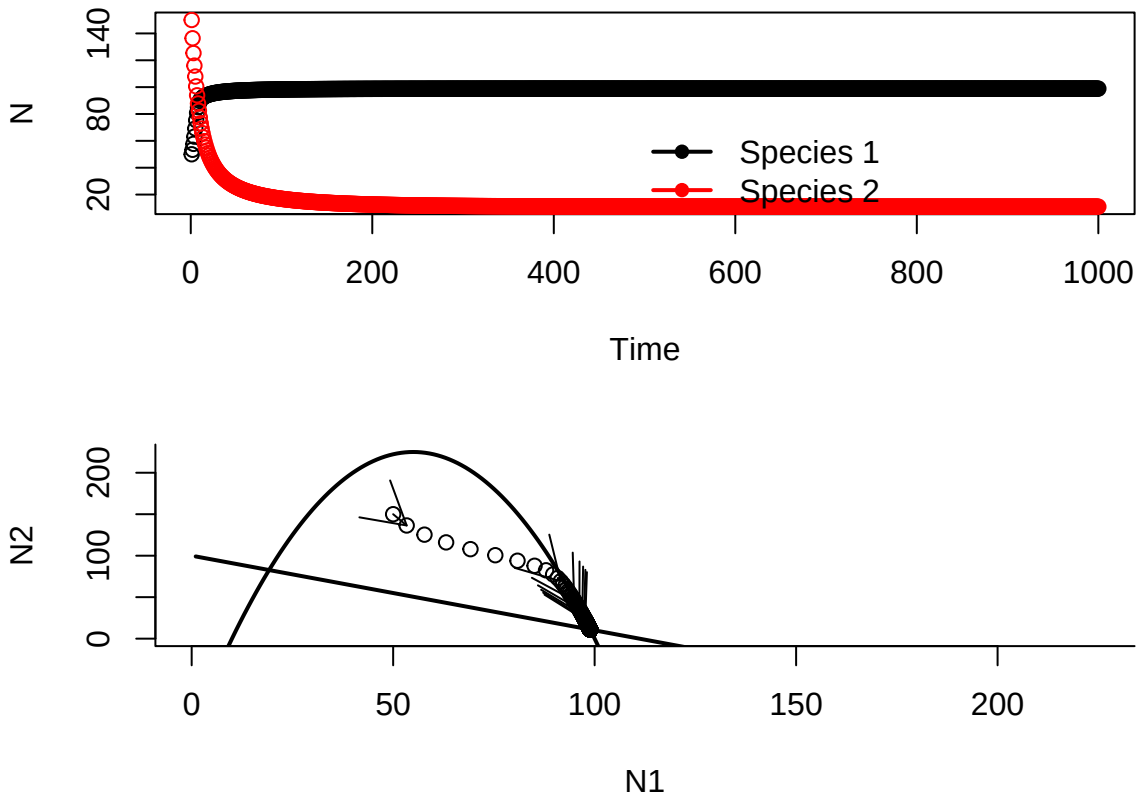
```
ricker.allee(r1=0.1,r2=0.1,K1=100,K2=100,A=10,alpha12=0.9,
             alpha21=0.9,N1.init=10,N2.init=150)
```



In this example, the values of the interaction coefficients are such that the two species should coexist, yet we predict that they do not given the initial starting population sizes. One action of Allee effects is to bend the zero growth isocline of species 1. The rules for null clines are the same here as before–starting above the isocline leads to a decline in the population size. Thus there is a threshold value that the population size of species 1 must be above in order to invade a community of species 2.

Let's try a different initial value for species 1, say 50.

```
ricker.allee(r1=0.1,r2=0.1,K1=100,K2=100,A=10,alpha12=0.9,
             alpha21=0.9,N1.init=50,N2.init=150)
```



So above the threshold, the two species are predicted to coexist.

Part III Exercises

1) Unlike the phase planes without Allee effects, the phase plane with Allee effects has five areas of the plot where we could draw our arrows that represent the changes in population sizes in those sections. Rerun the analysis above with starting population sizes in each of those sections. How many of the five sections have arrows that will lead to coexistence?

8.4 Part IV: Empirical Tests

One way to test whether species can coexist is to use an experiment where the densities of both species are manipulated in a factorial fashion. If the experiment is a two by two factorial

this means to take all the combinations of two densities of each species. This experiment can be done in nature, mesocosms, or a lab setting. It is often difficult to do such experiments in nature and the lab is a bit artifical to ask questions of this type. Mesocosms provide a happy medium ground.

Let's take a look at some experimental results from a mesocosm experiment of guppies and killifish. In this experiment, there were 28 total mesocosms. Each mesocosm had one of the following combinations of guppies and killifish: 8 guppies, 6 killifish; 16 guppies, 6 killifish; 8 guppies, 12 killifish. This means there were three total combinations of guppies and killifish. This is not quite a factorial experiment, it lacks the treatment with 16 guppies and 12 killifish. Thus this is not quite a factorial experiment and is known as a fractional factorial.

Get the data.

```
lamb <- read.csv(file = "./Data/sppcomp.csv",colClasses = "character")
```

Let's take a look at the file

```
head(lamb)
```

```
##   X   spp channel block replicate              lamb           g.dens k.dens
## 1 1 guppy       1     1         1 0.444444444444444 5.33333333333333      4
## 2 2 guppy       1     1         2 0.690476190476191 5.33333333333333      4
## 3 3 guppy      10     2         1 0.523809523809524 5.33333333333333      4
## 4 4 guppy      10     2         2  1.35714285714286 5.33333333333333      4
## 5 5 guppy      11     2         1 0.209183673469388 10.6666666666667      4
## 6 6 guppy      11     2         2 0.403061224489796 10.6666666666667      4
```

The first column is the species for which we are looking at their response. Channel, block, and replicate identify the replicate of the treatment. The column lamb is the λ value for that mesocosm. This will be the dependent variable in our analyses. The last two are the densities (number / area) of guppies and killifish, respectively. One little thing we need to do is change lamb, and the density variable to numeric values in R.

```
lamb$lamb <- as.numeric(lamb$lamb)
lamb$g.dens <- as.numeric(lamb$g.dens)
lamb$k.dens <- as.numeric(lamb$k.dens)
```

Now let's run two analyses. These are similar to our regressions that we used before, but they also have some random effects that account for spatial heterogeneity between groups of mesocosms (i.e. block). One of the analyses is for the guppies and the other is for the killifish. You will need to load the R package *lme4* to run this.

```
require(lme4)
```

```
## Loading required package: lme4
```

```
lamb$log.lamb <- log(lamb$lamb)

g.mod <- lmer(log.lamb ~ 1  + g.dens + k.dens + (1|block) +
                (1|replicate),data=lamb[which(lamb$spp=='guppy'),] )
summary(g.mod)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log.lamb ~ 1 + g.dens + k.dens + (1 | block) + (1 | replicate)
##    Data: lamb[which(lamb$spp == "guppy"), ]
##
## REML criterion at convergence: 17.1
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.56896 -0.43130  0.02304  0.30331  2.38410
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  block     (Intercept) 0.01296  0.1139
##  replicate (Intercept) 0.24186  0.4918
##  Residual              0.05574  0.2361
## Number of obs: 28, groups:  block, 2; replicate, 2
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  0.798576   0.427962   1.866
## g.dens      -0.201847   0.020205  -9.990
## k.dens       0.000871   0.026940   0.032
##
## Correlation of Fixed Effects:
##        (Intr) g.dens
## g.dens -0.453
## k.dens -0.453  0.400
```

```
k.mod <- lmer(log.lamb ~ 1  + g.dens + k.dens + (1|block) +
                (1|replicate),data=lamb[which(lamb$spp=='killifish'),] )
summary(k.mod)
```

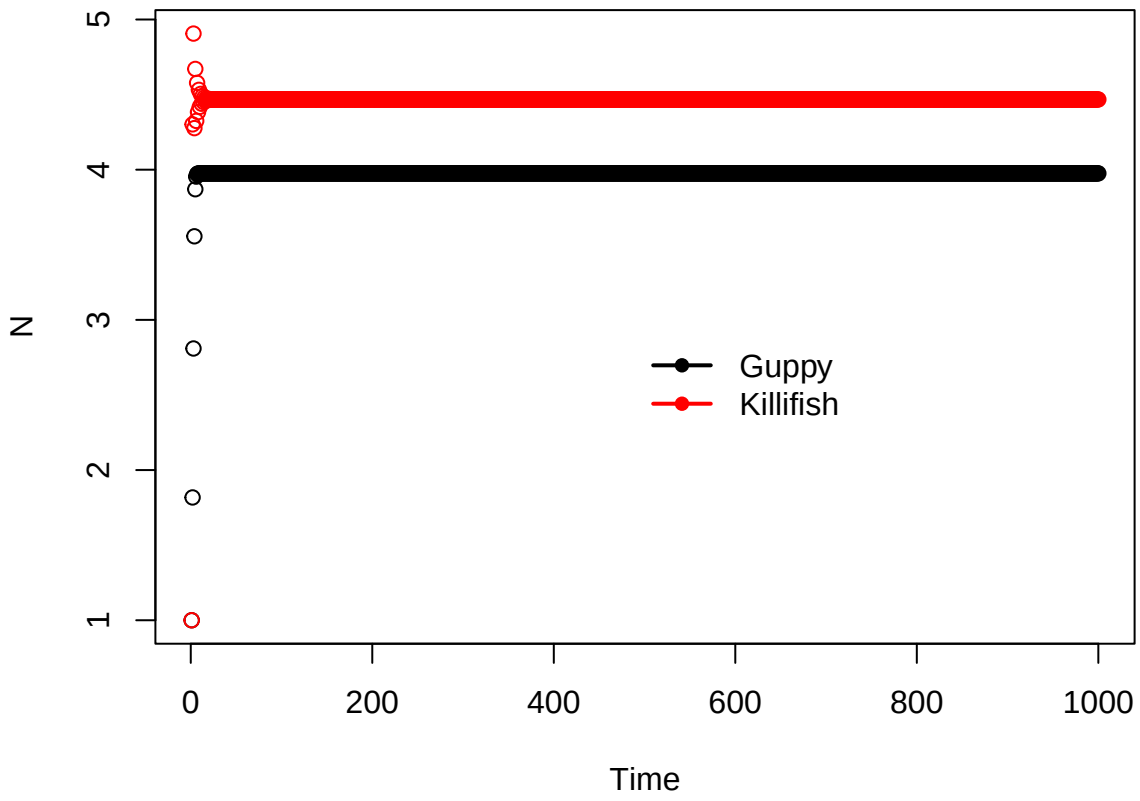
```
## Linear mixed model fit by REML ['lmerMod']
## Formula: log.lamb ~ 1 + g.dens + k.dens + (1 | block) + (1 | replicate)
##    Data: lamb[which(lamb$spp == "killifish"), ]
##
## REML criterion at convergence: 56.1
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.7655 -0.7104  0.2812  0.5469  2.2179
##
## Random effects:
##  Groups    Name        Variance Std.Dev.
##  block     (Intercept) 0.144413 0.38002
##  replicate (Intercept) 0.006518 0.08074
##  Residual              0.301623 0.54920
## Number of obs: 28, groups:  block, 2; replicate, 2
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  1.88416    0.61408   3.068
## g.dens      -0.03062    0.04700  -0.651
## k.dens      -0.39449    0.06267  -6.295
##
## Correlation of Fixed Effects:
##        (Intr) g.dens
## g.dens -0.735
## k.dens -0.735  0.400
```

Given these effects of density of the two species on each of the two species, do you think they should coexist?

Let's use the parameters from the model to project the population sizes forward in time to find out. The code below allows the two species to interact.

```
g.parm <- coef(summary(g.mod))
k.parm <- coef(summary(k.mod))

time <- 1000
N <- array(NA,c(2,time))

N[1,1] <- 1
N[2,1] <- 1

for (t in 1:(time-1)){
```

```
  N[1,t+1] <- exp(g.parm['(Intercept)','Estimate'] +
                    g.parm['g.dens','Estimate'] * N[1,t] +
                    g.parm['k.dens','Estimate'] * N[2,t]  ) * N[1,t]
  N[2,t+1] <- exp(k.parm['(Intercept)','Estimate'] +
                    k.parm['k.dens','Estimate'] * N[2,t] +
                    k.parm['g.dens','Estimate'] * N[1,t]  ) * N[2,t]

}

par(mfrow = c(1, 1),mar = c(5,6,1,1))

plot(c(1:time),N[1,],ylim=range(N),xlab='Time',ylab='N')
points(c(1:time),N[2,],col='red')
legend(time/2,mean(range(N)),c("Guppy","Killifish"),
       pch=c(16,16),col=c('black','red'),lwd=2,bty='n')
```



Looks like they can because each has a positive population size in when living with the other species. But, this analysis only considers one starting condition. In the exercises below, you will look at other conditions.

Part IV Exercises

1) Use the fitted equations for each species to calculate the zero growth isocline for each species. Use the code in part 1 to then plot the zero growth isoclines and check that the two species equilibrium is stable.

2) Conduct an invasion analysis using the estimated parameters. Does the invasion analysis agree with the zero growth isocline approach?