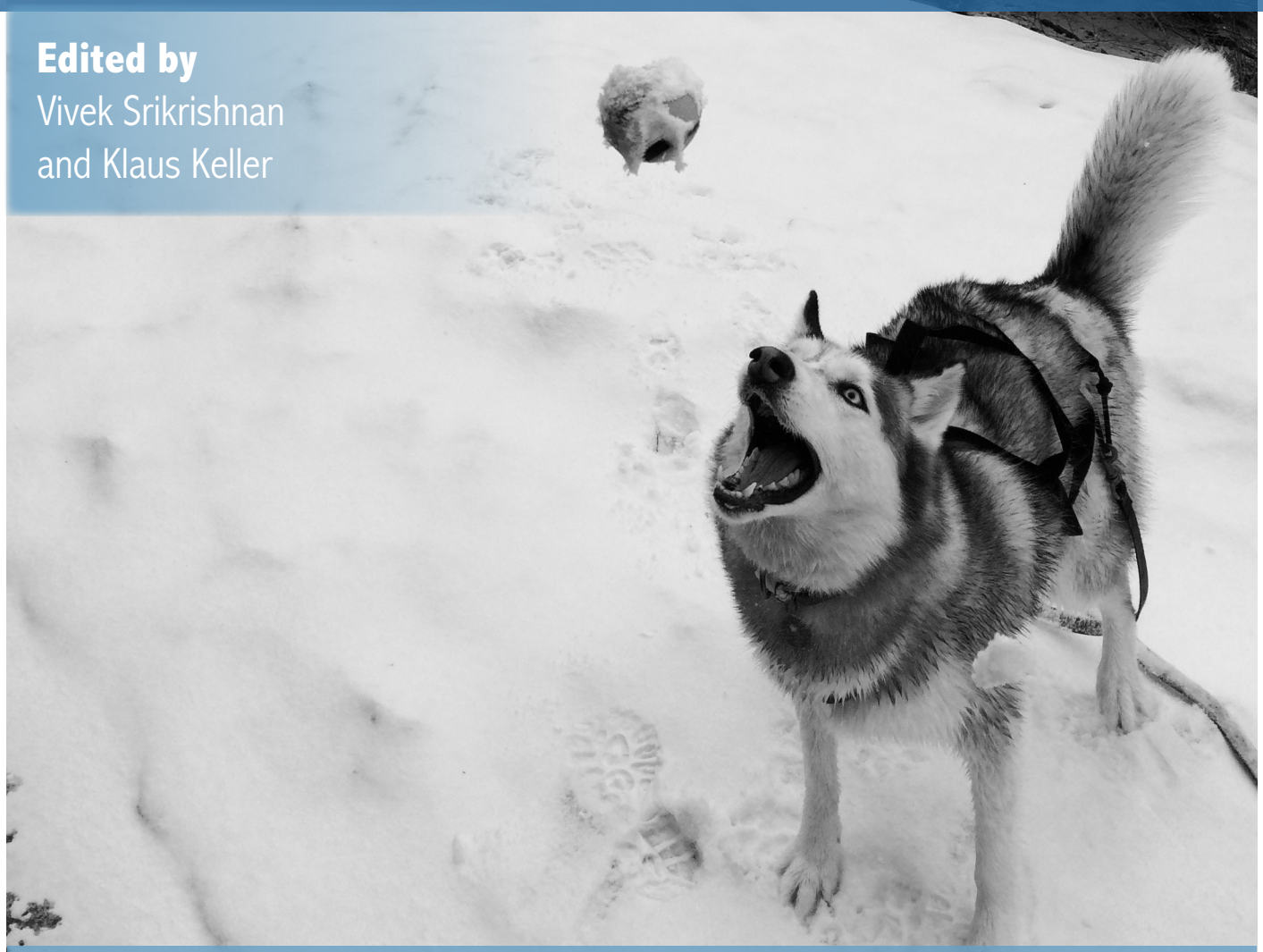
SCRiM

Risk Analysis in the Earth Sciences: A Lab Manual with Exercises in R

3rd edition

Edited by
Vivek Srikrishnan
and Klaus Keller



With contributions from Patrick J. Applegate, Vivek Srikrishnan, Tony E. Wong, Ryan L. Sriver, Alexander Bakker, Kelsey L. Ruckert, and Klaus Keller

Risk Analysis in the Earth Sciences

Vivek Srikrishnan[1] Klaus Keller[2]

2021-12-05

[1]Department of Biological & Environmental Engineering, Cornell University, viveks@cornell.edu
[2]Thayer School of Engineering, Dartmouth University, klaus.keller@dartmouth.edu

Contents

Preface

Greenhouse gas emissions have caused considerable changes in climate, including increased surface air temperatures and rising sea levels. Rising sea levels increase the risks of flooding for people living near the world's coastlines. Managing such risks requires an understanding of many fields, including Earth science, statistics, and economics. At the same time, the free, open-source programming environment Julia is growing in popularity among statisticians and scientists due to its flexibility and graphics capabilities, as well as its large collection of existing software libraries.

This textbook presents a series of exercises in Julia that teach the Earth science and statistical concepts needed for assessing climate-related risks. These exercises are intended for upper-level undergraduates, beginning graduate students, and professionals in other areas who wish to gain insight into academic climate risk analysis.

About This Book

This textbook presents a series of chapters that introduce students to the statistical aspects of risk analysis in the Earth sciences using Julia. Most of these chapters have to do with sea-level rise, which is a major focus of the authors' research activities.

The exercises are intended for upper-level undergraduates, beginning graduate students, and professionals in other areas who wish to gain insight into academic climate risk analysis. Previous programming experience is helpful, but not required; the first exercise explains how to learn the basics of Julia.

Each chapter begins with a description of the Earth science and/or statistical concepts that the chapter teaches. Next, each chapter presents a detailed explanation of a piece of existing Julia code. Finally, students are asked to either modify the existing code or write their own code to produce a well-defined outcome. Each chapter also includes questions to encourage classroom discussion and reflection on the part of individual students.

A few chapters contain appendices that present additional discussion of the topics raised in the preceding parts of those chapters. These appendices are intended for graduate students or researchers who have additional background in programming or the Earth sciences. They can be skipped by other readers with no loss in comprehension.

The book is designed to be modular. The first eight chapters (0-7) were written by Patrick Applegate, with contributions by others as noted in the individual chapters. Chapter 8 was written by Vivek Srikrishnan, and Chapter 9 by Tony Wong. Other chapters may be added by members of the Keller research group in the future.

Warning

Completing the chapters in this textbook will not teach you to perform publication-quality or consulting-grade risk analyses. This textbook is intended primarily for advanced undergraduates, and the material presented here should only be applied to "real" problems after additional training. The authors and editors of this textbook specifically disclaim any liability for damages associated with the use of the material presented in this textbook.

Contributor Bios

Vivek Srikrishnan is an Assistant Professor in the Department of Biological & Environmental Engineering at Cornell University. He studies decision-making under uncertain climate risks and feedbacks between natural and human systems. His research combines model development, uncertainty quantification, complex systems dynamics, and decision-making under uncertainty. He has a Ph.D. in Energy & Mineral Engineering from Penn State. Vivek's email address is viveks@cornell.edu, and his website is http://www.viveks.me.

Patrick J. Applegate is an Academic Editor at Research Square. In this role, Patrick helps researchers communicate their science and works on data science projects in Python. Patrick earned his Ph.D. in Geosciences at Penn State and is first author or co-author of 20 peer-reviewed scientific papers. His research treats topics including ice sheets and their contributions to sea level rise, methods for estimating the ages of glacial deposits, and the application of statistical methods to problems in the geosciences.

Tony E. Wong is an Assistant Professor in the School of Mathematical & Statistical Sciences at the Rochester Institute of Technology. He received his PhD in Applied Mathematics from CU Boulder in 2016, and later was a Postdoctoral Scholar at Penn State's Earth and Environmental Systems Institute. Prior to joining RIT, Tony returned to CU as an Instructor in the Department of Computer Science to pursue his love of teaching. His current research focuses on uncertainty quantification and model calibration in sea level and coastal storm surge projections. Tony's email address is aeswma@rit.edu and his website is https://tonyewong.github.io.

Ryan L. Sriver is an Associate Professor of Atmospheric Sciences at the University of Illinois at Urbana-Champaign (UIUC). Prior to joining UIUC in the fall of 2012, he worked as a research associate in Penn State's Department of Geosciences and as a NOAA Climate and Global Change postdoctoral fellow in Penn State's Department of Meteorology. He graduated from Purdue University with a Ph.D. in Earth and Atmospheric Sciences. His research seeks to develop a deeper understanding about the physical processes influencing variability within Earth's climate system, and to quantify climate change uncertainties

relevant to adaptation planning and decision-making. Ryan's email address is rsriver@illinois.edu, and his website is at https://www.atmos.illinois.edu/~rsriver/index.html.

Alexander Bakker is a senior advisor at Rijkswaterstraat. He received a Masters in Civil Engineering from the Delft University of Technology and obtained a PhD in Regional climate change scenarios at VU University of Amsterdam.

Kelsey L. Ruckert is a geologist by training. She is a Coastal Extension Specialist for the Earth and Environmental Systems Institute at Penn State. She specializes in geology, flood hazards, water quality, and climate change impacts where she excels in GIS, statistical analysis, modeling, and fieldwork. Her research focuses on understanding sea-level rise, storm surge, and water resources, and their implications to vulnerability and the design of risk management strategies. Her email address is klr324@psu.edu and her website is http://klr324.github.io.

Klaus Keller is a Professor of Geosciences at Penn State and a Visiting Professor at the Thayer School of Engineering at Dartmouth University. Keller directs the Center for Climate Risk Management as well as the research network for Sustainable Climate Risk Management. Before joining Penn State, he worked as a research scientist and lecturer at Princeton University and as an engineer in Germany. Professor Keller graduated from Princeton with a Ph.D. in civil and environmental engineering. He received master's degrees from M.I.T. and Princeton as well as an engineer's degree from the Technische Universität Berlin. His research addresses two interrelated questions. First, how can we mechanistically understand past and potentially predict future changes in the climate system? Second, how can we use this information to design sustainable, scientifically sound, technologically feasible, economically efficient, and ethically defensible climate risk management strategies? He analyzes these questions by mission-oriented basic research covering a wide range of disciplines such as Earth system science, economics, engineering, philosophy, decision science, and statistics. Klaus' email address is klaus@psu.edu, and website is at http://www3.geosc.psu.edu/~kzk10/.

Acknowledgements

This work was supported by the National Science Foundation through the Network for Sustainable Climate Risk Management (SCRiM) under NSF cooperative agreement GEO-1240507. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. Other support was provided by the Center for Climate Risk Management and the Rock Ethics Institute.

Most of these exercises were originally developed as part of a course titled Risk Analysis in the Earth Sciences, developed and taught by Klaus Keller in the Department of Geosciences

at the Pennsylvania State University. The exercises discussed in this lab manual were co-developed by a large group (see the individual chapters).

Katerina Kostadinova designed the cover using a photograph provided by Frank Erickson.

Randy Miller created the Github repository.

Thomas Patrick Walsh and Naomi Radke made suggestions for improving the text and lab exercises.

Vivek Srikrishnan thanks his colleagues in the Keller research group and beyond for being willing and able to work on this textbook. Special thanks go to Patrick Applegate for leading the development of an excellent first edition of this book.

Klaus Keller thanks all the students from the Risk Analysis course, the members of the Keller research group, and his SCRiM collaborators for invaluable inputs. Special thanks go to his family, many mentors, and friends for support.

Book Source and Scripts

As noted above, most of the exercises presented in this e-textbook ask students to modify existing R scripts. These R scripts can be downloaded as a `.zip` file from https://www.scrimhub.org/raes.

The source files for the book are stored in a Github repository at https://github.com/scrim-network/raes.

Comments and Questions

We'd like to make *Risk Analysis in the Earth Sciences* as useful as possible. If you have a comment about the book or a question about one of the chapters, please post an issue to the Github repository mentioned above. If we make a change to the book based on your comments, we will be sure to add your name to the Acknowledgements section in future versions.

Chapter 1

Introduction

This chapter was written by Patrick J. Applegate and Klaus Keller. It briefly discusses climate change, uncertainty, and risk.

1.1 Climate Risks

We begin with a very brief discussion of some key findings of climate science and the problems posed by climate change. There are many useful reviews of these subjects, in both textbook and popular formats [Kump et al., 2009, Alley, 2011]. For high-level discussions of these topics, see the Intergovernmental Panel on Climate Change's recent reports, particularly the Technical Summaries of the IPCC's Working Groups I and II [Stocker et al., 2013, Field et al., 2013]. We particularly recommend that you read chapters 1, 3, 6, 15, and 16 in Kump et al. [2009] before working on the exercises in this e-textbook.

Most scientists who study the climate system agree that human activities are causing surface air temperatures to increase [Oreskes, 2007]. Combustion of fossil fuels and changes in land use lead to increases in the concentrations of carbon dioxide, methane, and nitrous oxide in the atmosphere. The increased concentrations of these gases trap heat near the earth's surface that would normally escape to outer space. Temperatures have risen measurably over the last century and will probably continue to increase in the future [IPCC, 2013, Collins et al., 2013].

This increase in surface air temperatures affects other parts of the Earth system, notably sea level. Rising surface air temperatures cause the water near the ocean's surface to expand, and also lead to increased rates of melting from land ice. Global mean sea level has risen by a few tens of centimeters over the past 1-2 centuries, and could go up by more than one meter by 2100 [Parris et al., 2012, IPCC, 2013, Church et al., 2013].

Increases in sea level create risks for people living near present-day sea level. Sea level at a particular point along the world's coastline rises and falls over time scales of hours to days, in response to tides and storms. Stacked on top of the long-term rise in sea level, these short-period oscillations flood normally-dry land with increased frequency [e.g. Spanger-Siegfried et al., 2014].

Fossil fuel consumption also creates benefits for people, although these benefits are unevenly distributed. Countries with low per capita gross domestic products also tend to emit very little carbon dioxide per person. Among countries with high per capita gross domestic products, some emit large amounts of carbon dioxide per person each year, whereas others emit comparatively little. However, achieving good economic outcomes in today's energy economy seems to require at least some fossil fuel consumption. Moreover, the benefits of fossil fuel consumption occur now, whereas the negative effects of climate change will take place primarily in the future.

Thus, reducing greenhouse gas emissions creates costs in terms of lost economic productivity. Modern societies require large amounts of energy to manufacture goods, produce food, and move people from place to place. Today, most of that energy comes from fossil fuels. Reducing fossil fuel consumption likely means cuts to manufacturing, agriculture, or transportation, at least in the short term.

1.2 Some Questions

The discussion in Chapter 1.1 leads to several interrelated questions:

1. How much carbon dioxide, and other greenhouse gases, will human societies emit in the future [e.g. Sanford et al., 2014]?
2. How much will surface air temperatures rise in response to past and future greenhouse gas emissions?
3. How much will sea level rise as a consequence of surface air temperature increases?
4. What effects will increases in temperature and sea level, as well as changes in precipitation and other climatic variables, have on people?
5. Given the benefits associated with fossil fuel consumption and the risks presented by climate change, what is the appropriate balance between economic growth, cutting greenhouse gas emissions, and building protective measures to moderate the negative effects of climate change?
6. In choosing a strategy for addressing climate change, how do we take account of values like fairness? For example, how do we ensure that the negative impacts of climate change do not fall disproportionately on people who have not contributed to the problem?

Answering these questions requires contributions from several disciplines, including the

social sciences (especially economics; questions 1, 4, and 5), meteorology (2), oceanography (3), glaciology (3), and philosophy (6). It also requires estimating future changes, often with computer models.

Statistics also plays an important role in answering these questions, because our answers always contain some level of uncertainty. Future changes in temperature and sea level depend on emissions (question 1), which are difficult to predict. Moreover, climate projections are made with imperfect computer models that depend on input parameters whose values we aren't sure of. When we make an estimate of future changes using a computer model, how sure should we be about the answer? How can we report our answers in a way that accurately reflects the state of our knowledge? Statistics is the discipline that provides answers to these questions.

1.3 Types of (Un)certainty

Suppose that we wish to characterize the state of our knowledge about a future event whose outcomes can take values on the real number line. Depending on the event in question, this quantity can be *certain*, *uncertain*, or *deeply uncertain*. These "levels of uncertainty" [Riesch, 2013] have different mathematical representations:

- If the outcome is certain, it can take just one value.
- If the outcome is uncertain, but not deeply uncertain, there is a well-defined and generally agreed-upon probability density function that relates different potential outcomes to the probability that they will occur (see Exercise 3 for a discussion of probability density functions). For example, rolling a die has six possible outcomes (the integers 1, 2,… 6), and each of these outcomes is equally likely to occur if the die is fair.
- If the outcome is deeply uncertain, there is no single, agreed-upon probability density function describing the chance of observing different outcomes.

Many important questions in climate science involve deep uncertainty. Published estimates of the climate sensitivity provide perhaps the clearest demonstration of this point. Climate sensitivity is the change in global mean temperature that would result if carbon dioxide concentrations in the atmosphere were to double. Many studies have estimated this quantity. Though these studies have obtained overlapping answers, their probability density functions differ [Collins et al., 2013, their Fig. 1 in Box 12.2; Meehl et al., 2007, their Fig. 9.20].

1.4 What is Risk?

Answering many important questions requires careful consideration of *risk*. Risk is the harm associated with a particular bad outcome, multiplied by the chance of that bad

outcome occurring. For example, answering questions 5 and 6, above, requires considering the harms associated with future climate change, given that we are not sure how severe it will be (questions 1-4). A possible future event can represent an important risk if it is relatively benign, but very likely to occur, or if it is very unlikely, but extremely damaging if it does happen.

Classic risk analysis typically assumes that the probability density functions relating outcomes to the chances of their occuring are well-known. However, as noted above, many questions in climate science have deeply uncertain answers. In these cases, other methods have to be used to assess the merits of different approaches for managing climate risk. One such approach is robust decision making, or RDM [Sriver et al., 2017, Hadka et al., 2015], which focuses on identifying the weaknesses of particular systems and then identifying solutions which produce good results over a wide range of potential outcomes.

Risk analysis in the Earth sciences often involves considering low-probability, high-impact events. For example, at least one study suggests that large temperature increases could make important parts of the Earth's surface metabolically uninhabitable [Sherwood and Huber, 2010]. Animals, including humans, have to maintain a body temperature within a relatively restricted range in order to survive. Climate modeling indicates that future climate change could raise temperatures near the equator so much that human beings could not live there, except in air-conditioned buildings. This possibility is unlikely, but would have important effects on human societies.

1.5 "Uncertainty" vs. "Risk"

The meanings that we assign to "uncertainty", "deep uncertainty", and "risk" seem intuitive to us; however, other researchers use these words to mean quite different things. The taxonomy we use here separates future events into groups depending on whether or not we have a good sense of the probabilities associated with different outcomes, and it distinguishes the chances of different outcomes from their consequences. However, an influential early taxonomy of uncertainty [Knight, 1921] used "risk" and "uncertainty" to represent the concepts that we term "uncertainty" and "deep uncertainty," respectively. We have also heard influential scientists use the word "uncertainty" to mean the same thing as what we term "risk." Readers of this e-textbook should be aware of these varying definitions when looking at other sources.

Chapter 2

Learning the basics of Julia

This chapter was written by Vivek Srikrishnan and Patrick J. Applegate.

Learning objectives

After completing this chapter, you should be able to * install Julia * write basic Julia commands with appropriate syntax + assigning values to variables + the differences between scalars and arrays + using indexing to extract individual values, or ranges of values, from variables + use loops and `if-else` statements to control the flow of your script's execution + write functions for more readable code which is easier to debug * modify a Julia script to produce a desired result * get help by + searching the Web + asking questions in a way that increases your chance of receiving a helpful response

2.1 Introduction

The computing environment and programming language Julia is a powerful tool for risk analysis. This chapter will guide you through installing Julia and using a text editor or development environment (such as Visual Studio Code or Atom) to write and edit Julia scripts. You will also learn the basics of using Julia, including how to manage, install, and load packages. Finally, you'll become familiar with different ways of getting help when you encounter problems with your Julia code.

2.2 Getting Started

2.2.1 Why Julia?

Julia is a powerful emerging programming language. Julia code is relatively readable while still executing quickly. While initially the Julia community was relatively limited, this has changed in recent years, and there is a wide ecosystem of packages which provide additional functionality to the base Julia release.

2.2.2 Installing Julia

To install Julia, navigate to https://julialang.org/ and look for the link that says, "Download" near the top of the page. Click on this link and follow the instructions for installing the current stable release version which is appropriate for your computing environment. This textbook is currently built using Julia v.1.6.4.

By default, Julia provides a command-line interface for running commands and executing scripts, as well as the interactive "REPL" (read-eval-print-loop). The REPL is started by calling `julia` with no arguments or by double-clicking the Julia executable icon, while calling `julia <scriptname.jl>` will execute `scriptname.jl`. We recommend that you install and use a text editor or integrated development environment (IDE) for opening and editing Julia scripts, which can then be executed at the command line. One popular IDE is Visual Studio Code, which has a powerful Julia extension which includes autocompletion, syntax monitoring, an integrated REPL, a plot pane, and other features. Atom is another popular text editor which can be extended using Juno for advanced Julia features (though Juno is currently depreciated in favor of the VS Code Julia extension). However, if you currently use a particular IDE or editor for coding, it will work well for Julia programming!

2.2.3 Why write scripts?

It's possible to do quite a bit in Julia just by issuing commands one-at-a-time in the REPL. However, real scientific analyses should be *reproducible*; that is, there should be a list of steps that anyone can follow to get the same results as the person who originally did the analysis. Saving these steps as a script allows others to execute your code easily and helps ensure consistent results.

2.2.4 Package and environment management

Another important component of reproducibility is ensuring that others are using the same packages and package *versions* as you did, as updates could result in different output or different commands. Julia has a powerful package management system, `Pkg`. In the REPL, this interface can be accessed by entering `]` at the `julia>` prompt. In a script, we will directly interact with the `Pkg` package, as we will see over the course of this book.

2.2.5 Getting help

What should you do if you don't understand what a Julia command does or how to achieve a particular coding goal? If you have a specific command that you'd like more information on, you can type `?functionname` at a code prompt or in the REPL to get help. Your first external resource should be the Julia documentation or the documentation for the particular package you're drawing from. If you can't find the solution in the documentation (or don't know where to start), a web search can often be useful, as there is an active Julia community which answers questions at the Julia discourse and Stack Overflow.

2.2.6 When the documentation and searching the Internet don't help

When you encounter problems in writing R code that you can't solve by looking at the documentation or searching the Internet, you'll want to ask someone who knows more about Julia than you do. When you do, you should try to ask your questions in a way that maximizes your chances of getting the help you need.

Info

Before asking any questions, you should use the resources available to you, as mentioned above: search the documentation and the Web. In particular, if you get an error message, try pasting it into Google.

If you still require assistance, your question should say clearly

1. what you are trying to do (the goal of your program)
2. what your program does that addresses the problem
3. what you were expecting to have happen when you ran your program
4. what actually happened (did you get an error message? if so, what did it say?)
5. what you did to try and solve your problem before asking your question

For an example of a well-written question, have a look at this post on Stack Overflow. Note that the original poster (user tjnel) addressed all of the five points above in his/her question; he/she

- says explicitly that he/she is trying to reproduce the results from a tutorial (question #1, above),
- gives the code that is being used, allowing others to fully understand the problem (#2)
- provides the expected output from running the code in the tutorial (#3),
- shows the error message that he/she obtained when running the tutorial code (#4),
- shows that he/she tried to reproduce a different part of the tutorial exercise before asking a question (#5).

For other ideas about asking good questions, have a look at Stack Overflow's article on "How do I ask a good question?", Jon Skeet's blog post on "Writing the Perfect Question", and Eric S. Raymond's classic "How to Ask Questions the Smart Way".

2.3 Julia Syntax

Now we will introduce some basic Julia syntax. There are several good introductions to Julia; we recommend the introductory material for the open-source MIT course Introduction to Computational Thinking.

2.3.1 Comments

Comments hide statements from the interpreter or compiler. It's a good idea to liberally comment your code so readers (including yourself!) know why your code is structured and written the way it is.

Single-line comments in Julia are preceded with a `#`. Multi-line comments are preceded with `#=` and ended with `=#`

```
# This is a single-line comment. Everything after the "#"
#   is not evaluated.

#=
This is a multi-line comment.

We might want to really organize our thoughts on what a
function is doing in this space. We can take as much room
as we want, with as many paragraphs.
=#

# now that we've exited the multi-line comment,
#subsequent expressions are evaluated

3+5
```

```
8
```

2.3.2 Variables

Variables are names which correspond to some type of object. These names are bound to objects (and hence their values) using the `=` operator.

```
x=5
```

```
5
```

Variables can be manipulated with standard arithmetic operators.

```
x+4
```

```
9
```

2.3.3 Data Types

Each datum (importantly, *not* the variable which is bound to it) has a data type.

Info

Strictly speaking, a variable points to a particular memory address, which holds the information associated with some datum or data. These pieces of memory can be used to store different data as the variable is overwritten. This can be restricted to varying degrees depending on the programming language. In a *statically typed* language like C, the compiler needs to allocate memory based on the data type, and so once a variable is initialized with a given type, this type cannot be changed, even if the data itself can be. In a *dynamically typed* language such as Python, the types associated with variables can be changed, which may mean the variable needs to be assigned to a different piece of memory. This is one reason why compiled (and usually statically-typed) languages are often faster than interpreted (and usually dynamically-typed) languages.

Julia is a dynamically-typed language, which means that you do not need to specify the type of a variable when you define it, and you can change types mid-program. While Julia is dynamically-typed, you can specify the type of a variable when it is declared by using double-colons (`::`) in the form `variable::type`. This can increase speed, but perhaps just as importantly, can make it easier to identify type errors when debugging. In general, there will be no need for the tasks in this book to worry about any data types other than `Int64` (integers), `Float64` (real numbers), `Bool` (Booleans), and `String` (strings). Julia types are similar to C types, in that they require not only the *type* of data (Int, Float, String, etc), but also the precision (which is related to the amount of memory allocated to the variable). Issues with precision won't be a big deal in this book, though they matter when you're concerned about performance vs. decimal accuracy of code.

You can identify the type of a variable or expression with the `typeof()` function.

```
typeof("This is a string")
```

```
String
```

```
typeof(x)
```

```
Int64
```

To specify that a numeric value should be a `Float64` instead of an `Int64` (which might be needed for some functions which expect `Float64` inputs), add a decimal (*e.g.* `9.` instead of `9`).

2.4 Data Structures

Julia has several native data structures (that is, data structures that don't require additional packages to implement).

Chapter 3

Plotting Time Series Data

This chapter was written by Patrick J. Applegate and Vivek Srikrishnan.

Learning Objectives

After completing this chapter, you should be able to

- explain the relationship between atmospheric carbon dioxide concentration, surface air temperature, and sea level
- download data from the Internet using Julia
- read data files into Julia
- make simple plots in Julia, including
  - plots with multiple panels
  - plots with multiple curves in the same panel

3.1 Introduction

This textbook on climate risk management will discuss sea-level rise extensively. Many people live near present-day sea level [e.g. Nicholls et al., 2008], and rises in sea level expose these people to the possibility of flooding. For example, the ongoing increase in sea level will likely cause communities on the United States' eastern coast to experience frequent flooding within the next few decades [Spanger-Siegfried et al., 2014].

Sea level rise is caused by temperature increases, which in turn are driven by increases in carbon dioxide concentrations in the atmosphere. Carbon dioxide is produced by human activities and natural processes. Increases in carbon dioxide concentrations in the atmosphere enhance the trapping of infrared radiation near the Earth's surface and contribute to rises in surface air temperatures. As the ocean absorbs some excess heat from

the atmosphere, its temperature increases, causing it to expand and causing sea level rise. Temperature increases cause melt of glaciers and ice sheets, which leads to sea level rise by adding mass to the oceans.

Data covering the last century support this relationship between atmospheric carbon dioxide concentrations, temperature, and sea level (Fig. 3.1). The curves in the three panels of Figure 3.1 rise together, suggesting that these variables are related.

Info

Although correlation does not prove causation, the combination of a clear relationship between variables with a plausible explanation for why they should be related is evidence for causation.

Frequent, accurate measurements of carbon dioxide in the atmosphere began in the late 1950s at Mauna Loa in Hawaii [Keeling et al., 1976, blue curve in Fig. 3.1, top panel]. These measurements show an annual cycle that represents Northern Hemisphere seasons. Plants lose their leaves or die during the winter, releasing carbon dioxide to the atmosphere. The Northern Hemisphere has much more land, and therefore more plants, than the Southern Hemisphere. Thus, the Northern Hemisphere's seasons largely control the variability in atmospheric carbon dioxide concentrations within any individual year. However, there is a definite upward trend in this curve that is larger than the amplitude of the annual cycle.

Measurements of former atmospheric compositions from bubbles trapped in ice cores allow us to extend the observational record of carbon dioxide concentrations farther back in time [MacFarling Meure et al., 2006, black curve in Fig. 3.1, top panel]. As snow falls on the Greenland and Antarctic ice sheets, it traps samples of the atmosphere. Because new snow buries and compresses old snow, the time at which different snow samples fell can be estimated by counting the layers in an ice sheet. The ice core measurements of atmospheric carbon dioxide concentrations are less finely resolved in time than the direct measurements, and therefore don't reflect the annual cycle of CO_2 in the atmosphere. However, the trend of the ice core data is similar to that of the direct observations during the period when they overlap, suggesting that the ice core data are reliable.

Because carbon dioxide mixes readily in the atmosphere, measurements of atmospheric carbon dioxide concentrations at most places on the Earth's surface are relatively representative of the globe as a whole. In contrast, both surface air temperatures and sea levels are measured at widely dispersed stations and must be aggregated to give a global mean value. Global mean temperatures must be estimated from individual weather stations with long records [Hansen et al., 2010]; past sea levels are estimated using data from tide gages [Jevrejeva et al., 2014]. As one might expect, there are various methods for performing this aggregation, and the different methods give somewhat different answers. However, it seems clear that both global mean surface air temperature and global mean sea level are

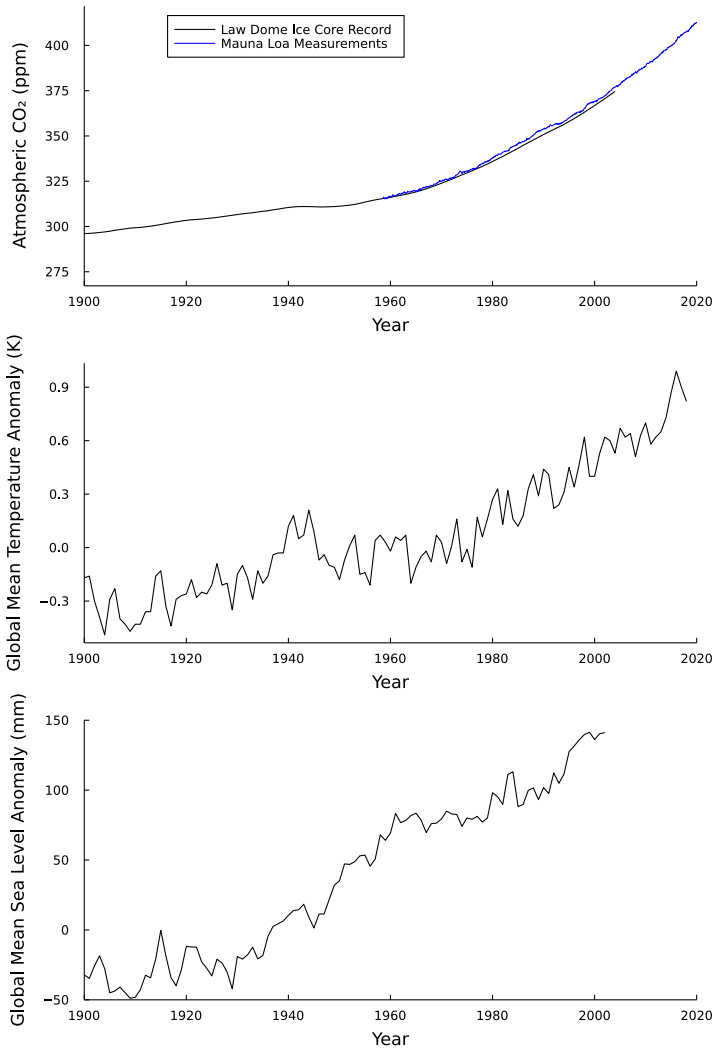


Figure 3.1: Atmospheric carbon dioxide concentrations (top panel), surface air temperature change (middle panel), and sea level change (bottom panel), between 1900 and 2015. All three quantities rise over this period, possibly suggesting a causal relationship between them given a plausible theoretical connection. See text for discussion.

rising.

In this chapter, you'll examine an R script that downloads the data files needed to make the top two panels of Figure 3.1 from the Internet and plots them. You'll then modify the script so that it plots all three panels of Figure 3.1.

3.2 Tutorial

We'll start by looking at several Web pages that describe the data we'll be using in this exercise. As of May 2015, the various data sets displayed in Figure 3.1 are archived in the following places:

Table 3.1: Internet sources of data used in Exercise #1 and associated references. NOAA, National Oceanic and Atmospheric Administration; DOE CDIAC, United States Department of Energy – Carbon Dioxide Analysis Center; NASA GISS, National Aeronautics and Space Administration – Goddard Institute for Space Studies; PSMSL, Permanent Service for Mean Sea Level.

Data type	Reference	Location on the Web
CO_2, direct measurements	Keeling et al. [1976]	NOAA
CO_2, Law Dome ice core	MacFarling Meure et al. [2006]	DOE CDIAC
Surface air temperature change	Hansen et al. [2010]	NASA GISS
Sea level anomaly	Jevrejeva et al. [2014]	PSMSL

Click on the links in the right-hand column of Table 3.1 and look at the descriptions of the data stored there. Also look for links to the particular data sets mentioned in the Introduction. Some Web pages contain links to multiple data sets; we want the "Mauna Loa CO_2 monthly mean data," the Law Dome data (scroll down to near the bottom of the page), and the "global-mean monthly, seasonal, and annual means, 1880-present, updated through most recent month" of the "Land-Ocean Temperature Index, LOTI.

3.2.1 Downloading the Data

Let's step through how to make the first two panels of the plot above. Notice the liberal use of comments throughout the code; using comments like this is a good way to document your code and help yourself (and others!) interpret it.

```
# if "data/" does not exist, create it
isdir("data") || mkdir("data");
# download files and save in "data"
download("ftp://aftp.cmdl.noaa.gov/products/trends/co2/co2_mm_mlo.txt",
    "data/co2_mm_mlo.txt");
download("ftp://ftp.ncdc.noaa.gov/pub/data/paleo/icecore/antarctica/law/law2006.txt",
    "data/law2006.txt");
download("https://data.giss.nasa.gov/gistemp/tabledata_v3/GLB.Ts+dSST.txt",
    "data/GLB.Ts+dSST.txt");
```

The first line creates a `data/` directory if one does not already exist. It actually contains a compound command. As discussed in Section **??**, `||` is a *short-circuit* or operator. In this context, it first evaluates `isdir("data")`, which is `true` if `data/` exists and is a directory and `false` if not. If `isdir("data")` is `true`, there is no need to continue. If `data/` does not exist and `isdir("data")` is `false`, then it evaluates `mkdir("data")`, which makes the `data/` directory. This is a useful alternative in Julia to the following code:

```
# if "data/" is not a directory, evaluate the rest
if ! isdir("data")
    # make the "data/" directory
    mkdir("data")
end
```

The `download(url, location)` commands download files and save them. The `url` argument is the source for the data, while `location` is the local path where the file should be saved.

3.2.2 Reading the Data

Now that we have the files, we need to read the information that they contain into Julia. One of the tricky things with reading files is that you have to know something about how the file is formatted to specify how the code should read it. Open each of the data files using a text editor and examine them. Which files contain "extra" information at the top or bottom of the file? In each file, which columns contain the information we need?

You may notice that these files are *tab-delimited*, which means that columns are separated using tabs. Ither files may be space-delimited or comma-delimited, also known as comma-separated or CSV files. With this sort of structure, we can read in the files using the `readdlm()` command, which is short for "read delimited." `readdlm()` is from the `DelimitedFiles.jl` package, which means we need to load it. `DelimitedFiles.jl` comes with base Julia, so there's no installation required:

```
using DelimitedFiles # load DelimitedFiles.jl

# begin reading in the data
#  read Mauna Loa CO2 Data
loa_co2data = readdlm("data/co2_mm_mlo.txt", skipstart=57,
    header=false);
# read Law Dome CO2 Data
law_co2data = readdlm("data/law2006.txt", skipstart = 183,
    header = false)[1:2004, :];
```

In each of the `readdlm()` commands, the first argument is the file to open. Look at the documentation for `readdlm()`. We also use two other options: * `skipstart` skips the specified number of lines from the beginning of the file, which allows us to ignore headings and comments at the beginning of the file; * `header = false` tells `readdlm()` not to interpret the first read line (after the skipped lines) as column headers for the resulting array. We also tell Julia to only keep the first 2004 rows of the Law Dome data (after skipping the first 183 lines of the file). We do this because there is additional information at the bottom of the file that we do not need to keep.

The temperature data file is formatted in blocks, each of which contains about 20 years of data. For that reason, it is much more complicated to read into Julia. The code block below accomplishes this task using a `for` loop.

```
# read in temperature data; this requires reading in sections
# of the file one at a time
# specify the row numbers where each chunk of data begins
start_rows = [9, 31, 53, 75, 97, 119, 141];
# how many rows are in each chunk
num_rows = [19, 20, 20, 20, 20, 20, 18];
# this computes the total amount of data before reading in
# each chunk, which we need to index the resulting array
cum_num_rows = cumsum(num_rows);
# initialize the storage for the temperature data array
temp_data = zeros(sum(num_rows), 20);
# read in the first chunk
temp_data[1:num_rows[1], :] = readdlm("data/GLB.Ts+dSST.txt",
    skipstart = start_rows[1], header = false)[1:num_rows[1], :];
# loop over the remaining chunks and read them in
for i = 2:length(start_rows)
    temp_data[(cum_num_rows[i-1] + 1):cum_num_rows[i], :] =
        readdlm("data/GLB.Ts+dSST.txt", skipstart = start_rows[i],
            header = false)[1:num_rows[i], :];
```

```
end
```

3.2.3 Plotting the Data

Finally, we make the figure. This requires loading the `Plots.jl` package (see its documentation), which provides plotting functionality. If this were our first time making plots in Julia, we would need to install `Plots`, which we would do using `Pkg`. These lines are commented out in the code below because we've already done this for this book.

```
## install and load relevant packages
# these two lines are commented out as we already have the
# `Plots` library installed; if we did not, we would need to
# include them
# using Pkg
# Pkg.add("Plots")

# now we load `Plots.jl` as well as the `Plots.Measures`
# subpackage, which lets us refer to units like "in" and "mm"
using Plots
using Plots.Measures
# use GR backend; see text discussion
gr();

## first plot CO2 data
# save the base Law Dome data plot as `p1`
p1 = plot(law_co2data[:, 1], law_co2data[:, 6], color="black",
    label="Law Dome Ice Core Record", legend=:top, grid=false);
# add the Mauna Loa data on top of the Law Dome data
plot!(loa_co2data[:, 3], loa_co2data[:, 5], color="blue",
    label="Mauna Loa Measurements");
# label the x-axis
xlabel!("Year");
# we only want to plot data after 1900 even though we have
# more from Law Dome
xlims!((1900, 2020));
# label the y-axis
ylabel!("Atmospheric CO  (ppm)");

## then plot temperature data
# save the base temperature data plot as `p2`
p2 = plot(temp_data[:, 1], temp_data[:, 14] / 100,
```

```
    color="black", legend=false, grid=false);
xlabel!("Year");
xlims!((1900, 2020));
ylabel!("Global Mean Temperature Anomaly (K)");

## combine plots
# finally, combine the two plots as two subplot rows
plot(p1, p2, layout= (2, 1), left_margin=20mm,
    right_margin=20mm);
# adjust the size so the final combined plot isn't squished
# together
plot!(size=(800, 600))
```

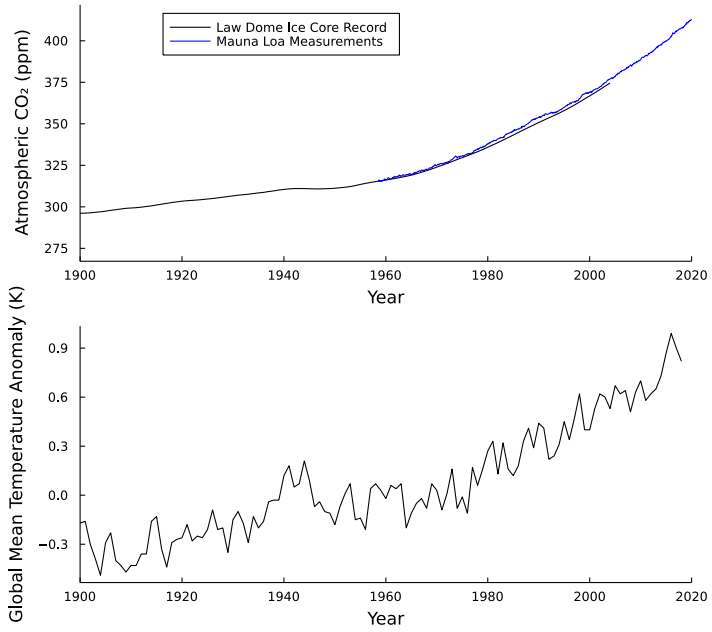


Figure 3.2: Plot results after running the code in this chapter. See text for discussion.

Let's walk through the code preceding Figure 3.2. First, `Plots` uses various backends to provide varying capabilities. We will use the basic `GR` backend, so we call `gr()` after loading `Plots` to set this up. Next, we plot the Law Dome data using the `plot()` function. The first argument to `plot()` contains the x-coordinates of the data; this is the column specifying the year. The second argument are the y-coordinates, which is the sixth column in the data. We can also set the color (`color=black`) and the legend label (`label="Law Dome Ice Core Record"`), since the atmospheric CO_2 plot will also include the Mauna

Loa data. Finally, we set the legend position to be top-center with `legend=:top` (there are lots of options for legend placement), and turn off the grid corresponding to the axis tick marks with `grid=false`. Notice that by default, `plots()` will plot this data as a line, though we could also set it to plot the data as points if desired.

Next, we want to overlay the Mauna Loa data. As discussed earlier, the presence of an exclamation mark after a function name (`plots!()` as opposed to `plots()`) means that the function will modify its argument in-place, rather than returning a new variable. This allows us to avoid creating a new set of axes. We color this data blue to distinguish it from the Law Dome data series. We then can add axis labels (using `xlabel!()` and `ylabel!()`, which once again modify the current plot rather than creating a new plot object) and change the limits (in this case, we only want to change the x-axis limits to crop the data to post-1900).

Why did we save this plot object in the `p1` variable? This will allow us to later include it as a subplot in a final combined plot.

The process of creating the second subplot panel, stored as `p2`, is similar, only we have just the single temperature series, so we never need to make a call to `plot!()` or set up a legend. In fact, we turn off the legend with `legend=:false`.

We can now create our combined, two-panel plot with a call to `plot()`. In this case, we can just specify the two variables which stored our panels, `p1` and `p2`, and provide a layout (in the form `(rows, cols)`). It turns out that in this case, doing this cuts off the left and right edges of the plot, so we set a larger margin with `left_margin` and `right_margin`. We can use intuitive units like `mm` because we loaded `Plots.Measures`. Lastly, we modify the size plotting region with `plot!(size=(800, 600))` to spread out the plot to 800 pixels wide by 600 pixels tall and avoid the two panels from running into each other.

What if we wanted to save our figure as a file? To do that, we would use `savefig(filename)` to save the last plot object in memory to `filename`. `savefig()` automatically encodes the plot properly based on the specified file extension.

3.3 Exercises

Part 1

Copy the code from this chapter to a new file. Now, using what you've learned from following the detailed discussion above, modify the new file so that it produces a `.pdf` file like Figure 3.1, with three panels. The bottom panel should show the sea level anomalies from Jevrejeva et al. [2014]. Specifically, you'll need to add commands to your new file that

1. download the sea level anomaly data from Jevrejeva et al. [2014] to your `data` folder (have a look at Table 3.1);
2. read in this data to a new variable (you might name your new variable `slr_data`);
3. causes the figure to have three panels instead of two;
4. plots the sea level anomaly data in the bottom panel;
5. adjusts the size of the resulting figure;
6. saves the figure to a file.

Make sure that the lower panel of your plot has sensible axis limits in the x and y directions, as well as descriptive labels for the axes (see Fig. 3.1 for an example).

Part 2

1. By how much have atmospheric carbon dioxide concentrations, global mean temperatures, and sea levels changed between 1900 and the early part of the present century? Hint: Check Figure 3.1 and/or the figure produced by your updated Julia script.
2. How do the changes in atmospheric carbon dioxide concentrations, global mean temperatures, and sea levels between 1900 and 2015 compare to the changes in the same variables between the last ice age (~20,000 years ago) and preindustrial times (around 1850)? Give your results as the changes from Question #1, above, divided by the changes between the last ice age and the preindustrial period. Note that changes in temperature and sea level were small between 1850 and 1900. You'll need to search the Web for values of these quantities during the last ice age and the preindustrial period. If this exercise is for a class, be sure to give the Internet sources you used in this part of the assignment.
3. How do the *rates* of change in atmospheric carbon dioxide concentrations, global mean temperatures, and sea levels between 1900 and 2015 compare to the rates of change in these quantities between the last ice age and preindustrial times? Recall that a rate of change is the change divided by the amount of time over which the change happens.

Chapter 4

Normal distributions and the Galton board

This chapter was written by Patrick J. Applegate and Vivek Srikrishnan.

Learning objectives

After completing this chapter, you should be able to

- describe under what circumstances normal distributions arise
- use quantile-quantile plots to determine whether individual data sets are approximately normal or not
- write simple `for` loops in Julia

4.1 Introduction

Suppose that we ask a class of students to walk individually from one end of a field to the other and count the number of steps that they take in order to cover the distance. More than likely, each student will get a somewhat different answer. After each student has paced off the field, we then make a histogram of the number of steps taken by the individual students.

If we were to perform this experiment with a large number of students, the histogram would likely resemble a normal distribution. Normal distributions arise when a large number of measurements are taken of a single quantity, and the individual measurements are affected by random processes that are additive and that can be either positive or negative. The general property that a sufficiently large random sample will be well-described by a normal distribution is called the Central Limit Theorem, which is foundational in statistics (the

details of the Central Limit Theorem are not essential for this book). In our example, the random processes have to do with the varying stride lengths of the students (some take longer strides and some shorter), and counting errors (which are likely to be small and can cause any given student to either over- or underestimate the actual number of paces they took).

A Galton board is a physical device that demonstrates this concept (Fig. 4.1). A Galton board has rows of pins arranged in a triangular shape, with each row of pins offset horizontally relative to the rows above and below it. The top row has one pin, the second row has two pins, and so forth.

If a ball is dropped into the Galton board, it falls either to the right or the left when it bounces off the top pin. The ball then falls all the way to the bottom of the board, moving slightly to the right or left as it passes through each row of pins. Bins at the bottom of the board capture the ball and record its final position. If this experiment is repeated with many balls, and if the Galton board includes many rows of pins, the number of balls in each bin resembles a normal distribution (Fig. 4.1).

In this exercise, you'll experiment with a simple representation of a Galton board in Julia and examine the distributions of final ball positions that it produces. You'll also examine two data sets that can easily be loaded into Julia and see whether they are well described by a normal distribution or not.

4.2 Tutorial

4.2.1 Writing our own simple Galton board script

Building computer models involves thinking about how to represent the behavior of physical (or chemical, or social) systems in a form that can be understood by a computer. In thinking about how we could represent the Galton board in terms of Julia code, we might start by looking at the diagram of the Galton board in the top part of Figure 4.1. The horizontal spacing between the pins is always the same, but the rows are offset relative to one another; that is, none of the pins in any given row are immediately above or below the pins in the adjacent rows. This offset is exactly half of the horizontal spacing between pins in the same row. Thus, we can imagine that the ball bounces 0.5 distance units to one side whenever it hits a pin.

We first need to specify the number of levels in the Galton board.

```
levels = 15
```

```
15
```

We can then represent the path that a single ball takes through the Galton board by

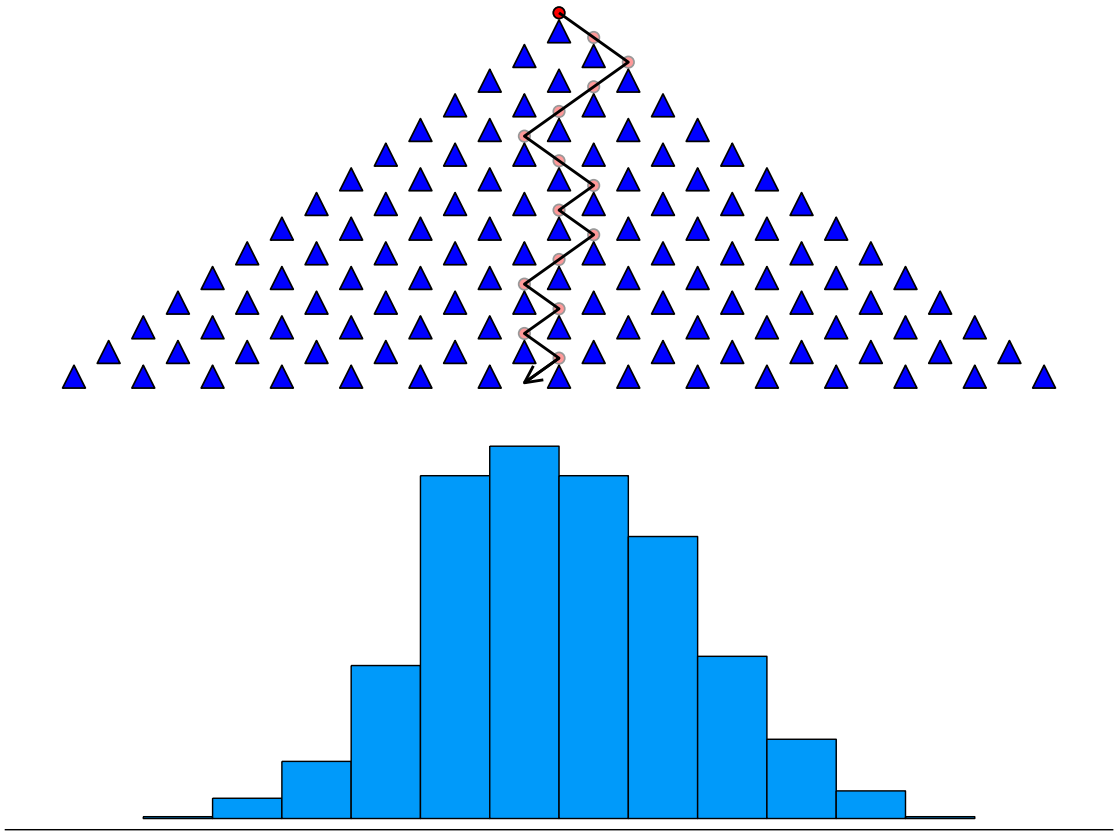


Figure 4.1: A Galton board simulation with 15 rows of pins. The top panel shows the board and one sample ball trajectory. The bottom panel shows one possible distribution of final ball positions resulting from dropping 1000 balls through the board.

randomly sampling how an individual ball bounces as it hits each pin. The Julia package which allows us to do random sampling is `Random.jl`, which is built into Julia's basic distribution. The function which we call is `rand()`:

```
# load the `Random` package
using Random

# sample
path = rand([-0.5, 0.5], levels)
```

```
15-element Vector{Float64}:
  0.5
  0.5
 -0.5
  0.5
 -0.5
  0.5
  0.5
  0.5
 -0.5
 -0.5
 -0.5
 -0.5
  0.5
 -0.5
  0.5
```

Positive values indicates that the ball bounces to the right; negative values indicate that the ball bounces to the left.

The `rand()` command generates sequences of values that come from a vector of possibilities supplied in the first argument. The second argument is `dims`, the dimension of the final sample set. In this case, we are just sampling a single trajectory, so we just need a number of samples equal to `levels`, which will return a 1-dimensional array.

The bin that the ball lands in is then just

```
sum(path)
```

```
0.5
```

4.2.2 Doing things over and over again: `for` loops

The commands above tell us about the path that any single ball takes through the Galton board, as well as the bin it finally lands in. But, suppose we want to write a piece of R code that performs this calculation for many balls and gives us a histogram of the bins that they land in. How could we do that?

Most, perhaps all, programming languages include a method for carrying out a set of instructions a fixed number of times. In most programming languages, including Julia, this method is called a `for` loop. The syntax of `for` loops in Julia looks like this:

```
# set the number of times the loop should be evaluated;
# this is not part of the for loop syntax
neval = 3;
for i = 1:neval
  # instructions to be repeated go here
  println(i^2)
end
```

```
1
4
9
```

```
# note the end; this is required to tell Julia where the
# loop instructions end!
```

This sample `for` loop is not very exciting: it simply `println()`s the square of the *index variable* `i` during each iteration of the indented commands (between `for` and `end`).

In "real" `for` loops, the name of the index variable, the vector of values that the index variable can take (here `1:neval`), and the evaluated instructions can be more complex. In particular, there can be more than one line of instructions evaluated. In Julia, you can also iterate over arbitrary collections, such as arrays, not just sequences of index variables.

We usually will want to create a variable that will store the results from the commands carried out within the `for` loop. The following block of code generates and `print()`s the first five Fibonacci numbers:

```
neval = 5
```

```
5
```

```
 # make a vector of `neval` zeroes, which will store the output
fibonacci = zeros(neval)
```

```
5-element Vector{Float64}:
```

```
 0.0
 0.0
 0.0
 0.0
 0.0
```

```
for i = 3:neval
  # stores the sum of the preceding two elements of `fibonacci`
  # in the ith element of `fibonacci`
  fibonacci[i] = sum(fibonacci[i-2:i-1])
end
println(fibonacci)
```

```
[0.0, 0.0, 0.0, 0.0, 0.0]
```

In general, it's good practice to pre-allocate storage for the output of `for` loops as we did by defining the `fibonacci` array. If we were to just start with the initial two Fibonacci numbers and add to the length of the array at each iteration, this would slow down the program's execution due to the need to assign new memory to the resulting array every time it grew. This might not be noticeable for the code block above due to the small number of evaluations, but it will be for more complex code.

In the Exercise below, you'll put together the information above to write your own script for making histograms of the outcomes from a Galton board. First, though, let's examine how we determine whether a particular data set is normal or not.

4.2.3 Normal or not normal?: Generating and interpreting quantile-quantile plots

It can be tempting to assume that a particular process gives normally-distributed output, or that data are normally distributed; however, assuming normality when it isn't present can lead to large errors. How can we check individual data sets for normality?

One robust and simple, but approximate, test for normality involves making a quantile-quantile (Q-Q) plot. The details of how Q-Q plots work aren't important for our purposes here. However, if data are drawn from a normal distribution, they will lie on a line connecting the first and third quartiles of the data when drawn on an appropriately-constructed Q-Q plot. The `StatsPlots.jl` package makes drawing Q-Q plots easy.

To see how this looks, let's examine samples from different distributions. We can do this using the `Distributions.jl` package, which lets us specify distributions to sample from. We can then use `rand()` from the

For example, the following block of code generates 100 random numbers from a normal

distribution, makes a Q-Q plot from them, and draws a 1:1 line on the plot:

```
# load the packages we will use
using Distributions
using Random
using StatsPlots

# specify the normal distribution with mean = 0
# snd standard deviation = 1, which are the default choices
normal_distribution = Normal();
# draw 100 samples
normal_samples = rand(normal_distribution, 100);
# make the Q-Q plot
qqplot(Normal(), normal_samples)
```

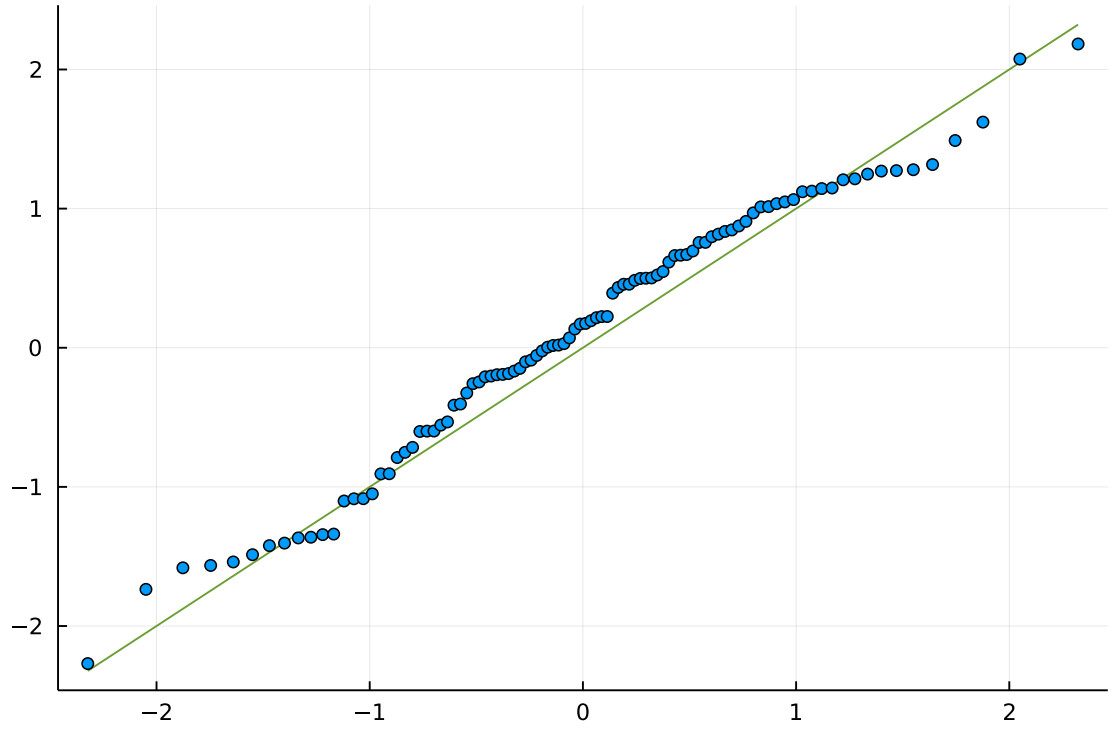


Figure 4.2: A Q-Q plot showing 100 normally-distributed random values. The points, which represent the random values, lie approximately on the line, showing that their distribution could be treated as a normal distribution.

If we have a data set, we might not know what the parameters of the normal distribution

are for the comparison (in this case, they were just the defaults). One strategy is to use the sample mean and standard deviation, which can be obtained with `mean()` and `std()` (this is justified by the Central Limit Theorem).

Let's see what a Q-Q plot might look like if we draw samples from a distribution that isn't normal. We will sample from a Cauchy distribution, which is a distribution which has a higher probability of extreme values than a normal distribution.

```
# specify the default Cauchy distribution
cauchy_distribution = Cauchy();
# draw 100 samples
cauchy_samples = rand(cauchy_distribution, 100);
# make the Q-Q plot
qqplot(Normal(), cauchy_samples)
```

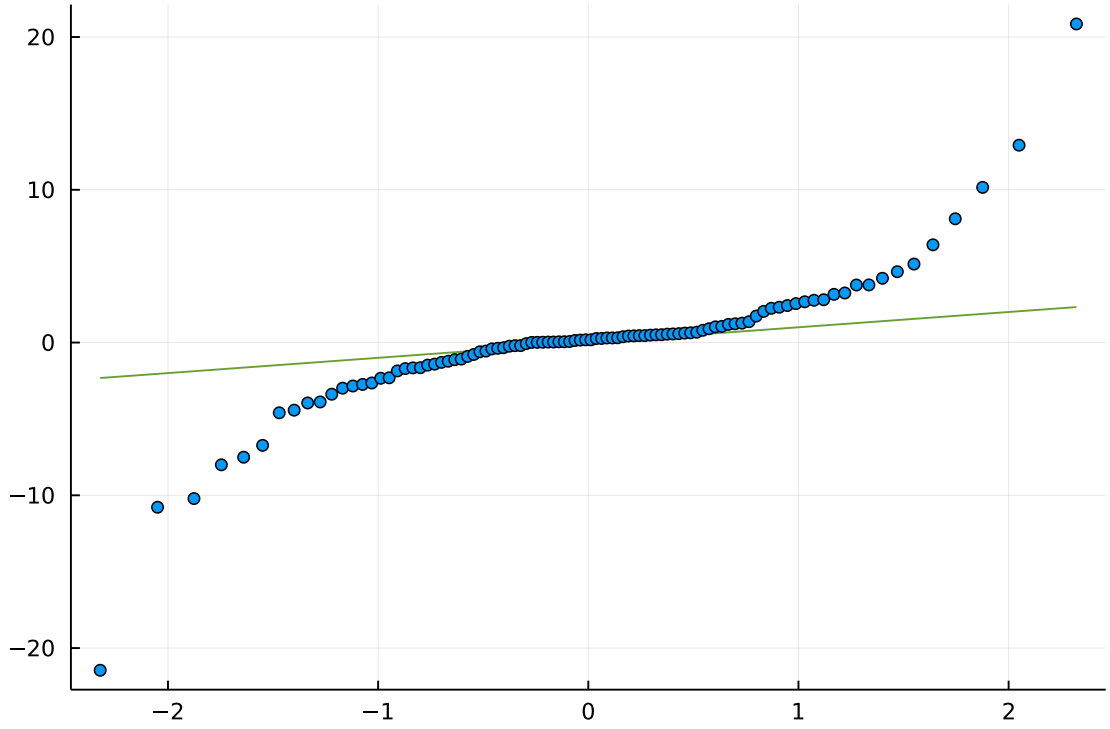


Figure 4.3: A Q-Q plot showing 100 Cauchy-distributed random values. The points, which represent the random values, do not lie on the line for the majority of the plot. In this case, it could be dangerous to assume that the data were normally-distributed.

Unlike in Figure 4.2, the points in Figure 4.3 are generally not along the line, which should

suggest to use that the data are not normally distributed. On the left side of the Q-Q plot, the points are well below the line, and on the right side, they are well above it. This actually gives us more information about how the sample deviates from being normally distributed! The fact that the left side is below the line shows us that the distribution is "fatter" below the mean than a normal distribution, or that its left tail is heavier than that of a normal distribution. This is because a normal distribution's samples are more limited in range than the samples from the Cauchy distribution. This means that it is more likely to see large negative values in this sample than we would expect from a normal sample. Similarly, the right side being above the line shows us that large positive values are more present than we would expect from a normal sample, and the right tail is fatter than a normal distribution's. If the points and the line had the opposite relationship, we could conclude that the sample had thinner tails, and was more concentrated, than would be expected from a normal distribution. And if we had a mixed situation, that would suggest to us that our sample was more skewed to one side or the other.

4.3 Exercises

Part 1

Based on the information presented in this chapter, write a Julia script that

1. sets values for the number of balls to drop and the number of rows of pins in the Galton board (you might use the variable names `nballs` and `levels`)
2. creates a vector `output` that contains a number of elements equal to `nballs`; these elements can be populated with `0`s or `nothing`s to start;
3. uses a `for` loop that

- runs through values of the index variable `i` from `1:nballs`,
- determines which bin of the Galton board each ball lands in, and
- stores these values in the different elements of `output`

4. makes a histogram of `output`. The code block below shows one way of carrying out this step.

```
# Make a histogram of the results.
using Plots
gr();
histogram(output, xlabel = "Bin", ylabel = "Count",
  legend=:false)
```

Set the number of rows of pins in your Galton board R script to 10. Also set the number of balls to drop to a low value, say 10 to begin. + How does the histogram change as you repeatedly run the script? + What is the smallest number of balls that you can drop that

gives more or less the same histogram each time you run the script, for 10 rows of pins?

Info

We could also do this problem differently by sampling all the trajectories for all the balls simultaneously. We would do this by specifying a tuple for the `dims` argument of `rand()`. If we wanted to sample `nballs` ball paths at once, we could set `dims=(nballs, levels)` and the result will be formatted as an array with `nballs` rows and `levels` columns. We would then iterate over the array and sum to get the final positions for each of the `nballs`.

Part 2

Add commands to your Julia script so that it also generates a Q-Q plot for the results in `output`.

Recall from the discussion above that a Galton board only gives an approximately-normal distribution if both the number of balls that are dropped and the number of bins are large. Leaving the number of rows of pins at 10, increase the number of balls to a large value like 10^4 (in code, that would be just `10^4`). Confirm that the `output` under these circumstances is not normally distributed. How large does the number of rows of pins have to be to get approximately-normal results?

Part 3

Generate Q-Q plots for two commonly-used data sets,

- the waiting time between Old Faithful eruptions, and
- the sepal length of setosa irises on the Gaspe Peninsula.

1. Install and load the `RDatasets` package.
2. Load the Old Faithful (`dataset("datasets", "faithful")[:, 2]`) and iris sepal (`dataset("datasets", "iris3")[:, 1, 1]`) datasets.
3. Generate your Q-Q plots.

Which of these datasets is approximately normally distributed, and which is not? How can you tell?

References

Richard B Alley. *Earth: The Operators' Manual*. W. W. Norton and Company, 2011.

John A Church, Peter U Clark, Anny Cazenave, Jonathan M Gregory, Svetlana Jevrejeva, Anders Levermann, Mark A Merrifield, Glenn A Milne, R Steven Nerem, Patrick D Nunn, Antony J Payne, W Tad Pfeffer, Detlef Stammer, and Alakkat S Unnikrishnan. Sea level change. In T F Stocker, D Qin, G-K Plattner, M Tignor, S K Allen, J Boschung, A Nauels, Y Xia, V Bex, and P M Midgley, editors, *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change.*, pages 1137–1216. Cambridge University Press, Cambridge, UK and New York, NY, 2013.

Matthew Collins, Reto Knutti, Julie Arblaster, Jean-Louis Dufresne, Thierry Fichefet, Pierre Friedlingstein, Xuejie Gao, William J Gutowski, Jr., Tim Johns, Gerhard Krinner, Mxolisi Shongwe, Claudia Tebaldi, Andrew J Weaver, and Michael Wehner. Long-term climate change: Projections, commitments and irreversibility. In T F Stocker, D Qin, G-K Plattner, M Tignor, S K Allen, J Boschung, A Nauels, Y Xia, V Bex, and P M Midgley, editors, *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, Cambridge, UK and New York, NY, 2013.

Christopher B Field, Vicente R Barros, Katharine J Mach, Michael D Mastrandrea, Maarten K van Aalst, W Neil Adger, Douglas J Arent, Jonathon Barnett, Richard A Betts, T Eren Bilir, Joern Birkmann, Joann Carmin, Dave D Chadee, Andrew J Challinor, Monalisa Chatterjee, Wolfgang Cramer, Debra J Davidson, Yuka Otsuki Estrada, Jean-Pierre Gattuso, Yasuaki Hijioka, Ove Hoegh-Guldberg, He-Qing Huang, Gregory E Insarov, Roger N Jones, R Sari Kovats, Joan Nymand Larsen, Iñigo J Losada, José A Marengo, Roger F McLean, Linda O Mearns, Reinhard Mechler, John F Morton, Isabelle Niang, Taikan Oki, Jane Mukarugwiza Olwoch, Maggie Opondo, Elvira S Poloczanska, Hans-O Pörtner, Margaret Hiza Redsteer, Andy Reisinger, Aromar Revi, Patricia Romero-Lankao, Daniela N Schmidt, M Rebecca Shaw, William Solecki, Dáithí A Stone, John M R Stone, Kenneth M Strzepek, Avelino G Suarez, Petra Tschakert, Riccardo Valentini, Sebastián Vicuña, Alicia Villamizar, Katharine E Vincent, Rachel Warren,

Leslie L White, Thomas J Wilbanks, Poh Poh Wong, and Gary W Yohe. Technical summary. In T F Stocker, D Qin, G-K Plattner, M Tignor, S K Allen, J Boschung, A Nauels, Y Xia, V Bex, and P M Midgley, editors, *Climate Change 2013: The Physical Science Basis. Contribution of Working Group II to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. 2013.

David Hadka, Jonathan Herman, Patrick Reed, and Klaus Keller. An open source framework for many-objective robust decision making. *Environ. Modell. Softw.*, 74:114–129, 2015. ISSN 1364-8152. doi: 10.1016/j.envsoft.2015.07.014.

J Hansen, R Ruedy, M Sato, and K Lo. Global surface temperature change. *Rev. Geophys.*, 48(4):644, December 2010. ISSN 8755-1209. doi: 10.1029/2010RG000345.

IPCC. Summary for policymakers. In T F Stocker, D Qin, G-K Plattner, M Tignor, S K Allen, J Boschung, A Nauels, Y Xia, V Bex, and P M Midgley, editors, *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. Cambridge University Press, Cambridge, UK and New York, NY, 2013.

S Jevrejeva, J C Moore, A Grinsted, A P Matthews, and G Spada. Trends and acceleration in global and regional sea levels since 1807. *Glob. Planet. Change*, 113:11–22, February 2014. ISSN 0921-8181. doi: 10.1016/j.gloplacha.2013.12.004.

Charles D Keeling, Robert B Bacastow, Arnold E Bainbridge, Carl A Ekdahl, Jr., Peter R Guenther, Lee S Waterman, and John F S Chin. Atmospheric carbon dioxide variations at mauna loa observatory, hawaii. *Tell'Us*, 28(6):538–551, December 1976. ISSN 0804-6042. doi: 10.1111/j.2153-3490.1976.tb00701.x.

Frank H Knight. *Risk, Uncertainty, and Profit*. Houghton Mifflin Company, Boston, MA and New York, NY, 1921.

Lee R Kump, James F Kasting, and Robert G Crane. *The Earth System (3rd Edition)*. Pearson, 3rd edition, 2009.

C MacFarling Meure, D Etheridge, C Trudinger, P Steele, R Langenfelds, T van Ommen, A Smith, and J Elkins. Law dome CO2 , CH4 and N2O ice core records extended to 2000 years BP. *Geophys. Res. Lett.*, 33(14):101, 2006. ISSN 0094-8276. doi: 10.1029/2006GL026152.

Gerald A Meehl, Thomas F Stocker, William D Collins, Pierre Friedlingstein, Amadou T Gaye, Jonathan M Gregory, Akio Kitoh, Reto Knutti, James M Murphy, Akira Noda, Sarah C B Raper, Ian G Watterson, Andrew J Weaver, and Zong-Ci Zhao. Global climate projections. In Susan Solomon, Qin Dahe, Martin Manning, Melinda Marquis, Kristen Averyt, Melinda M B Tignor, Henry L Miller, Jr., and Zhenlin Chen, editors, *Climate Change 2007: The Physical Science Basis. Contribution of Working Group I*

to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change. Cambridge University Press, Cambridge, UK and New York, NY, 2007.

Robert J Nicholls, Richard S J Tol, and Athanasios T Vafeidis. Global estimates of the impact of a collapse of the west antarctic ice sheet: an application of FUND. *Clim. Change*, 91(1):171, June 2008. ISSN 0165-0009, 1573-1480. doi: 10.1007/s10584-008-9424-y.

Naomi Oreskes. The scientific consensus on climate change: How do we know we're not wrong? In Elisabeth A. Lloyd and Eric Winsberg, editors, *Climate Modelling: Philosophical and Conceptual Issues*, pages 31–64. Springer International Publishing, Cham, 2007. ISBN 9783319650586. doi: 10.1007/978-3-319-65058-6_2.

Adam Parris, Peter Bromirsky, Virginia Burkett, Dan Cayan, Mary Culver, John Hall, Radley Horton, Kevin Knuuti, Richard Moss, Jayantha Obeysekara, Abby Sallenger, and Jeremy Weiss. Global sea level rise scenarios for the united states national climate assessment. Technical Report OAR CPO-1, NOAA, 2012.

Hauke Riesch. Levels of Uncertainty. In Sabine Roeser, Rafaela Hillerbrand, Per Sandin, and Martin Peterson, editors, *Essentials of Risk Theory*, SpringerBriefs in Philosophy, pages 29–56. Springer Netherlands, Dordrecht, 2013. ISBN 978-94-007-5455-3. doi: 10.1007/978-94-007-5455-3_2.

Todd Sanford, Peter C Frumhoff, Amy Luers, and Jay Gulledge. The climate policy narrative for a dangerously warming world. *Nat. Clim. Chang.*, 4:164, February 2014. ISSN 1758-678X. doi: 10.1038/nclimate2148.

Steven C. Sherwood and Matthew Huber. An adaptability limit to climate change due to heat stress. *Proceedings of the National Academy of Sciences*, 107(21):9552–9555, May 2010. ISSN 0027-8424, 1091-6490. doi: 10/ddfgzk.

Erika Spanger-Siegfried, Melanie Fitzpatrick, and Kristina Dahl. Encroaching tides - union of concerned scientists. Technical report, Union of Concerned Scientists, Cambridge, MA, 2014.

Ryan Sriver, Robert Lempert, Per Wikman-Svahn, and Klaus Keller. Characterizing uncertain sea level rise projections to support investment decisions. *PLoS One*, page 37, 2017. doi: PublicationNumber:CEC-500-2012-056.

Thomas F Stocker, Qin Dahe, Gian-Kasper Plattner, Lisa V Alexander, Simon K Allen, Nathaniel L Bindoff, François-Marie Bréon, John A Church, Ulrich Cubasch, Seita Emori, Piers Forster, Pierre Friedlingstein, Nathan Gillett, Jonathan M Gregory, Dennis L Hartmann, Eystein Jansen, Ben Kirtman, Reto Knutti, Krishna Kumar Kanikicharla, Peter Lemke, Jochem Marotzke, Valérie Masson-Delmotte, Gerald A Meehl, Igor I Mokhov, Shilong Piao, Venkatachalam Ramaswamy, David Randall, Monika Rhein, Maisa Rojas,

Christopher Sabine, Drew Shindell, Lynne D Talley, David G Vaughan, and Shang-Ping Xie. Technical summary. In T F Stocker, D Qin, G-K Plattner, M Tignor, S K Allen, J Boschung, A Nauels, Y Xia, V Bex, and P M Midgley, editors, *Climate Change 2013: The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change*. 2013.