Population Genetics Modeling

Table of Contents

# 1 Homepage

This is the course homepage and digital textbook for Population Genetics Modeling (AS.020.369).

#### 1.0.0.1 Instructor

Andrew Bortvin, abortvi2[at]jhu.edu

#### 1.0.0.2 Schedule & Logistics

| Session | Content |
| --- | --- |
| **Session 1:** | Course Introduction |

# 2 Course Syllabus

**COMPUTATIONAL APPROACHES TO POPULATION GENETICS** Instructor: Andrew Bortvin Times: Tuesday, Thursday 3:00PM - 4:30PM Office Hours: To be determined by when2meet at the start of the semester, or by appointment Course website: <https://andrew-bortvin.github.io/popGenModeling/>

### 2.0.1 **Course Description**

The size, composition, and genetics of populations fluctuate over time. These fluctuations are the product of dynamics between individuals, the interactions between populations, and the context of a population within a broader ecological landscape. The quantitative tools developed to study population genetics allow biologists to discover the simple fundamental principles that govern these complex systems. This course will introduce the basic theory of population genetics while teaching students the fundamental skills of programming in the R programming language, which will allow them to directly implement and visualize theoretical concepts. Students will model and simulate theoretical populations and analyze population-scale genomic data. This course will examine evolution on a variety of scales, ranging from the competition between cells within a single organism, to population dynamics in conservation biology that span decades, to the evolution of contemporary human populations over hundreds of thousands of years.

### 2.0.2 **Learning Goals**

By the end of this course, students will be able to:

• outline, using biology theory and terms, how populations grow and interact with each other • describe the external factors that can cause changes in genetic diversity and use this information to predict how specific demographic scenarios would impact a population • manipulate, analyze, and visualize data using R programming • describe and interpret common formats used to store genomic data, and implement standard analytic protocols used to analyze these data types

### 2.0.3 **Grading**

30% Participation 40% Weekly Assignments 30% Final Project

\*\*Weekly Assignments

Class sessions will consist of a lecture in which theory is introduced, instructor-led live-coding sessions that implement the models studied in class, and time for students to work independently on assignments that further develop the models designed in class. There will be one assignment per week, assigned each Thursday and due the following Thursday. After submission, students will receive feedback from the instructor on each assignment and will have until the end of the semester to submit any revisions necessary.

Each assignment will consist of a set of required exercises that can be completed by students of any coding background. These will be followed by optional, more computationally-focused exercises, which will allow students to leverage their computational knowledge in creative application to genomic questions.

Work will be graded on reasonable completion–that is to say, code that demonstrates an understanding of an algorithm and its general implementation will receive full marks, regardless of whether output is exactly correct. Students will also be assessed on the clarity and interpretability of the data visualizations that their code outputs and the accuracy of their responses to short questions prompting biological interpretation of their results.

**Googling and AI**

Googling is always an acceptable way to find answers or help, and I encourage you to utilize it extensively. If you adopt a solution following a Google search, make sure you understand what you incorporate, rather than just copy/paste without comprehension of the logic or code. Google is also a good way to learn more about any error messages you encounter in your code.

You may be familiar with ChatGPT and other large language models. After trying each problem/assignment/task on your own, if you’re still running into issues, feel free to use ChatGPT as you would any other online resource (Google, stack overflow, etc.). Learning how to succinctly describe exactly what you want to accomplish is a skillset in itself, so this can be good practice. If you find code that seems to work (e.g., from Google) but you’re not sure how exactly it works, you can also type it into ChatGPT and ask it to explain what’s happening. As always, please do not submit any code if you are not familiar entirely with how it works; flag it and ask an instructor for assistance. Be aware that ChatGPT might confidently offer an answer that is not correct; so always check the output on your own.

#### 2.0.3.1 **Schedule**

Week 1: Introduction to Python Python Fundamentals: Variables, arithmetic, built-in functions, loops and conditionals, plotting Creating and maintaining a Git repository

Unit 1: Population Biology

Week 2: One Population Models Exponential growth Growth with limited resources: the logistic growth model Age-structured population dynamics

Week 3: Multi-population models Competition, cooperation Predator-Prey Interactions and Lotka-Volterra Dynamics

Week 4: Advanced multi-population models Host-Parasite Interactions and the Nicholson-Bailey model Disease modeling: the SEIR model and community spread

Unit 2: Population Genetics – Theory and Simulation

Week 5: Introduction to Population Genetics The Wright-Fisher model and evolutionarily neutral scenarios

Week 6: Mutation and Natural Selection The Wright-Fisher model with mutation and selection

Week 7: Population Structure and Genetics Population Size Changes Non-random mating

Week 8: Spatial Modeling Migration and local adaptation

Unit 3: Studying Genetic Variation – Data Analysis

Week 9: Quantifying Variation Measures of genome-wide genetic diversity The Site Frequency Spectrum

Week 10: Measuring Selection Tests for selection Local adaptation and adaptive introgression

Week 11: Population Structure Quantification of population structure Measuring admixture and admixture mapping

Week 12: Phylogenetics and Historical Genetics Inferring the timing of selection – coalescent modeling

Week 13: Independent Projects Small-group work sessions In-class presentations

# 3 Introduction to R

R is a programming language developed primarily for use by statisticians, and it is a common tool in the field for data exploration, analysis, visualization, etc. We will be using it throughout the course to manipulate, view, and interpret data.

In this module, we will learn the basics of R by manipulating some real-world data. In particular, we’ll look at data from the Hudson Bay looking at the size of the hare and lynx populations.

We’ll talk a lot more about this dataset when we study predator-prey models in a few weeks, but for now, let’s make some exploratory plots.

Data adapted from here: <http://people.whitman.edu/~hundledr/courses/M250F03/M250.html>

Please note that input data is available on the course Posit cloud page - please feel free to work alongside this page!

## 3.1 Variables in R

## 3.2 Data import

The bread and butter of R is the dataframe, a tabular data structure which contains data of multiple types.

R has a few built-in functions that allow for easy and efficient import of data. As we go on, we’ll learn a lot of nuances about how these work, but for now we’ll start with a tidy, well-behaved dataset that is easy to work with.

We would like for R to import our data and save it into memory. We also want to give it a name, so that we can refer to it later.

In R, we use the assignment operator <- to associate a name with some data. The general syntax for this is:

variableName <- value

Where **left - name right -> value**

**if you know python or c or something, “+” is equivalent**

**lets test it out**

populationData <- read.table("hare\_lynx.csv", sep = ",", header = TRUE)

Let’s break down what’s happening here.

On the left of the <-, we have the text populationData

head(populationData)

## Year Species Count  
## 1 1845 Hare 19.58  
## 2 1846 Hare 19.60  
## 3 1847 Hare 19.61  
## 4 1848 Hare 11.99  
## 5 1849 Hare 28.04  
## 6 1850 Hare 58.00

## 3.3 The Data Frame

R has the ability to manipulate multiple different types of data. The primary data types we will use in this course are:

**Numeric** are whole numbers. For example:

## 3.4 Lecture Notes

In this module, we will learn the basics of R by manipulating some real-world data. In particular, we’ll work with data from the Hudson Bay looking at the size of historical hare and lynx populations.

We’ll talk a lot more about this dataset and the biology it represents when we study predator-prey models, but for now let’s use it make some exploratory plots.

Data adapted from here: <http://people.whitman.edu/~hundledr/courses/M250F03/M250.html>

## 3.5 Data import

Our data is available as a comma-delimited text file, or csv. Here’s the top of it as it appears in a text editor:



The first thing we have to do to work with it is to import it into R. R has a few built-in functions that allow for easy and efficient import of data. As we go on, we’ll learn a lot of nuances about how these work, but for now we’ll start with a tidy, well-behaved dataset that is easy to work with. The read.table function will be our workhorse for reading in data.

Let’s try it out. To run this function, we need to provide one **argument**: the location of the data we want to load.

read.table("hare\_lynx.csv")

## V1  
## 1 Year,Species,Count  
## 2 1845,Hare,19.58  
## 3 1846,Hare,19.6  
## 4 1847,Hare,19.61  
## 5 1848,Hare,11.99  
## 6 1849,Hare,28.04  
## 7 1850,Hare,58  
## 8 1851,Hare,74.6  
## 9 1852,Hare,75.09  
## 10 1853,Hare,88.48  
## 11 1854,Hare,61.28  
## 12 1855,Hare,74.67  
## 13 1856,Hare,88.06  
## 14 1857,Hare,68.51  
## 15 1858,Hare,32.19  
## 16 1859,Hare,12.64  
## 17 1860,Hare,21.49  
## 18 1861,Hare,30.35  
## 19 1862,Hare,2.18  
## 20 1863,Hare,152.65  
## 21 1864,Hare,148.36  
## 22 1865,Hare,85.81  
## 23 1866,Hare,41.41  
## 24 1867,Hare,14.75  
## 25 1868,Hare,2.28  
## 26 1869,Hare,5.91  
## 27 1870,Hare,9.95  
## 28 1871,Hare,10.44  
## 29 1872,Hare,70.64  
## 30 1873,Hare,50.12  
## 31 1874,Hare,50.13  
## 32 1875,Hare,101.25  
## 33 1876,Hare,97.12  
## 34 1877,Hare,86.51  
## 35 1878,Hare,72.17  
## 36 1879,Hare,38.32  
## 37 1880,Hare,10.11  
## 38 1881,Hare,7.74  
## 39 1882,Hare,9.67  
## 40 1883,Hare,43.12  
## 41 1884,Hare,52.21  
## 42 1885,Hare,134.85  
## 43 1886,Hare,134.86  
## 44 1887,Hare,103.79  
## 45 1888,Hare,46.1  
## 46 1889,Hare,15.03  
## 47 1890,Hare,24.2  
## 48 1891,Hare,41.65  
## 49 1892,Hare,52.34  
## 50 1893,Hare,53.78  
## 51 1894,Hare,70.4  
## 52 1895,Hare,85.81  
## 53 1896,Hare,56.69  
## 54 1897,Hare,16.59  
## 55 1898,Hare,6.16  
## 56 1899,Hare,2.3  
## 57 1900,Hare,12.82  
## 58 1901,Hare,4.72  
## 59 1902,Hare,4.73  
## 60 1903,Hare,37.22  
## 61 1904,Hare,69.72  
## 62 1905,Hare,57.78  
## 63 1906,Hare,28.68  
## 64 1907,Hare,23.37  
## 65 1908,Hare,21.54  
## 66 1909,Hare,26.34  
## 67 1910,Hare,53.1  
## 68 1911,Hare,68.48  
## 69 1912,Hare,75.58  
## 70 1913,Hare,57.92  
## 71 1914,Hare,40.97  
## 72 1915,Hare,24.95  
## 73 1916,Hare,12.59  
## 74 1917,Hare,4.97  
## 75 1918,Hare,4.5  
## 76 1919,Hare,11.21  
## 77 1920,Hare,56.6  
## 78 1921,Hare,69.63  
## 79 1922,Hare,77.74  
## 80 1923,Hare,80.53  
## 81 1924,Hare,73.38  
## 82 1925,Hare,36.93  
## 83 1926,Hare,4.64  
## 84 1927,Hare,2.54  
## 85 1928,Hare,1.8  
## 86 1929,Hare,2.39  
## 87 1930,Hare,4.23  
## 88 1931,Hare,19.52  
## 89 1932,Hare,82.11  
## 90 1933,Hare,89.76  
## 91 1934,Hare,81.66  
## 92 1935,Hare,15.76  
## 93 1845,Lynx,30.09  
## 94 1846,Lynx,45.15  
## 95 1847,Lynx,49.15  
## 96 1848,Lynx,39.52  
## 97 1849,Lynx,21.23  
## 98 1850,Lynx,8.42  
## 99 1851,Lynx,5.56  
## 100 1852,Lynx,5.08  
## 101 1853,Lynx,10.17  
## 102 1854,Lynx,19.6  
## 103 1855,Lynx,32.91  
## 104 1856,Lynx,34.38  
## 105 1857,Lynx,29.59  
## 106 1858,Lynx,21.3  
## 107 1859,Lynx,13.69  
## 108 1860,Lynx,7.65  
## 109 1861,Lynx,4.08  
## 110 1862,Lynx,4.09  
## 111 1863,Lynx,14.33  
## 112 1864,Lynx,38.22  
## 113 1865,Lynx,60.78  
## 114 1866,Lynx,70.77  
## 115 1867,Lynx,72.77  
## 116 1868,Lynx,42.68  
## 117 1869,Lynx,16.39  
## 118 1870,Lynx,9.83  
## 119 1871,Lynx,5.8  
## 120 1872,Lynx,5.26  
## 121 1873,Lynx,18.91  
## 122 1874,Lynx,30.95  
## 123 1875,Lynx,31.18  
## 124 1876,Lynx,46.34  
## 125 1877,Lynx,45.77  
## 126 1878,Lynx,44.15  
## 127 1879,Lynx,36.33  
## 128 1880,Lynx,12.03  
## 129 1881,Lynx,12.6  
## 130 1882,Lynx,18.34  
## 131 1883,Lynx,35.14  
## 132 1884,Lynx,43.77  
## 133 1885,Lynx,65.69  
## 134 1886,Lynx,79.35  
## 135 1887,Lynx,51.65  
## 136 1888,Lynx,32.59  
## 137 1889,Lynx,22.45  
## 138 1890,Lynx,16.16  
## 139 1891,Lynx,14.12  
## 140 1892,Lynx,20.38  
## 141 1893,Lynx,33.33  
## 142 1894,Lynx,46  
## 143 1895,Lynx,51.41  
## 144 1896,Lynx,46.43  
## 145 1897,Lynx,33.68  
## 146 1898,Lynx,18.01  
## 147 1899,Lynx,8.86  
## 148 1900,Lynx,7.13  
## 149 1901,Lynx,9.47  
## 150 1902,Lynx,14.86  
## 151 1903,Lynx,31.47  
## 152 1904,Lynx,60.57  
## 153 1905,Lynx,63.51  
## 154 1906,Lynx,54.7  
## 155 1907,Lynx,6.3  
## 156 1908,Lynx,3.41  
## 157 1909,Lynx,5.44  
## 158 1910,Lynx,11.65  
## 159 1911,Lynx,20.35  
## 160 1912,Lynx,32.88  
## 161 1913,Lynx,39.55  
## 162 1914,Lynx,43.36  
## 163 1915,Lynx,40.83  
## 164 1916,Lynx,30.36  
## 165 1917,Lynx,17.18  
## 166 1918,Lynx,6.82  
## 167 1919,Lynx,3.19  
## 168 1920,Lynx,3.52  
## 169 1921,Lynx,9.94  
## 170 1922,Lynx,20.3  
## 171 1923,Lynx,31.99  
## 172 1924,Lynx,42.36  
## 173 1925,Lynx,49.08  
## 174 1926,Lynx,53.99  
## 175 1927,Lynx,52.25  
## 176 1928,Lynx,37.7  
## 177 1929,Lynx,19.14  
## 178 1930,Lynx,6.98  
## 179 1931,Lynx,8.31  
## 180 1932,Lynx,16.01  
## 181 1933,Lynx,24.82  
## 182 1934,Lynx,29.7  
## 183 1935,Lynx,35.4

A couple things went wrong here:

1. The data is imported as a single column - we want R to know that there are multiple columns delimited by commas
2. The header (Year,Species,Count) was imported as the first row of our data - we would instead like R to recognize that this is a header rather than a row of data

How do we fix this? Along with the essential argument of the file location, we can provide read.table with many **optional arguments**. Because there are many possibilities for what our second argument may be, we need to explicitly reference the name of our optional arguments.

Let’s inform R that our data is split by commas. We can do this using the sep (short for “separator”) argument.

read.table("hare\_lynx.csv", sep = ",")

## V1 V2 V3  
## 1 Year Species Count  
## 2 1845 Hare 19.58  
## 3 1846 Hare 19.6  
## 4 1847 Hare 19.61  
## 5 1848 Hare 11.99  
## 6 1849 Hare 28.04  
## 7 1850 Hare 58  
## 8 1851 Hare 74.6  
## 9 1852 Hare 75.09  
## 10 1853 Hare 88.48  
## 11 1854 Hare 61.28  
## 12 1855 Hare 74.67  
## 13 1856 Hare 88.06  
## 14 1857 Hare 68.51  
## 15 1858 Hare 32.19  
## 16 1859 Hare 12.64  
## 17 1860 Hare 21.49  
## 18 1861 Hare 30.35  
## 19 1862 Hare 2.18  
## 20 1863 Hare 152.65  
## 21 1864 Hare 148.36  
## 22 1865 Hare 85.81  
## 23 1866 Hare 41.41  
## 24 1867 Hare 14.75  
## 25 1868 Hare 2.28  
## 26 1869 Hare 5.91  
## 27 1870 Hare 9.95  
## 28 1871 Hare 10.44  
## 29 1872 Hare 70.64  
## 30 1873 Hare 50.12  
## 31 1874 Hare 50.13  
## 32 1875 Hare 101.25  
## 33 1876 Hare 97.12  
## 34 1877 Hare 86.51  
## 35 1878 Hare 72.17  
## 36 1879 Hare 38.32  
## 37 1880 Hare 10.11  
## 38 1881 Hare 7.74  
## 39 1882 Hare 9.67  
## 40 1883 Hare 43.12  
## 41 1884 Hare 52.21  
## 42 1885 Hare 134.85  
## 43 1886 Hare 134.86  
## 44 1887 Hare 103.79  
## 45 1888 Hare 46.1  
## 46 1889 Hare 15.03  
## 47 1890 Hare 24.2  
## 48 1891 Hare 41.65  
## 49 1892 Hare 52.34  
## 50 1893 Hare 53.78  
## 51 1894 Hare 70.4  
## 52 1895 Hare 85.81  
## 53 1896 Hare 56.69  
## 54 1897 Hare 16.59  
## 55 1898 Hare 6.16  
## 56 1899 Hare 2.3  
## 57 1900 Hare 12.82  
## 58 1901 Hare 4.72  
## 59 1902 Hare 4.73  
## 60 1903 Hare 37.22  
## 61 1904 Hare 69.72  
## 62 1905 Hare 57.78  
## 63 1906 Hare 28.68  
## 64 1907 Hare 23.37  
## 65 1908 Hare 21.54  
## 66 1909 Hare 26.34  
## 67 1910 Hare 53.1  
## 68 1911 Hare 68.48  
## 69 1912 Hare 75.58  
## 70 1913 Hare 57.92  
## 71 1914 Hare 40.97  
## 72 1915 Hare 24.95  
## 73 1916 Hare 12.59  
## 74 1917 Hare 4.97  
## 75 1918 Hare 4.5  
## 76 1919 Hare 11.21  
## 77 1920 Hare 56.6  
## 78 1921 Hare 69.63  
## 79 1922 Hare 77.74  
## 80 1923 Hare 80.53  
## 81 1924 Hare 73.38  
## 82 1925 Hare 36.93  
## 83 1926 Hare 4.64  
## 84 1927 Hare 2.54  
## 85 1928 Hare 1.8  
## 86 1929 Hare 2.39  
## 87 1930 Hare 4.23  
## 88 1931 Hare 19.52  
## 89 1932 Hare 82.11  
## 90 1933 Hare 89.76  
## 91 1934 Hare 81.66  
## 92 1935 Hare 15.76  
## 93 1845 Lynx 30.09  
## 94 1846 Lynx 45.15  
## 95 1847 Lynx 49.15  
## 96 1848 Lynx 39.52  
## 97 1849 Lynx 21.23  
## 98 1850 Lynx 8.42  
## 99 1851 Lynx 5.56  
## 100 1852 Lynx 5.08  
## 101 1853 Lynx 10.17  
## 102 1854 Lynx 19.6  
## 103 1855 Lynx 32.91  
## 104 1856 Lynx 34.38  
## 105 1857 Lynx 29.59  
## 106 1858 Lynx 21.3  
## 107 1859 Lynx 13.69  
## 108 1860 Lynx 7.65  
## 109 1861 Lynx 4.08  
## 110 1862 Lynx 4.09  
## 111 1863 Lynx 14.33  
## 112 1864 Lynx 38.22  
## 113 1865 Lynx 60.78  
## 114 1866 Lynx 70.77  
## 115 1867 Lynx 72.77  
## 116 1868 Lynx 42.68  
## 117 1869 Lynx 16.39  
## 118 1870 Lynx 9.83  
## 119 1871 Lynx 5.8  
## 120 1872 Lynx 5.26  
## 121 1873 Lynx 18.91  
## 122 1874 Lynx 30.95  
## 123 1875 Lynx 31.18  
## 124 1876 Lynx 46.34  
## 125 1877 Lynx 45.77  
## 126 1878 Lynx 44.15  
## 127 1879 Lynx 36.33  
## 128 1880 Lynx 12.03  
## 129 1881 Lynx 12.6  
## 130 1882 Lynx 18.34  
## 131 1883 Lynx 35.14  
## 132 1884 Lynx 43.77  
## 133 1885 Lynx 65.69  
## 134 1886 Lynx 79.35  
## 135 1887 Lynx 51.65  
## 136 1888 Lynx 32.59  
## 137 1889 Lynx 22.45  
## 138 1890 Lynx 16.16  
## 139 1891 Lynx 14.12  
## 140 1892 Lynx 20.38  
## 141 1893 Lynx 33.33  
## 142 1894 Lynx 46  
## 143 1895 Lynx 51.41  
## 144 1896 Lynx 46.43  
## 145 1897 Lynx 33.68  
## 146 1898 Lynx 18.01  
## 147 1899 Lynx 8.86  
## 148 1900 Lynx 7.13  
## 149 1901 Lynx 9.47  
## 150 1902 Lynx 14.86  
## 151 1903 Lynx 31.47  
## 152 1904 Lynx 60.57  
## 153 1905 Lynx 63.51  
## 154 1906 Lynx 54.7  
## 155 1907 Lynx 6.3  
## 156 1908 Lynx 3.41  
## 157 1909 Lynx 5.44  
## 158 1910 Lynx 11.65  
## 159 1911 Lynx 20.35  
## 160 1912 Lynx 32.88  
## 161 1913 Lynx 39.55  
## 162 1914 Lynx 43.36  
## 163 1915 Lynx 40.83  
## 164 1916 Lynx 30.36  
## 165 1917 Lynx 17.18  
## 166 1918 Lynx 6.82  
## 167 1919 Lynx 3.19  
## 168 1920 Lynx 3.52  
## 169 1921 Lynx 9.94  
## 170 1922 Lynx 20.3  
## 171 1923 Lynx 31.99  
## 172 1924 Lynx 42.36  
## 173 1925 Lynx 49.08  
## 174 1926 Lynx 53.99  
## 175 1927 Lynx 52.25  
## 176 1928 Lynx 37.7  
## 177 1929 Lynx 19.14  
## 178 1930 Lynx 6.98  
## 179 1931 Lynx 8.31  
## 180 1932 Lynx 16.01  
## 181 1933 Lynx 24.82  
## 182 1934 Lynx 29.7  
## 183 1935 Lynx 35.4

Now we correctly have three columns of data! We now have to tell R that the first row is a header. There is an optional argument header which can be set to TRUE or FALSE (case sensitive - more on this later).

read.table("hare\_lynx.csv", sep = ",", header = TRUE)

## Year Species Count  
## 1 1845 Hare 19.58  
## 2 1846 Hare 19.60  
## 3 1847 Hare 19.61  
## 4 1848 Hare 11.99  
## 5 1849 Hare 28.04  
## 6 1850 Hare 58.00  
## 7 1851 Hare 74.60  
## 8 1852 Hare 75.09  
## 9 1853 Hare 88.48  
## 10 1854 Hare 61.28  
## 11 1855 Hare 74.67  
## 12 1856 Hare 88.06  
## 13 1857 Hare 68.51  
## 14 1858 Hare 32.19  
## 15 1859 Hare 12.64  
## 16 1860 Hare 21.49  
## 17 1861 Hare 30.35  
## 18 1862 Hare 2.18  
## 19 1863 Hare 152.65  
## 20 1864 Hare 148.36  
## 21 1865 Hare 85.81  
## 22 1866 Hare 41.41  
## 23 1867 Hare 14.75  
## 24 1868 Hare 2.28  
## 25 1869 Hare 5.91  
## 26 1870 Hare 9.95  
## 27 1871 Hare 10.44  
## 28 1872 Hare 70.64  
## 29 1873 Hare 50.12  
## 30 1874 Hare 50.13  
## 31 1875 Hare 101.25  
## 32 1876 Hare 97.12  
## 33 1877 Hare 86.51  
## 34 1878 Hare 72.17  
## 35 1879 Hare 38.32  
## 36 1880 Hare 10.11  
## 37 1881 Hare 7.74  
## 38 1882 Hare 9.67  
## 39 1883 Hare 43.12  
## 40 1884 Hare 52.21  
## 41 1885 Hare 134.85  
## 42 1886 Hare 134.86  
## 43 1887 Hare 103.79  
## 44 1888 Hare 46.10  
## 45 1889 Hare 15.03  
## 46 1890 Hare 24.20  
## 47 1891 Hare 41.65  
## 48 1892 Hare 52.34  
## 49 1893 Hare 53.78  
## 50 1894 Hare 70.40  
## 51 1895 Hare 85.81  
## 52 1896 Hare 56.69  
## 53 1897 Hare 16.59  
## 54 1898 Hare 6.16  
## 55 1899 Hare 2.30  
## 56 1900 Hare 12.82  
## 57 1901 Hare 4.72  
## 58 1902 Hare 4.73  
## 59 1903 Hare 37.22  
## 60 1904 Hare 69.72  
## 61 1905 Hare 57.78  
## 62 1906 Hare 28.68  
## 63 1907 Hare 23.37  
## 64 1908 Hare 21.54  
## 65 1909 Hare 26.34  
## 66 1910 Hare 53.10  
## 67 1911 Hare 68.48  
## 68 1912 Hare 75.58  
## 69 1913 Hare 57.92  
## 70 1914 Hare 40.97  
## 71 1915 Hare 24.95  
## 72 1916 Hare 12.59  
## 73 1917 Hare 4.97  
## 74 1918 Hare 4.50  
## 75 1919 Hare 11.21  
## 76 1920 Hare 56.60  
## 77 1921 Hare 69.63  
## 78 1922 Hare 77.74  
## 79 1923 Hare 80.53  
## 80 1924 Hare 73.38  
## 81 1925 Hare 36.93  
## 82 1926 Hare 4.64  
## 83 1927 Hare 2.54  
## 84 1928 Hare 1.80  
## 85 1929 Hare 2.39  
## 86 1930 Hare 4.23  
## 87 1931 Hare 19.52  
## 88 1932 Hare 82.11  
## 89 1933 Hare 89.76  
## 90 1934 Hare 81.66  
## 91 1935 Hare 15.76  
## 92 1845 Lynx 30.09  
## 93 1846 Lynx 45.15  
## 94 1847 Lynx 49.15  
## 95 1848 Lynx 39.52  
## 96 1849 Lynx 21.23  
## 97 1850 Lynx 8.42  
## 98 1851 Lynx 5.56  
## 99 1852 Lynx 5.08  
## 100 1853 Lynx 10.17  
## 101 1854 Lynx 19.60  
## 102 1855 Lynx 32.91  
## 103 1856 Lynx 34.38  
## 104 1857 Lynx 29.59  
## 105 1858 Lynx 21.30  
## 106 1859 Lynx 13.69  
## 107 1860 Lynx 7.65  
## 108 1861 Lynx 4.08  
## 109 1862 Lynx 4.09  
## 110 1863 Lynx 14.33  
## 111 1864 Lynx 38.22  
## 112 1865 Lynx 60.78  
## 113 1866 Lynx 70.77  
## 114 1867 Lynx 72.77  
## 115 1868 Lynx 42.68  
## 116 1869 Lynx 16.39  
## 117 1870 Lynx 9.83  
## 118 1871 Lynx 5.80  
## 119 1872 Lynx 5.26  
## 120 1873 Lynx 18.91  
## 121 1874 Lynx 30.95  
## 122 1875 Lynx 31.18  
## 123 1876 Lynx 46.34  
## 124 1877 Lynx 45.77  
## 125 1878 Lynx 44.15  
## 126 1879 Lynx 36.33  
## 127 1880 Lynx 12.03  
## 128 1881 Lynx 12.60  
## 129 1882 Lynx 18.34  
## 130 1883 Lynx 35.14  
## 131 1884 Lynx 43.77  
## 132 1885 Lynx 65.69  
## 133 1886 Lynx 79.35  
## 134 1887 Lynx 51.65  
## 135 1888 Lynx 32.59  
## 136 1889 Lynx 22.45  
## 137 1890 Lynx 16.16  
## 138 1891 Lynx 14.12  
## 139 1892 Lynx 20.38  
## 140 1893 Lynx 33.33  
## 141 1894 Lynx 46.00  
## 142 1895 Lynx 51.41  
## 143 1896 Lynx 46.43  
## 144 1897 Lynx 33.68  
## 145 1898 Lynx 18.01  
## 146 1899 Lynx 8.86  
## 147 1900 Lynx 7.13  
## 148 1901 Lynx 9.47  
## 149 1902 Lynx 14.86  
## 150 1903 Lynx 31.47  
## 151 1904 Lynx 60.57  
## 152 1905 Lynx 63.51  
## 153 1906 Lynx 54.70  
## 154 1907 Lynx 6.30  
## 155 1908 Lynx 3.41  
## 156 1909 Lynx 5.44  
## 157 1910 Lynx 11.65  
## 158 1911 Lynx 20.35  
## 159 1912 Lynx 32.88  
## 160 1913 Lynx 39.55  
## 161 1914 Lynx 43.36  
## 162 1915 Lynx 40.83  
## 163 1916 Lynx 30.36  
## 164 1917 Lynx 17.18  
## 165 1918 Lynx 6.82  
## 166 1919 Lynx 3.19  
## 167 1920 Lynx 3.52  
## 168 1921 Lynx 9.94  
## 169 1922 Lynx 20.30  
## 170 1923 Lynx 31.99  
## 171 1924 Lynx 42.36  
## 172 1925 Lynx 49.08  
## 173 1926 Lynx 53.99  
## 174 1927 Lynx 52.25  
## 175 1928 Lynx 37.70  
## 176 1929 Lynx 19.14  
## 177 1930 Lynx 6.98  
## 178 1931 Lynx 8.31  
## 179 1932 Lynx 16.01  
## 180 1933 Lynx 24.82  
## 181 1934 Lynx 29.70  
## 182 1935 Lynx 35.40

Looks good! Now, a couple nuances:

1. The order **named arguments** (here, sep and header doesn’t matter. The following lines of code are equivalent:

read.table("hare\_lynx.csv", sep = ",", header = TRUE)  
read.table("hare\_lynx.csv", header = TRUE, sep = ",")

1. For better legibility, we can add line breaks after commas, like so:

read.table("hare\_lynx.csv",   
 sep = ",",   
 header = TRUE)

This does not impact how data is evaluated.

# Authors

| Credits | Names |
| --- | --- |
| **Pedagogy** |  |
| Instructor | [Andrew Bortvin](https://andrew-bortvin.github.io/) |
| Content Author | [Andrew Bortvin](https://andrew-bortvin.github.io/) |
| **Website** |  |
| Template | [Jeff Leek](https://jtleek.com/) & [The Johns Hopkins Data Science Lab](https://jhudatascience.org/index.html) |
| Design Inspiration | [Stephanie Yan](https://stephaniemyan.github.io/) & [Ali Madooei](https://engineering.jhu.edu/faculty/ali-madooei/) & [JHU Data Structures](https://cs226sp22.github.io/) |
| **Funding** |  |
| JHU Dean’s Teaching Fellowship |  |

## ─ Session info ───────────────────────────────────────────────────────────────  
## setting value  
## version R version 4.3.2 (2023-10-31)  
## os Ubuntu 22.04.4 LTS  
## system x86\_64, linux-gnu  
## ui X11  
## language (EN)  
## collate en\_US.UTF-8  
## ctype en\_US.UTF-8  
## tz Etc/UTC  
## date 2024-08-14  
## pandoc 3.1.1 @ /usr/local/bin/ (via rmarkdown)  
##   
## ─ Packages ───────────────────────────────────────────────────────────────────  
## package \* version date (UTC) lib source  
## askpass 1.2.0 2023-09-03 [1] RSPM (R 4.3.0)  
## bookdown 0.39.1 2024-06-11 [1] Github (rstudio/bookdown@f244cf1)  
## cachem 1.0.8 2023-05-01 [1] RSPM (R 4.3.0)  
## cli 3.6.2 2023-12-11 [1] RSPM (R 4.3.0)  
## devtools 2.4.5 2022-10-11 [1] RSPM (R 4.3.0)  
## digest 0.6.34 2024-01-11 [1] RSPM (R 4.3.0)  
## ellipsis 0.3.2 2021-04-29 [1] RSPM (R 4.3.0)  
## evaluate 0.23 2023-11-01 [1] RSPM (R 4.3.0)  
## fansi 1.0.6 2023-12-08 [1] RSPM (R 4.3.0)  
## fastmap 1.1.1 2023-02-24 [1] RSPM (R 4.3.0)  
## fs 1.6.3 2023-07-20 [1] RSPM (R 4.3.0)  
## glue 1.7.0 2024-01-09 [1] RSPM (R 4.3.0)  
## hms 1.1.3 2023-03-21 [1] RSPM (R 4.3.0)  
## htmltools 0.5.7 2023-11-03 [1] RSPM (R 4.3.0)  
## htmlwidgets 1.6.4 2023-12-06 [1] RSPM (R 4.3.0)  
## httpuv 1.6.14 2024-01-26 [1] RSPM (R 4.3.0)  
## httr 1.4.7 2023-08-15 [1] RSPM (R 4.3.0)  
## jsonlite 1.8.8 2023-12-04 [1] RSPM (R 4.3.0)  
## knitr 1.47.3 2024-06-11 [1] Github (yihui/knitr@e1edd34)  
## later 1.3.2 2023-12-06 [1] RSPM (R 4.3.0)  
## lifecycle 1.0.4 2023-11-07 [1] RSPM (R 4.3.0)  
## magrittr 2.0.3 2022-03-30 [1] RSPM (R 4.3.0)  
## memoise 2.0.1 2021-11-26 [1] RSPM (R 4.3.0)  
## mime 0.12 2021-09-28 [1] RSPM (R 4.3.0)  
## miniUI 0.1.1.1 2018-05-18 [1] RSPM (R 4.3.0)  
## openssl 2.1.1 2023-09-25 [1] RSPM (R 4.3.0)  
## ottrpal 1.2.1 2024-06-11 [1] Github (jhudsl/ottrpal@828539f)  
## pillar 1.9.0 2023-03-22 [1] RSPM (R 4.3.0)  
## pkgbuild 1.4.3 2023-12-10 [1] RSPM (R 4.3.0)  
## pkgconfig 2.0.3 2019-09-22 [1] RSPM (R 4.3.0)  
## pkgload 1.3.4 2024-01-16 [1] RSPM (R 4.3.0)  
## profvis 0.3.8 2023-05-02 [1] RSPM (R 4.3.0)  
## promises 1.2.1 2023-08-10 [1] RSPM (R 4.3.0)  
## purrr 1.0.2 2023-08-10 [1] RSPM (R 4.3.0)  
## R6 2.5.1 2021-08-19 [1] RSPM (R 4.3.0)  
## Rcpp 1.0.12 2024-01-09 [1] RSPM (R 4.3.0)  
## readr 2.1.5 2024-01-10 [1] RSPM (R 4.3.0)  
## remotes 2.4.2.1 2023-07-18 [1] RSPM (R 4.3.0)  
## rlang 1.1.4 2024-06-04 [1] CRAN (R 4.3.2)  
## rmarkdown 2.27.1 2024-06-11 [1] Github (rstudio/rmarkdown@e1c93a9)  
## rstudioapi 0.15.0 2023-07-07 [1] RSPM (R 4.3.0)  
## sessioninfo 1.2.2 2021-12-06 [1] RSPM (R 4.3.0)  
## shiny 1.8.0 2023-11-17 [1] RSPM (R 4.3.0)  
## stringi 1.8.3 2023-12-11 [1] RSPM (R 4.3.0)  
## stringr 1.5.1 2023-11-14 [1] RSPM (R 4.3.0)  
## tibble 3.2.1 2023-03-20 [1] CRAN (R 4.3.2)  
## tzdb 0.4.0 2023-05-12 [1] RSPM (R 4.3.0)  
## urlchecker 1.0.1 2021-11-30 [1] RSPM (R 4.3.0)  
## usethis 2.2.3 2024-02-19 [1] RSPM (R 4.3.0)  
## utf8 1.2.4 2023-10-22 [1] RSPM (R 4.3.0)  
## vctrs 0.6.5 2023-12-01 [1] RSPM (R 4.3.0)  
## xfun 0.44.4 2024-06-11 [1] Github (yihui/xfun@9da62cc)  
## xml2 1.3.6 2023-12-04 [1] RSPM (R 4.3.0)  
## xtable 1.8-4 2019-04-21 [1] RSPM (R 4.3.0)  
## yaml 2.3.8 2023-12-11 [1] RSPM (R 4.3.0)  
##   
## [1] /usr/local/lib/R/site-library  
## [2] /usr/local/lib/R/library  
##   
## ──────────────────────────────────────────────────────────────────────────────