Quantitative Text Analysis for Linguistics

Reproducible Research using R

Jerid Francom

August 2, 2023

Table of contents

Welcome

The goal of this textbook is to provide readers with foundational knowledge and practical skills in quantitative text analysis using the R programming language.

By the end of this textbook, readers will be able to identify, interpret and evaluate data analysis procedures and results to support research questions within language science. Additionally, readers will gain experience in designing and implementing research projects that involve processing and analyzing textual data employing modern programming strategies. This textbook aims to instill a strong sense of reproducible research practices, which are critical for promoting transparency, verification, and sharing of research findings.

This textbook is geared towards advanced undergraduates, graduate students, and researchers looking to expand their methodological toolbox. It assumes no prior knowledge of programming or quantitative methods and prioritizes practical application and intuitive understanding over technical details.

About the author

Dr. Jerid Francom is Associate Professor of Spanish and Linguistics at Wake Forest University. His research focuses on the use of language corpora from a variety of sources (news, social media, and other internet sources) to better understand the linguistic and cultural similarities and differences between language varieties for both scholarly and pedagogical projects. He has published on topics including the development, annotation, and evaluation of linguistic corpora and analyzed corpora through corpus, psycholinguistic, and computational methodologies. He also has experience working with and teaching statistical programming with R.

License

[1] https://francojc.github.io/

Credits

Font Awesome Icons[2] are SIL OFL 1.1 Licensed

Acknowledgements

The development of this book has benefited from the generous feedback from the following people: Andrea Bowling, Caroline Brady, Declan Golsen, Asya Little, Claudia Valdez, *(add your name here!)*. As always, any errors or omissions are my own.

Build information

This textbook was built with the `quarto` package (Allaire 2022) and the `bookdown` package (Xie 2023a) for R. The source code for this book is available on GitHub[3].

This version of the textbook was built with R version 4.3.1 (2023-06-16) on macOS Ventura 13.4.1 with the following packages:

package	version	source
dplyr	1.1.2	CRAN (R 4.3.0)
forcats	1.0.0	CRAN (R 4.3.0)
ggplot2	3.4.2	CRAN (R 4.3.0)
here	1.0.1	CRAN (R 4.3.0)
knitr	1.43	CRAN (R 4.3.0)
lubridate	1.9.2	CRAN (R 4.3.0)
purrr	1.0.1	CRAN (R 4.3.0)
qtalrkit	0.0.2.0	Github (qtalr/qtalrkit@34a75e96a47e624b5af5f253c1d74f1659589ceb)
readr	2.1.4	CRAN (R 4.3.0)
readtext	0.90	CRAN (R 4.3.0)
rmarkdown	2.23	CRAN (R 4.3.1)
stringr	1.5.0	CRAN (R 4.3.0)
tadr	0.1.2	local (/Users/francojc/Documents/Academic/Research/Projects/1Active/tad/tadr)
tibble	3.2.1	CRAN (R 4.3.0)
tidyr	1.3.0	CRAN (R 4.3.0)
tidytext	0.4.1	CRAN (R 4.3.0)
tidyverse	2.0.0	CRAN (R 4.3.1)

[2]https://fontawesome.com/
[3]https://github.com/qtalr/book

package	version	source
webshot	0.5.5	CRAN (R 4.3.1)

Preface

Draft

Ready for review.

> The journey of a thousand miles begins with one step.
>
> — Lao Tzu

Outcomes

- Comprehend the book's rationale, learning goals, and pedagogical approach.
- Navigate and engage with the book's structure and content effectively.
- Interpret conventions used in the book reliably.
- Set up the computing environment and utilize textbook and support resources for an optimal learning experience.

The purpose of this chapter is to present the rationale behind this textbook, outline the key learning objectives, describe the pedagogical approach, and identify the intended audience. Additionally, this chapter will provide readers with a guide to the book's structure and the scope of its content, as well as instructions for the instructor and a summary of supporting resources available. Finally, this chapter will provide readers with information on setting up their computing environment and where to seek support.

Rationale

Data science, an interdisciplinary field that combines knownledge and skills from statistics, computer science, and domain-specific expertise to extract meaningful insight from structured and unstructured data, has emerged as an exciting and rapidly growing field in recent years, driven in large part by the increase in computing power available to the average individual and the abundance of electronic data now available through the internet. These advances have become an integral part of the modern scientific landscape, with data-driven insights now being used to inform decision-making in a wide variety of academic fields, including linguistics and language-related disciplines.

This textbook seeks to meet this growing demand by providing an introduction to the fundamental concepts and practical programming skills from data science applied to the task of quantitative text analysis. It is intended primarily for undergraduate students, but may also be useful for graduates and researchers seeking to expand their methodological toolbox. The textbook takes a pedagogical approach which assumes no prior experience with statistics or programming, making it an accessible resource for novices beginning their exploration of quantitative text analysis methods.

Aims

The overarching goal of this textbook is to provide readers with foundational knowledge and practical skills to conduct and evaluate quantitative text analysis using the R programming language and other open source tools and technologies. The specific aims are to develop the reader's proficiency in three main areas:

- **Data literacy:** Identify, interpret and evaluate data analysis procedures and results

  Throughout this textbook we will explore topics which will help you understand how data analysis methods derive insight from data. In this process you will be encouraged to critically evaluate connections across linguistic and language-related disciplines using data analysis knowledge and skills. Data literacy is an invaluable skillset for academics and professionals but also is an indispensable aptitude for in the 21st century citizens to navigate and actively participate in the 'Information Age' in which we live (Carmi et al. 2020).

- **Research skills:** Design, implement, and communicate quantitative research

  This aim does not differ significantly, in spirit, from common learning outcomes in a research methods course. However, working with text will incur a series of key steps in the selection, collection, and preparation of the data that are unique to text analysis projects. In addition, I will stress the importance of research documentation and creating reproducible research as an integral part of modern scientific inquiry (Buckheit and Donoho 1995).

- **Programming skills:** Apply programmatic strategies to develop and collaborate on reproducible research projects

  Modern data analysis, and by extension, text analysis is conducted using programming. There are various key reasons for this: a programming approach (1) affords researchers unlimited research freedom –if you can envision it, you can program it, (2) underlies well-documented and reproducible research (Gandrud 2015), and (3) invites researchers to engage more intimately with the data and the methods for analysis.

These aims are important for linguistics students because they provide a foundation for concepts and in the skills required to succeed in the rapidly evolving landscape of 21st-century research. These abilities enable researchers to evaluate and conduct high-quality empirical investigation across linguistic fields on a wide variety of topics. Moreover, these skills go beyond linguistics research; they are widely applicable across many disciplines where quantitative data analysis and programming are becoming increasingly important. Thus, this textbook provides students with a comprehensive introduction to quantitative text analysis that is relevant to linguistics research and that equips them with valuable skills for their future careers.

Approach

The approach taken in this textbook is designed to accomodate linguistics students and researchers with little to no prior experience with programming or quantitative methods. With this in mind the objective is connect conceptual understanding with practical application. Real-world data and research tasks relevant to linguistics are used thoughtout the book to provide context and to motivate the learning process[4]. Furthermore, as an introduction to the field, the textbook focuses on the most common and fundamental methods and techniques for quantitative text analysis and prioritizes breadth over depth and intuitive understanding over technical explanations. On the programming side, the Tidyverse approach to programming in R will be adopted. This approach provides a consistent syntax across different packages and is known for its legibility, making it easier for readers to understand and write code. Together, these strategies form an approach that is intended to provide readers with an accessible resource to gain a foothold in the field and to equip them with the knowledge and skills to apply quantitative text analysis in their own research.

Structure

The aims and approach described above is reflected in the overall structure of the book and each chapter.

Book level

At the book level, there are five interdependent parts:

Part I "Orientation" provides the necessary background knowledge to situate quantitative text analysis in the wider context of data analysis and linguistic research and to provide a clearer picture of what text analysis entails and its range of applications.

[4]Research data and questions are primarily based on English for wide accessibility as it is the *de facto* language of academics and research. However, the methods and techniques presented in this textbook are applicable to many other languages.

The subsequent parts are directly aligned with the data analysis process. The building blocks of this process are reflected in 'Data to Insight Hierarchy (DIKI)' visualized in Figure 1[5].

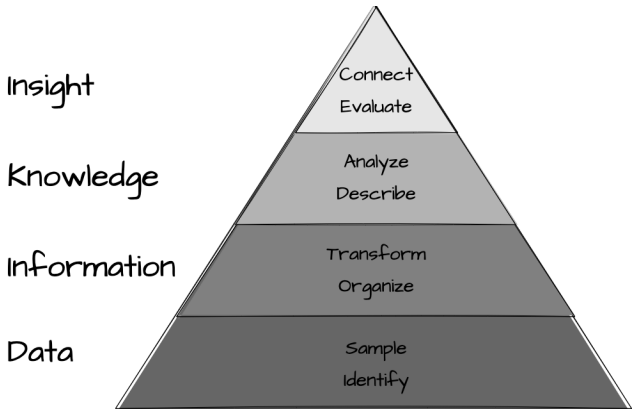


Figure 1: Data to Insight Hierarchy (DIKI)

The DIKI Hierarchy highlights the stages and intermediate steps required to derive insight from data. Part II "Foundations" provides a conceptual introduction to the DIKI Hierarchy and establishes foundational knowledge about data, information, knowledge, and insight which is fundamental to developing a viable research plan.

Parts III "Preparation" and IV "Analysis" focus on the implementation process. Part III covers the steps involved in preparing data for analysis, including data acquisition, curation, and transformation. Part IV covers the steps involved in conducting analysis, including exploratory, predictive, and inferential data analysis.

The final part, Part V "Communication", covers the final stage of the data analysis process, which is to communicate the results of the analysis. This includes the structure and content of research reports as well as the process of publishing, sharing, and collaborating on research.

Chapter level

At the chapter level, both conceptual and programming skills are developed in stages[6]. The chapter-level structure is consistent across chapters and can be seen in Table 2.

[5]Adapted from Ackoff (1989) and Rowley (2007).

[6]These stages attempt to capture the general progression of learning reflected in Bloom's Taxonomy (see Krathwohl (2002) for a description and revision).

Table 2: The general structure of a chapter including: the component, its purpose, where to find the resource, and the target learning stage.

Component	Purpose	Resource	Stage
Outcomes	Identify the learning objectives for the chapter	Textbook	Introduction
Overview	Provide a brief introduction to the chapter topic	Textbook	Introduction
Coding Lessons	Teach programming techniques with hands-on interactive exercises	GitHub	Skills
Content	Combine conceptual discussions and programming skills, incorporating thought-provoking questions, relevant studies, and advanced topic references	Textbook	Knowledge
Recipes	Offer step-by-step programming examples related to the chapter	Resources website	Comprehension
Labs	Allow readers to apply chapter-specific concepts and techniques	GitHub	Application
Summary	Review the key concepts and skills covered in the chapter	Textbook	Review
Questions	Assess and expand the reader's knowledge and abilities	Textbook	Assessment

Each chapter will begin with a list of key learning outcomes followed by a brief introduction to the chapter's content. The goal is to orient the reader to the chapter. Next there will be a prompt to complete the interactive coding lesson(s) to introduce reader's to key programming concepts related to the chapter though hands-on experience and then the main content of the chapter will follow. The content will be a combination of conceptual discussions and programming skills, incorporating thought-provoking questions ('Consider this'), relevant studies ('Case study'), and advanced topic references ('Dive deeper'). Together these components form the skills and knowledge phase. The next phase is the application phase. This phase will include step-by-step programming demonstrations related to the chapter (Recipes) and lab exercises that allow readers to apply their knowledge and skills chapter-related tasks. Finally the chapters conclude with a summary of the key concepts and skills covered in the chapter and a set of questions to assess and expand the reader's knowledge and abilities.

Resources

There are three main resources available to support the aims and approach of this textbook. Firstly, the textbook itself provides prose discussion, figures/ tables, R code, case studies, and thought and practical exercises. Secondly, there is a companion R package called `qtalrkit`

(Francom 2023), which includes functions for accessing data and datasets, as well as various useful functions developed specifically for this textbook. In addition, there is a comprehensive website Quantitative Text Analysis for Linguistics Resources[7](qtalr website) that includes programming tutorials and demonstrations to enhance the reader's recognition of how programming strategies are implemented. Finally, a GitHub repository[8] is provided which contains both a set of interactive R programming lessons (Swirl) and lab exercises designed to guide the reader through practical hands-on programming applications. The companion `qtalrkit` package and the GitHub repository are both under active development and will be updated regularly to ensure that supplementary materials remain relevant to the content of the text[9].

Getting started

Before jumping in to this and subsequent chapter's textbook activities, it is important to prepare your computing environment and understand how to take advantage of the resources available, both those directly and indirectly associated with the textbook.

R and IDEs

Programming is the backbone for modern quantitative research. Among the many programming languages available, R is a popular open-source language and software environment for statistical computing. R is popular with statisticians and has been adopted as the *de facto* language by many other fields in natural and social sciences, including linguistics. It is freely downloadable from The R Project for Statistical Programming[10] website and is available for macOS, Linux, and Windows[11] operating systems.

Successfully installing R is rarely the last step in setting up your R-enabled computing environment. The majority of R users also install an **integrated development environment** (IDE). An IDE, such as RStudio[12] or Visual Studio Code[13], provide a **graphical user interface** (GUI) for working with R. In effect, IDEs provide a dashboard for working with R and are designed to make it easier to write and execute R code. IDEs also provide a number of other useful features such as syntax highlighting, code completion, and debugging. IDEs are not required to work with R but they are *highly* recommended.

Choosing to install R and an IDE on your personal computer, which is know as your **local environment**, is not the only option to work with R. You can also choose to work with R in the

[7]https://qtalr.github.io/qtalrkit/
[8]https://github.com/qtalr
[9]Errata for the textbook is found on the qtalr website.
[10]https://www.r-project.org/
[11]https://cloud.r-project.org/
[12]https://posit.co/products/open-source/rstudio/
[13]https://code.visualstudio.com/

cloud, a **remote environment**. There are a number of cloud-based options for working with R, including RStudio Cloud[14] and Microsoft Azure[15]. These options provide a pre-configured R environment that you can access from any computer with an internet connection. The advantage of working in the cloud is that you do not need to install R or an IDE on your local computer. The disadvantage is that you will need to be connected to the internet to work with R and the free tiers for these services are limited.

If you are new to R, you may want to consider working in the cloud to get started. If you plan to continue to work with R in the future, you will most likely want to install R and an IDE on your local computer or explore using a **virtual environment**. Virtual environments, such as Docker[16], provide a way to use a pre-configured computing environment or create your own that you can share with others. Virtual environments are a good option if you want to ensure that everyone in your research group is working with the same computing environment. Pre-configured virtual environments exist for R through the Rocker project[17] and can be used locally or in the cloud.

There are trade-offs in terms of cost, convenience, and flexibility when choosing to work with R in a local, remote, or virtual environment. The choice is yours and you can always change your mind later. The important thing is to get started and begin learning R. Furthermore, any of the approaches described here will be compatible with this textbook.

For more information and instructions on setting up an R environment consult the following guides.

Guides

- Installing R[a]
- Choosing and setting up an IDE[b]
- Working with R in remote and virtual environments[c]

[a] https://qtalr.github.io/qtalrkit/articles/guide-0.html
[b] https://qtalr.github.io/qtalrkit/articles/guide-1.html
[c] https://qtalr.github.io/qtalrkit/articles/guide-2.html

R packages

As you progress in your R programming experience, you'll find yourself leveraging code from other R users, which is typically provided as packages. Packages are sets of functions and/or datasets that are freely accessible for download, designed to perform a specific set of interrelated tasks. They enhance the capabilities of R. Official R packages can be found in repositories

[14] https://www.rstudio.com/products/cloud/
[15] https://learn.microsoft.com/en-us/azure/architecture/data-guide/technology-choices/r-developers-guide
[16] https://www.docker.com/
[17] https://rocker-project.org/

like CRAN[18] (Comprehensive R Archive Network), while other packages can be obtained from code-sharing platforms such as GitHub[19].

Consider this

The Comprehensive R Archive Network (CRAN) includes groupings of popular packages related to a given applied programming task called Task Views[a]. Explore the available CRAN Task Views listings. Note the variety of areas (tasks) that are covered in this listing. Now explore in more detail one of the following task views which are directly related to topics covered in this textbook noting the associated packages and their descriptions: (1) Cluster, (2) MachineLearning, (3) NaturalLanguageProcessing, or (4) ReproducibleResearch.

[a]https://cran.r-project.org/web/views/

You will download a number of packages at different stages of this textbook, but there is a set of packages that will be key to have from the get go. Once you have access to a working R/RStudio environment, you can proceed to install the following packages.

Install the following packages from CRAN.

- `tidyverse` (Wickham 2023)
- `rmarkdown` (Allaire et al. 2023)
- `quarto` (Allaire 2022)
- `tinytex` (Xie 2023b)
- `devtools` (Wickham et al. 2022)
- `usethis` (Wickham, Bryan, et al. 2023)
- `swirl` (Kross et al. 2020)

You can do this by running the following code in an R console:

```
 # install key packages from CRAN
install.packages(c("tidyverse", "rmarkdown", "quarto", "tinytex", "devtools",
↪  "usethis", "swirl"))
```

For instructions on how to install the `qtalrkit` package from GitHub and download and use the interactive R programming lessons for this textbook, see the following guides.

Guides

- Getting started[a]

[a]https://qtalr.github.io/qtalrkit/articles/qtalrkit.html

[18]https://cran.r-project.org/
[19]https://github.com/

Git and GitHub

GitHub[20] is a code sharing website. Modern computing is highly collaborative and GitHub is a very popular platform for sharing and collaborating on coding projects. The lab exercises for this textbook[21] are shared on GitHub. To access and complete these exercises you will need to sign up for a (free) GitHub account[22] and then set up the version control software `git` on your computing environment. `git` is the conduit to interfacing GitHub and for many `git` will already be installed on your computer (or cloud computing environment).

For more information and instructions on setting up version control consult the following guide.

Guides

- Setting up Git and GitHub[a]

[a]https://qtalr.github.io/qtalrkit/articles/guide-3.html

Getting help

The technologies employed in this approach to text analysis will include a somewhat steep learning curve. And in all honesty, the learning never stops! Both seasoned programmers and beginners alike need assistance. Fortunately there is a very large community of programmers who have developed many official support resources and who actively contribute to official and unofficial discussion forums. Together these resources provide many avenues for overcoming challenges.

In Table 3, I provide a list of steps for seeking help with R.

Table 3: Recommended order for seeking help with R.

Order	Resource	Description
1	Official R Documentation	Access the official documentation by running `help(package = "package_name")` in an R console. Use the `?` operator followed by the package or function name. Check out available Vignettes by running `browseVignettes("package_name")`.
2	Web Search	Look for package documentation and vignettes on the web. A popular site for this is R-Universe.

[20]https://github.com/

[21]https://github.com/stars/francojc/lists/labs

[22]https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home

Order	Resource	Description
3	RStudio IDE Help Toolbar	If you're using RStudio IDE, use the "Help" toolbar menu. It provides links to help resources, guides, and manuals.
4	Online Discussion Forums	Sites like Stack Overflow and RStudio Community are great platforms where the programming community asks and answers questions related to real-world issues.
5	Post Questions with Reprex	When posting a question, especially those involving coding issues or errors, provide enough background and include a reproducible example (reprex) - a minimal piece of code that demonstrates your issue. This helps others understand and answer your question effectively.

The first place to look for help with R is the official documentation of the R package you are using. You can access this documentation by running `help(package = "package_name")` in an R console or using the `?` operator and then the package or function name. Many R packages often include "Vignettes" (long-form documentation and demonstrations). These can be accessed either by running `browseVignettes()` in an R console with the package name in quotes (e.g. `browseVignettes("tidyverse")`).

You can also search the web for package documentation and vignettes. A popular site for this purpose is R-Universe[23].

If you are using the RStudio IDE, the easiest and most convenient place to get help with either R or RStudio is through the RStudio "Help" toolbar menu. There you will find links to help resources, guides, and manuals.

There are a number of very popular discussion forum websites where the programming community asks and answers questions to real-world issues. These sites often have subsections dedicated to particular programming languages or software. The most popular of these sites is Stack Overflow[24]. There are also R-specific discussion forums such as RStudio Community[25].

If you post a question on one of these communities ensure that if your question involves some coding issue or error that you provide enough background such that the community will be able to help you. This is often referred to as a **reproducible example** or "reprex". A reprex is a minimal piece of code that demonstrates the issue you are having. It is a very useful tool for both asking and answering questions.

For information on how to create a reprex consult the following guide.

[23]https://r-universe.dev/search/
[24]https://stackoverflow.com/
[25]https://community.rstudio.com/

Guides

- Creating reproducible examples[a]

[a]https://qtalr.github.io/qtalrkit/articles/guide-4.html

The take-home message here is that you are not alone. There are many people world-wide that are learning to program and/ or contribute to the learning of others. The more you engage with these resources and communities the more successful your learning will be. As soon as you are able, pay it forward. Posting questions and offering answers helps the community and engages and refines your skills –a win-win.

Conventions

To facilitate the learning process, this textbook will employ a number of conventions. These conventions are intended to help the reader navigate the text and to signal the reader's attention to important concepts and information.

Prose

The following typographic conventions are used throughout the text:

- *Italics*
  - Filenames, file extensions, directory paths, and URLs.
- `Fixed-width`
  - Package names, function names, variable names, and in-line code including expressions and operators.
- **Bold**
  - Key concepts when first introduced.
- Linked text[26]
  - Links to internal and external resources, footnotes, and citations including references to R packages when first introduced.

[26]https://qtalr.github.io/qtalrkit/

Code blocks

More lengthy code will be presented in code blocks, as seen in Example 0.1.

Example 0.1.

```
# A function that takes a name and returns a greeting
greet <- function(name) { # function definition
  paste("Hello", name) # print greeting
} # end function definition

greet(name = "Jerid") # apply function to a name
```

```
> [1] "Hello Jerid"
```

There are a couple of things to note about the code in Example 0.1. First, it shows the code that is run in R as well as the ouput that is returned. The code will appear in a box and the output will appear below the box. Both code and output will appear in fixed-width font. Output which is text will be prefixed with `>`. Second, the `#` symbol is used to signal a **code comment**, a human-facing description. Everything right of a `#` is not run as code. In this textbook you will see code comments above code on a separate line and to the right of code on the same line. It is good practice to comment your code to enhance readability and to help others understand what your code is doing.

All figures, tables, and images in this textbook are generated by code blocks but only code for those elements that are relevant for discussion will be shown. However, if you wish to see the code for any element in this textbook, you can visit the GitHub repository https://qtalr.github.io/book/.

When a reference to a file and its contents is made, it will appear as in File 0.1.

File 0.1 `example.R`: Example R script

```
# Load libraries
library(tidyverse)

# Add 1 and 1
1 + 1
```

Callouts

Callouts are used to signal the reader's attention to content, activity, and other important sections. The following callouts are used in this textbook:

Content

> **Outcomes**
> Learning outcomes for the chapter appear here.

> **Consider this**
> Points for you to consider and questions to explore appear here.

> **Case study**
> Case studies for applying conceptual knowledge and coding skills covered in the chapter appear here.

> **Dive deeper**
> Links to additional resources for diving deeper into the topic appear here.

Activities

> **Swirl lesson**
> Links to swirl lessons for practicing coding skills for the chapter appear here.

> **Recipe**
> Links to demonstration programming tasks on the qtalr site for the chapter appear here.

> **Lab**
> Links to lab exercises for applying conceptual knowledge and coding skills on the qtalr GitHub repository for the chapter appear here.

Other

> **Tip**
> Tips for using R and related tools appear here.

⚠ Warning
Warnings for using R and related tools appear here.

To the instructor

Depending on the experience level and expectations of your readers, you may want to consider adopting one of the following course designs for using this textbook.

Basic Introduction

- Cover chapters 1-5 in sequence to give your readers a foundational understanding of quantitative text analysis.
- Culminate the course with a research proposal assignment that requires them to identify an interesting linguistic problem, propose ways of solving it using the methods covered in class, and identify potential data sources.
- If your readers have little to no experience with R, you may want to consider using the RStudio Cloud platform to host the course. This will provide them with a pre-installed R environment and allow them to focus on learning the material rather than troubleshooting.

Intermediate Introduction

- Cover chapters 1, 5-10 in sequence to give your readers a deeper understanding of quantitative text analysis methods. Explore additional case studies or dataset examples throughout the course if you wish to supplement your lectures.
- Culminate the course with a research project assignment that allows your readers to apply what they've learned to linguistic content of their choice.
- You may consider using the RStudio Cloud platform to host the course, but ensure that your readers have access to R and RStudio on their own computers as well.

Advanced Introduction

- Cover all 12 chapters to give your readers a thorough understanding of quantitative text analysis concepts and techniques. Devote more time chapters 5-10 providing demonstrations of how to approach different problems and evaluating alternative approaches.
- Culminate the course with a collaborative research project that requires your readers to work in groups to conduct a comprehensive analysis of a given dataset.

- Ensure that your readers install R and RStudio on their own computers as they will need full control over their coding environment.

For all course designs, it is strongly recommend that you evaluate the readers' success in understanding the material by providing a combination of quizzes, lab assignments, programming exercises, and written reports. Additionally, encourage your readers to ask questions[27], collaborate with peers, and seek help from the ample resources available online when they encounter scope-limited programming problems.

For more information on how to use this textbook in your course, visit the Instructor Guide[29] on the compansion website.

Activities

At this point you should have a working R environment with the core packages including `qtalrkit` installed. You should also have verified that you have a working Git environment and that you have a GitHub account. If you have not completed these tasks, return to the guides listed above in "Getting started" of this Preface and complete them before proceeding.

The following activities are designed to help you become familiar with the tools and resources that you will be using throughout this textbook. These and subsequent activities are designed to be completed in the order that they are presented in this textbook.

Swirl lesson
What: Intro to Swirl[a]
How: In the R console load `swirl`, run `swirl()`, and follow prompts to select the lesson.
Why: To familiarize you with navigating, selecting, and completing swirl lessons.

[a] https://github.com/qtalr/swirl

Recipe
What: Literate programming I[a]
How: Read Recipe 0 and participate in collaborative discussion with peers.
Why: To introduce the concept of Literate Programming and how to create literate documents using R and Quarto.

[a] https://qtalr.github.io/qtalrkit/articles/recipe-0.html

[27] If you are using this textbook in a course, consider using a CMS (*e.g.* Canvas, Blackboard, etc.) or the web-based social annotation tool Hypothes.is[28] to facilitate reader questions and discussion.
[29] https://qtalr.github.io/qtalrkit/articles/instructor-guide.html

Lab
What: Literate programming I[a]
How: Clone, fork, and complete the steps in Lab 0.
Why: To put literate programming techniques covered in Recipe 0 into practice. Specifically, you will create and edit a Quarto document and render a report in PDF format.

[a]https://github.com/qtalr/lab-0

Summary

In the Preface, we lay the groundwork by introducing the textbook's underlying principles, learning goals, teaching methods, and target audience. The chapter also offers advice on how to navigate the book's layout, comprehend its subject matter, and make use of supplementary materials. Crucial insights from this section involve grasping the book's objectives and aims, which center around instructing readers on quantitative text analysis for linguistics using R while emphasizing reproducible research. This chapter assists readers in setting up a working R development environment ensuring they can effectively engage with the material. Moreover, the Preface provides guidance on how to get help with R and other related software tools and deciphering conventions in the text. With this foundation, you're now prepared to delve into the captivating realm of text analysis in the subsequent chapter, titled "Text Analysis in Context."

Questions

☐ Revise/ add questions.

Conceptual questions

- How is the textbook designed to be accessible for both novice and seasoned practitioners in the area of quantitative text analysis?
- What is the purpose of the textbook and what are the three areas it aims to scaffold?
- What are the main components of each chapter, and how are they structured to support learning outcomes?
- How does the structure of the textbook and associated resources work to support learning and proficiency in areas?
- What is the role of programmatic approaches in quantitative text analysis?
- What is the relationship between R and an IDE (e.g. RStudio, VS Code)?
- What is the relationship between R and a version control system (e.g. Git)?

Technical exercises

- Install the latest version of R by following the instructions for your operating system. https://cran.r-project.org/
- Install RStudio Desktop https://www.rstudio.com/products/rstudio/download/
- Verify a Git installation or install Git (Windows: https://git-scm.com/downloads). Git a version control system that allows you to track changes to files and collaborate with others through GitHub.
- Create a free GitHub account at https://github.com/join.
- Install the `tidyverse` package in R by running `install.packages("tidyverse")` in the R Console pane.
- Install the `swirl` package by running `install.packages("swirl")` in the R Console pane.
- Open RStudio and create a new project for this textbook. This will help you keep your code and files organized.

Part I

Orientation

- [ ] 🔧 Update the overview of Part I "Orientation" to reflect the new structure of the chapter.

In this section the aims are to: 1) provide an overview of quantitative research and their applications, by both highlighting visible applications and notable research in various fields, 2) consider how quantitative research contributes to language research, and 3) layout the main types of research and situate quantitative text analysis inside these.

1 Text analysis in context

Draft

Ready for review.

> Everything about science is changing because of the impact of information technology and the data deluge.
>
> — Jim Gray

Outcomes

- Understand the role and goals of data analysis both within and outside of academia.
- Describe the various approaches to quantitative language research.
- Identify the applications of text analysis in different contexts.

In this chapter I will aim to introduce the topic of text analysis and provide the context needed to understand how text analysis fits in a larger universe of science and the ever-ubiquitous methods of data science, with attention to how linguistics and language-related studies employ data analysis down to the particular area of text analysis.

Swirl lesson
What: Variables and vectors, Workspace[a]
How: In an R console load `swirl`, run `swirl()`, and follow prompts to select the lesson.
Why: To explore some key building blocks of the R programming language and to examine your local workspace in R and understand the relationship between your R workspace and the file system of your computing environment.

[a] https://github.com/qtalr/swirl

1.1 Making sense of a complex world

1.1.1 Heuristic Understanding

The world around us is full of actions and interactions so numerous that it is difficult to really comprehend. As each individual sees and experiences this world, we gain knowledge and build up heuristic understanding about how it works and how we can interact with it. This happens regardless of your educational background. As humans we are built for this. Our minds process countless sensory inputs. They underlie skills and abilities that we take for granted like being able to predict what will happen if you see someone about to knock a wine glass off a table and onto a concrete floor. You've never seen this object before and this is the first time you've been to this winery, but somehow and from somewhere you 'instinctively' make an effort to warn the would-be-glass-breaker before it is too late. You most likely have not stopped to consider where this predictive knowledge comes from, or if you have, you may have just chalked it up to 'common sense'. As common as it may be, it is an incredible display of the brain's capacity to monitor your environment, relate the events and observations that take place, and store that information all the time not making a big fuss to tell your conscious mind what it's up to.

So wait, this is a textbook on text analysis, right? So what does all this have to do with that? Well, there are two points to make that are relevant for framing our journey: (1) the world is constantly churning out data in real-time at a scale that is daunting and (2) for all the power of the brain that works so efficiently behind the scene making sense of the world, we are one individual living one life that has a limited view of the world at large. Let me expand on these two points a little more.

First let's be clear. There is no way for anyone to experience all things at all times. But even extremely reduced slices of reality are still vastly outside of our experiential capacity, at least in real-time. One can make the point that since the inception of the internet an individual's ability to experience larger slices of the world has increased. But could you imagine reading, watching, and listening to every file that is currently accessible on the web? Or has been? (See the Wayback Machine[1].) Scale this down even further; let's take Wikipedia, the world's largest encyclopedia. Can you imagine reading every wiki entry? As large as a resource such as Wikipedia is [2], it is still a small fragment of the written language that is produced on the web, just the web [4]. Consider that for a moment.

To my second framing point, which is actually two points in one. I underscored the efficiency of our brain's capacity to make sense of the world. That efficiency comes from some clever evolutionary twists that lead our brain to take in the world but it makes some shortcuts that compress the raw experience into heuristic understanding. What that means is that the brain is not a supercomputer. It does not store every experience in raw form, we do not have

[1] https://web.archive.org/

[2] As of 22 July 2021, there are 6,341,359 articles in the English Wikipedia[3] containing over 3.9 billion words occupying around 19 gigabytes of information.

[4] For reference, Common Crawl[5] has millions of gigabytes collected since 2008.

access to the records of our experience like we would imagine a computer would have access to the records logged in a database. Where our brains do excel is in making associations and predictions that help us (most of the time) navigate the complex world we inhabit. This point is key –our brains are doing some amazing work, but that work can give us the impression that we understand the world in more detail that we actually do. Let's do a little thought experiment. Close your eyes and think about the last time you saw your best friend. What were they wearing? Can you remember the colors? If your like me, or any other human, you probably will have a pretty confident feeling that you know the answers to these questions and there is a chance you a right. But it has been demonstrated in numerous experiments on human memory that our confidence does not correlate with accuracy (Talarico and Rubin 2003; Roediger and McDermott 2000). You've experienced an event, but there is no real reason that we should bet our lives on what we experienced. It's a little bit scary, for sure, but the magic is that it works 'good enough' for practical purposes.

So here's the deal: as humans we are (1) clearly unable to experience large swaths of experience by the simple fact that we are individuals living individual lives and (2) the experiences we do live are not recorded in memory with perfect precision and therefore we cannot 'trust' our intuitions, at least not in an absolute sense.

> **Consider this**
> How might your own experiences and biases influence your understanding of the world? What are some ways that you can mitigate these biases? Is ever possible to be completely objective? How might biases influence the way you approach text analysis?

1.1.2 Science to advance understanding

What does that mean for our human curiosity about the world around us and our ability to reliably make sense of it? In short it means that we need to approach understanding our world with the tools of science. Science starts with a question, identifies and collects data, careful selected slices of the complex world, submits this data to analysis through clearly defined and reproducible procedures, and reports the results for others to evaluate. This process is repeated, modifying, and manipulating the procedures, asking new questions and positing new explanations, all in an effort to make inroads to bring the complex into tangible view.

In essence what science does is attempt to subvert our inherent limitations in understanding by drawing on carefully and purposefully collected slices of observable experience and letting the analysis of these observations speak, even if it goes against our intuitions (those powerful but sometime spurious heuristics that our brains use to make sense of the world).

1.2 Data analysis

1.2.1 Emergence of data science

At this point I've sketched an outline strengths and limitations of humans' ability to make sense of the world and why science is used to address these limitations. This science I've described is the one you are familiar with and it has been an indispensable tool to make sense of the world. If you are like me, this description of science may be associated with visions of white coats, labs, and petri dishes. While science's foundation still stands strong in the 21st century, a series of intellectual and technological events mid-20th century set in motion changes that have changed aspects about how science is done, not why it is done. We could call this Science 2.0, but let's use the more popularized term **data science**. The recognized beginnings of data science are attributed to work in the "Statistics and Data Analysis Research" department at Bell Labs during the 1960s. Although primarily conceptual and theoretic at the time, a framework for quantitative data analysis took shape that would anticipate what would come: sizable datasets which would "[…] require advanced statistical and computational techniques […] and the software to implement them." (Chambers 2020) This framework emphasized both the inference-based research of traditional science, but also embraced exploratory research and recognized the need to address practical considerations that would arise when working with and deriving insight from an abundance of machine-readable data.

Fast-forward to the 21st century a world in which machine-readable data is truly in abundance. With increased computing power, the emergence of the world wide web, and wide adoption of mobile devices electronic communication skyrocketed around the globe. To put this in perspective, in 2019 it was estimated that every minute 511 thousand tweets were posted, 18.1 million text messages were sent, and 188 million emails were sent ("Data Never Sleeps 7.0 Infographic" 2019). The data flood has not been limited to language, there are more sensors and recording devices than ever before which capture evermore swaths of the world we live in (Desjardins 2019). Where increased computing power gave rise to the influx of data, it is also one of the primary methods for gathering, preparing, transforming, analyzing, and communicating insight derived from this data (Donoho 2017). The vision laid out in the 1960s at Bell Labs had come to fruition.

1.2.2 Computing skills, statistical knowledge, and domain knowledge

Data science is not predicated on data alone. Turning data into insight takes **computing skills** (i.e. programming), **statistical knowledge**, and **domain expertise**. This triad has been popularly represented as a Venn diagram such as in Figure 1.1.

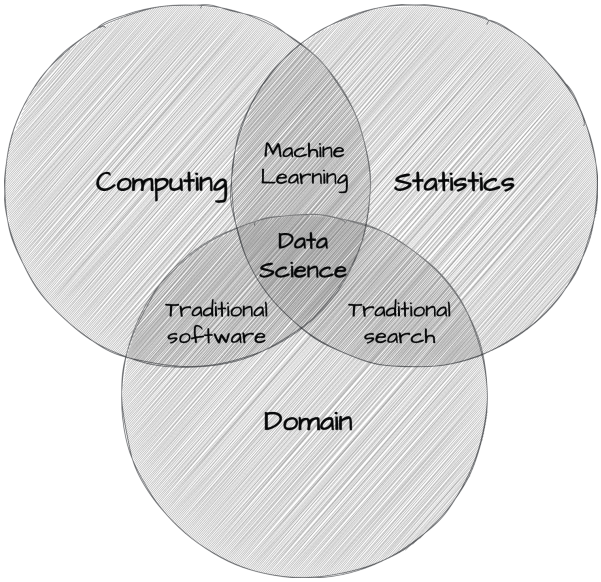


Figure 1.1: Data Science Venn Diagram adapted from Drew Conway[6].

The **computing skills** component of data science is the ability to write code to perform the data analysis process. This is the primary approach for working with data at scale. The **statistical knowledge** component of data science is the ability to apply statistical methods to data to derive insight. **Domain expertise** provides researchers insight at key junctures in the development of a research project and aid researchers in evaluating results.

This triad of skills in combination with reproducible research practices is the foundational toolbelt of data science (Hicks and Peng 2019). **Reproducible research** entails the use of computational tools to automate the process of data analysis. This automation is achieved by writing code that can be executed to replicate the data analysis. This code can then be shared through code sharing repositories, such as GitHub, where it can be viewed, downloaded, and executed by others. This adds transparency to the process and allows others to build on previous work. This is in contrast to traditional approaches where data analysis is performed (semi-)manually, results are reported in a static document such as a report or journal article, and the data analysis process is not shared. This approach is not reproducible because the data analysis process is not transparent and cannot be replicated. This is problematic because it is difficult to evaluate the results and build on previous work. Reproducible research practices are a key component of data science and are emphasized throughout this book.

1.2.3 Applications of data science

Equipped with the data science toolbelt, the interest in deriving insight from the available data is now almost ubiquitous. The science of data has now reached deep into all aspects of life

where making sense of the world is sought. Predicting whether a loan applicant will get a loan (Bao, Lianju, and Yue 2019), whether a lump is cancerous (Saxena and Gyanchandani 2020), what films to recommend based on your previous viewing history (Gomez-Uribe and Hunt 2015), what players a sports team should sign (Lewis 2004) all now incorporate a common set of data analysis tools.

The data science toolbelt also underlies well-known public-facing language applications. From the language-capable chat applications, plagiarism detection software, machine translation algorithms, and search engines, tangible results of quantitative approaches to language are becoming standard fixtures in our lives, as seen in Figure 1.2.

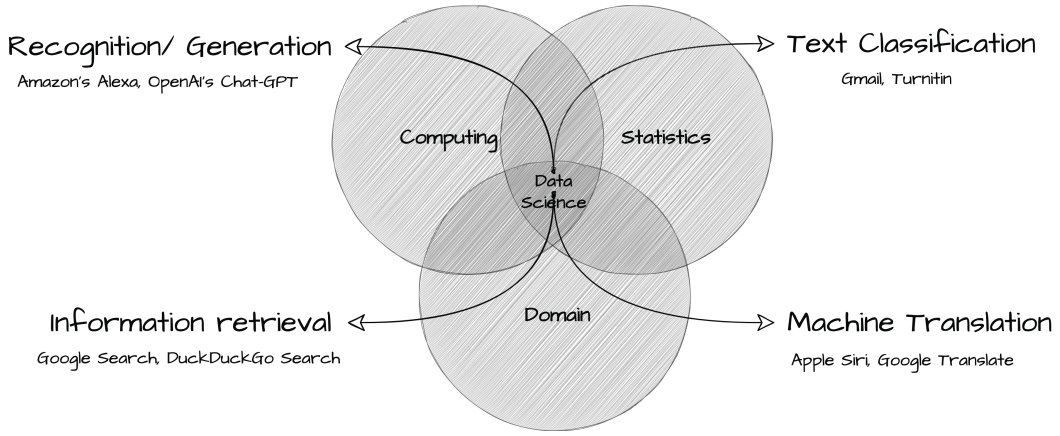


Figure 1.2: Well-known public-facing language applications

The spread of quantitative data analysis too has taken root in academia. Even in areas that on first blush don't appear readily approachable in a quantitative manner, such as fields in the social sciences and humanities, data science is making important and sometimes disciplinary changes to the way that academic research is conducted. This textbook focuses in on a domain that cuts across many of these fields; namely language. At this point let's turn to quantitative approaches to language analysis as we work closer to contextualizing text analysis.

1.3 Language analysis

Language is a defining characteristic of our species. Since antiquity, language has attracted interest across disciplines and schools of thought. In the early 20th century, the development of the rigorous approach to study of language as a field in its own right took root (Campbell 2001), yet a plurality of theoretical views and methodological approaches remained. Contemporary linguistics bares this complex history and is far from theoretically and methodologically unified.

Either based on the tenets of theoretical frameworks and/or the objects of study of particular fields, approaches to language research vary. On the one hand some language research commonly applies qualitative assessment of language structure and/ or use. **Qualitative approaches** describe and account for characteristics, or "qualities", that can be observed, but not measured (*e.g.* introspective methods, ethnographic methods, *etc.*)

On the other hand other language research programs employ quantitative research methods either out of necessity given the object of study (phonetics, psycholinguistics, *etc.*) or based on theoretical principles (Cognitive Linguistics, Connectionism, *etc.*). **Quantitative approaches** involve measurements of properties of language that can be observed and measured (*e.g.* frequency of use, reaction time, *etc.*).

These latter research areas and theoretical paradigms employ methods that share much of the common data analysis toolbox described in the previous section. In effect, this establishes a common methodological language between other language research fields but also with research outside of linguistics.

However, there is never a one-size-fits all approach to anything –much less data analysis. And even in quantitative language analysis there is a key methodological distinction that has downstream effects in terms of procedure but also in terms of interpretation. The key distinction that we need to make at this point, which will provide context for our introduction to quantitative text analysis, comes down to the approach to collecting language data and the nature of that data. This distinction is between **experimental data** and **observational data**.

Experimental approaches start with a intentionally designed hypothesis and lay out a research methodology with appropriate instruments and a plan to collect data that shows promise for shedding light on the validity of the hypothesis. Experimental approaches are conducted under controlled contexts, usually a lab environment, in which participants are recruited to perform a language related task with stimuli that have been carefully curated by researchers to elicit some aspect of language behavior of interest. Experimental approaches to language research are heavily influenced by procedures adapted from psychology. This link is logical as language is a central area of study in cognitive psychology. This approach looks much like the white-coat science that we made reference to earlier but, as in most quantitative research, has now taken advantage of the data analysis toolbelt to collect and organize much larger quantities of data and conduct statistically more robust analysis procedures and communicate findings more efficiently.

Observational approaches are a bit more of a mixed bag in terms of the rationale for the study; they may either start with a testable hypothesis or in other cases may start with a more open-ended research question to explore. But a more fundamental distinction between the two is drawn in the amount of control the researcher has on contexts and conditions in which the language behavior data to be collected is produced. Observational approaches seek out records of language behavior that is produced by language speakers for communicative purposes in natural(istic) contexts. This may take place in labs (language development, language disorders,

etc.), but more often than not, language is collected from sources where speakers are performing language as part of their daily lives –whether that be posting on social media, speaking on the telephone, making political speeches, writing class essays, reporting the latest news for a newspaper, or crafting the next novel destined to be a New York Times best-seller. What is more, data collected from the 'wild' varies more in structure relative to data collected in experimental approaches and requires a number of steps to prepare the data to sync up with the data analysis toolbelt.

I liken this distinction between experimental and observational data collection to the difference between farming and foraging. Experimental approaches are like farming; the groundwork for a research plan is designed, much as a field is prepared for seeding, then the researcher performs as series of tasks to produce data, just as a farmer waters and cares for the crops, the results of the process bear fruit, data in our case, and this data is harvested. Observational approaches are like foraging; the researcher scans the available environmental landscape for viable sources of data from all the naturally existing sources, these sources are assessed as to their usefulness and value to address the research question, the most viable is selected, and then the data is collected.

The data acquired from both of these approaches have their trade-offs, just as farming and foraging. Experimental approaches directly elicit language behavior in highly controlled conditions. This directness and level of control has the benefit of allowing researchers to precisely track how particular experimental conditions effect language behavior. As these conditions are an explicit part of the design and therefore the resulting language behavior can be more precisely attributed to the experimental manipulation. The primary shortcoming of experimental approaches is that there is a level of artificialness to this directness and control. Whether it is the language materials used in the task, the task itself, or the fact that the procedure takes place under supervision the language behavior elicited can diverge quite significantly from language behavior performed in natural communicative settings. Observational approaches show complementary strengths and shortcomings.

Whereas experimental approaches may diverge from natural language use, observational approaches strive to identify and collected language behavior data in natural, uncontrolled, and unmonitored contexts, as seen in Figure 1.3. In this way observational approaches do not have to question to what extent the language behavior data is or is not performed as a natural communicative act. On the flipside, the contexts in which natural language communication take place are complex relative to experimental contexts. Language collected from natural contexts are nested within the complex workings of a complex world and as such inevitably include a host of factors and conditions which can prove challenging to disentangle from the language phenomenon of interest but must be addressed in order to draw reliable associations and conclusions.

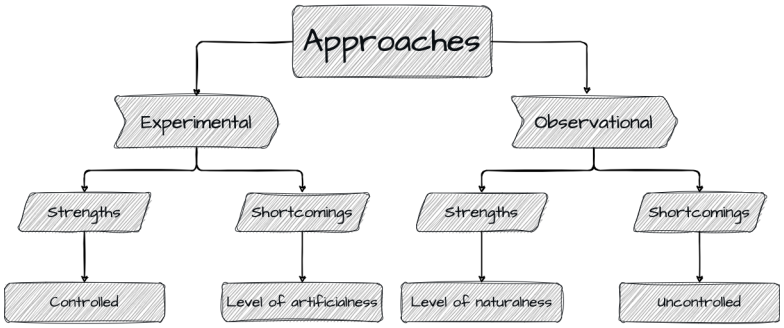


Figure 1.3: Trade-offs between experimental and observational data collection methods.

The upshot, then, is twofold: (1) data collection methods matter for research design and interpretation and (2) there is no single best approach to data collection, each have their strengths and shortcomings. In the ideal, a robust science of language will include insight from both experimental and observational approaches (Gilquin and Gries 2009). And evermore there is greater appreciation for the complementary nature of experimental and observational approaches and a growing body of research which highlights this recognition.

Case study
Manning (2003) discusses the use of probabilistic models in syntax to account for the variability in language usage and the presence of both hard and soft constraints in grammar. The paper touches on the statistical methods in text analysis, the importance of distinguishing between external and internal language, and the limitations of Generative Grammar. Overall, the paper suggests that usage-based and formal syntax can learn from each other to better understand language variation and change.

Given their particular trade-offs observational data is often used as an exploratory starting point to help build insight and form predictions that can then be submitted to experimental conditions. In this way, studies based on observational data serve as an exploratory tool to gather a better and more externally valid view of language use which can then serve to make prediction that can be explored with more precision in an experimental paradigm. However, this is not always the case; observational data is also often used in hypothesis-testing contexts as well. And furthermore, some in some language-related fields, a hypothesis-testing is not the approach for deriving knowledge and insight.

1.4 Text analysis

In a nutshell, **text analysis** is the process of leveraging the data science toolbelt to derive insight from textual data collected through observational methods. In the next subsections,

I will unpack this definition and discuss the primary components that make up text analysis including research appoaches and technical implementation, as well as practical applications.

1.4.1 Approaches

Text analysis is a multifacited research methodology. It can be used use facilitate the qualitative exploration of smaller, human-digestable textual information, but is more often employed quantitatively to bring to the surface patterns and relationships in large samples of textual data that would be otherwise difficult, if not impossible, to identify manually.

Text being text, there are a series of **data prepration** steps that must be taken to ready the data for analysis. In addition to collecting the data, the data must be organized, cleaned, and transformed into a format that is amenable to statistical analysis.

The statistical and evaluative approach employed in the analysis is dependent on the aim of the research. For research aimed at exploring and uncovering patterns and relationships in a data-driven manner, **Exploratory Data Analysis** (EDA) is employed. EDA combines descriptive statistics, visualizations, and statistical learning methods in an iterative and interactive way to provide the researcher the ability to identify patterns and relationships and to evaluate whether and why they are meaningful.

Predictive Data Analysis (PDA), applied in research for outcome prediction, is a supervised machine learning task. It uses feature sets to predict an outcome variable. Its primary evaluation metric is the prediction accuracy on new data. However, for many text analysis tasks, human interpretation is crucial to provide context and assess the significance of the results.

Research aimed at explaining relationships between variables and the population from which the sample was drawn will adopt an **Inferential Data Analysis** (IDA) approach. IDA is a theory-driven process that employs statistical models for hypothesis testing. The extent to which the results can be confidently generalized to the population is the primary evaluation metric.

As we see, text analysis can be used for a variety of purposes; from data-driven exploration and discovery to hypothesis testing and generalization.

1.4.2 Implementation

To ensure that the results of text analysis projects are replicable and transparent, programming strategies play an integral role at each stage of the implementation of a research project. While there are a number of programming languages that can be used for text analysis, R widely adopted in linguistics research. R is a free and open-source programming language that is specifically designed for statistical computing and graphics. It has a large and active community of users and developers, and a robust ecosystem of packages which make it a

powerful and flexible language that is well-suited for core text analysis tasks: data collection, organization, transformation, analysis, and visualization. When combined with Quarto for literate programming and GitHub for version control and collaboration, R provides a robust and reproducible workflow for text analysis.

1.4.3 Applications

So what are some applications of text analysis? Most public facing applications stem from Computational Linguistic research, often known as **Natural Language Processing** (NLP) by practitioners. Whether it be using search engines, online translators, submitting your paper to plagiarism detection software, *etc.* many of the text analysis methods we will cover are at play.

> **Consider this**
> What are some other public facing applications of text analysis that you are aware of?

In academia the use of quantitative text analysis is even more widespread, despite the lack of public fanfare. In linguistics, text analysis is applied to a wide range of topics and research questions in both theoretical and applied subfields, as seen in Example 1.1 and Example 1.2.

Example 1.1. Theoretical linguistics

- Hay (2002) use a corpus study to investigate the role of frequency and phonotatics in affix ordering in English.
- Riehemann (2001) explores the extent to which idiomatic expressions (*e.g.* 'raise hell') are lexical or syntactic units.
- Bresnan (2007) investigate the claim that possessed deverbal nouns in English (*e.g.* 'the city's destruction') are subject to a syntactic constraint that requires the possessor to be affected by the action denoted by the deverbal noun.

Example 1.2. Applied linguistics

- Wulff, Stefanowitsch, and Gries (2007) explore differences between British and American English at the lexico-syntactic level in the *into*-causative construction (*e.g.* 'He tricked me into employing him.').
- Eisenstein et al. (2012) track the geographic spread of neologisms (*e.g.* 'bruh', 'af', '-___-') from city to city in the United States using Twitter data collected between 6/2009 and 5/2011.
- Bychkovska and Lee (2017) investigates possible differences between L1-English and L1-Chinese undergraduate students' use of lexical bundles, multiword sequences which are extended collocations (*e.g.* 'as the result of'), in argumentative essays.

- Jaeger and Snider (2007) use a corpus study to investigate the phenomenon of syntactic persistence, the increased tendency for speakers to use a particular syntactic form over an alternate when the syntactic form has been recently processed.
- Voigt et al. (2017) explore potential racial disparities in officer respect in police body camera footage.
- Olohan (2008) investigate the extent to which translated texts differ from native texts do to 'explicitation'.

So too, text analysis is used in a variety of other fields where insight from language is sought, as seen in Example 1.3.

Example 1.3. Language-related fields

- Kloumann et al. (2012) explore the extent to which languages are positively, neutrally, or negatively biased.
- Mosteller and Wallace (1963) provide a method for solving the authorship debate surrounding The Federalist papers.
- Conway et al. (2012) investigate whether the established drop in language complexity of rhetoric in election seasons is associated with election outcomes.

> **Consider this**
> Language is a key component of human communication and interaction. What are some other areas of research in and outside linguistics that you think could be explored using text analysis methods?

These studies in Examples 1.1, 1.2, and 1.3 are just a few illustrations of the contributions of text analysis as the primary method to gain a deeper understanding of language structure, function, variation, and acquisition. As a method, however, text analysis can also be used to support other research methods. For example, text analysis can be used collect data, generate authentic materials, provide linguistic annotation, to generate hypotheses, for either qualitative and/ or quantitative approaches. Together these efforts contribute to a more robust language science by incorporating externally valid data and providing methodological triangulation (Francom 2022).

In sum, the applications highlighted in this section underscore the versatility of text analysis as a research method. Whether it be in the public sphere or in academia, text analysis methods furnish a set of powerful tools for gaining insight from language data.

Summary

In this chapter I started with some general observations about the difficulty of making sense of a complex world. The standard approach to overcoming inherent human limitations in sense

making is science. In the 21st century the toolbelt for doing scientific research and exploration has grown in terms of the amount of data available, the statistical methods for analyzing the data, and the computational power to manage, store, and share the data, methods, and results from quantitative research. The methods and tools for deriving insight from data have made significant inroads in and outside academia, and increasingly figure in the quantitative investigation of language. Text analysis is a particular branch of this enterprise based on observational data from real-world language and is used in a wide variety of fields.

In the end I hope that you enjoy this exploration into text analysis. Although the learning curve at times may seem steep –the experience you will gain will not only improve your data literacy, research skills, and programmings skills but also enhance your appreciation for the richness of human language and its important role in our everyday lives.

Actitivies

The following activities build on your introduction to R and Quarto in the previous chapter. In these activities you will uncover more features offered by Quarto which will enhance your ability to produce comprehensive reproducible research documents. You will apply the capabilities of Quarto in a practical context conveying the objectives and key discoveries from a primary research article.

Recipe
What: Literate programming II[a]
How: Read Recipe 1 and participate in the Hypothes.is online social annotation.
Why: To explore additional functionality in Quarto: numbered sections, table of contents, in-line citations and a document-final references list, and cross-referenced tables and figures.

[a]https://qtalr.github.io/qtalrkit/articles/recipe-.html

Lab
What: Literate programming II[a]
How: Clone, fork, and complete the steps in Lab 1.
Why: To put into practice Quarto functionality to communicate the aim(s) and main finding(s) from a primary research article and to interpret a related plot.

[a]https://github.com/qtalr/lab-1

Questions

Conceptual questions

- How has scientific research and exploration changed in the 21st century?
- What are the three basic skill sets that make up the data science toolbelt?
- What are the benefits of reproducible research in data science?
- Explain the trade-offs between experimental and observational data collection methods.
- What is text analysis and how is it used in various fields?
- Identify research in an area of interest in linguistics that has taken a quantitative approach to text analysis.
- In your own words, define literate programming?
- What are the benefits of literate programming?
- What are the benefits of using R and Quarto for literate programming?

Technical exercises

- Create a literate programming document in Quarto. Edit the yaml header to reflect details of the work and add your work with the data types in R to code chunks. Add, commit, and push the project to GitHub.
- In the Quarto document, explore using R to create vectors and explore their properties.
- Explore the following resources and with the goal to identify a quantitative text analysis project. Rpubs[a], GitHub[b], DataCamp[c], Kaggle[d], R-bloggers[e].
- 🔧 … more to come …

[a] https://rpubs.com/
[b] https://github.com
[c] https://datacamp.com
[d] https://kaggle.com
[e] https://r-bloggers.com

Part II

Foundations

Before working on the specifics of a data project, it is important to establish a fundamental understanding of the characteristics of each of the levels in the "Data, Information, Knowledge, and Insight Hierarchy (DIKI)" (see Figure 1) and the roles each of these levels have in deriving insight from data. In Chapter 2 we will explore the Data and Information levels drawing a distinction between two main types of data (populations and samples) and then cover how data is structured and transformed to generate information (datasets) that is fit for statistical analysis. In Chapter 3 I will outline the importance and distinct types of statistical procedures (descriptive and analytic) that are commonly used in text analysis. Chapter 4 aims to tie these concepts together and cover the required steps for preparing a research blueprint to conduct an original text analysis project.

2 Understanding data

Draft

Ready for review.

> The goal is to turn data into information, and information into insight.
>
> — Carly Fiorina

Outcomes

- Describe the difference between data and information.
- Understand how the tidy approach to data organization can enhance the quality and usability of data.
- Articulate the importance of documentation in promoting reproducible research.

In this chapter, the groundwork is laid for deriving insights from text analysis by focusing on content and structure of data and information. The concepts of populations and samples are introduced, highlighting their similarities and key differences. Connecting these topics to text analysis, language samples, or corpora, are explored, discussing their types, sources, formats, and ethical considerations. Subsequently, key concepts in creating information from data, such as organization and transformation, are examined. Finally, the importance of documentation in quantitative research is emphasized through addressing data origin and data dictionaries.

Swirl lesson
What: Objects, Packages and functions[a]
How: In the R Console pane load `swirl`, run `swirl()`, and follow prompts to select the lesson.
Why: To introduce you to the main types of objects in R and to understand the role and use of functions and packages in R programming.

[a] https://github.com/qtalr/swirl

2.1 Data

Data is data, right? The term 'data' is so common in popular vernacular it is easy to assume we know what we mean when we say 'data'. But as in most things, where there are common assumptions there are important details that require more careful consideration. Let's turn to the first key distinction that we need to make to start to break down the term 'data': the difference between populations and samples.

2.1.1 Populations

The first thing that comes to many people's mind when the term population is used is human populations (derived from Latin 'populus'). Say for example we pose the question –What's the population of Milwuakee? When we speak of a population in these terms we are talking about the total sum of individuals living within the geographical boundaries of Milwaukee. In concrete terms, a **population** an idealized set of objects or events in reality which share a common characteristic or belong to a specific category. The term to highlight here is idealized. Although we can look up the US Census report for Milwaukee and retrieve a figure for the population, this cannot truly be the population. Why is that? Well, whatever method that was used to derive this numerical figure was surely incomplete. If not incomplete, by the time someone recorded the figure some number of residents of Milwaukee moved out, moved in, were born, or passed away. In either case, this example serves to point out that populations are not fixed and are subject to change over time.

Likewise when we talk about populations in terms of language we dealing with an idealized aspect of linguistic reality. Let's take the words of the English language as an analog to our previous example population. In this case the words are the people and English is the grouping characteristic. Just as people, words move out, move in, are born, and pass away. Any compendium of the words of English at any moment is almost instananeously incomplete. This is true for all populations, save those relatively rare cases in which the grouping characteristics select a narrow slice of reality which is objectively measurable and whose membership is fixed (the complete works of Shakespeare, for example).

In sum, (most) populations are amorphous moving targets. We subjectively hold them to exist, but in practical terms we often cannot nail down the specifics of populations. So how do researchers go about studying populations if they are theoretically impossible to access directly? The strategy employed is called sampling.

2.1.2 Samples

A **sample** is the product of a subjective process of selecting a finite set of observations from an idealized population with the goal of capturing the relevant characteristics of the target population. The **degree of representativeness** of a sample is the extent to which the sample

reflects the characteristics of the population. The degree of representativeness is crucial for research as it directly impacts of any findings based on the sample.

To maximize the representativeness of a sample, researchers employ a variety of strategies. One of the first and sometimes the easiest strategy is to increase the **sample size**. A larger sample will always be more representative than a smaller sample. Sample size, however, is often not enough. It is not hard to imagine a large sample which by chance captures only a subset of the features of the population. Another step to enhance sample representativeness is to apply **random sampling**. Together a large random sample has an even better chance of reflecting the main characteristics of the population better than a large or random sample. But, random as random is, we still run the risk of acquiring a skewed sample (*i.e.* a sample with low representativeness).

To help mitigate these issues, there are two more strategies that can be applied to improve sample representativeness. Note, however, that while size and random samples can be applied to any sample with few assumptions about internal characteristics of the population, these next two strategies require decisions depend on the presumed internal characteristics of the population.

The first of these more informed sampling strategies is called **stratified sampling**. Stratified samples make (educated) assumptions about sub-components within the population of interest. With these sub-populations in mind, large random samples are acquired for each sub-population, or strata. At a minimum, stratified samples can be no less representative than random sampling alone, but the chances that the sample is better increases. Can there be problems in the approach? Yes, and on two fronts. First knowledge of the internal components of a population are often based on a limited or incomplete knowledge of the population (remember populations are idealized). In other words, strata are selected subjectively by researchers using various heuristics some of which are based on some sense of 'common knowledge'.

The second front on which stratified sampling can err concerns the relative sizes of the sub-components relative to the whole population, which is known as **balance**. Even if the relevant sub-components are identified, their relative size adds another challenge which researchers must address in order to maximize the representativeness of a sample.

Together, large randomly selected and balanced stratified samples set the benchmark for sampling. However, hitting this ideal is not always feasible. There are situations where sizeable samples are not accessible. Alternatively, there may be instances where the population or its strata are not well understood. In such scenarios, researchers have to work with the most suitable sample they can obtain given the limitations of their research project.

2.1.3 Corpora

A key feature of a sample is that it is purposely selected to model a target population. In text analysis, a purposely sampled collection of texts, of the type defined here, is known as a

corpus (*pl.* corpora). A set of texts or documents which have not been selected purposely lack a **sampling frame**, and therefore is not a corpus. The sampling frame, hence the populations modeled, in any given corpus will vary. It is key to vet corpora to ensure that the resource's sampling frame and the research project's target populations align as closely as possible to safeguard the integrity of research findings later in the research process.

Consider this

The 'Standard Sample of Present-Day American English' (known commonly as the Brown Corpus) is widely recognized as one of the first large, machine-readable corpora. Compiled by Kucera and Francis (1967), the corpus is comprised of 1,014,312 words from edited English prose published in the United States in 1961.Given the sampling frame for this corpus visualized in Figure 2.1, can you determine what language population this corpus aims to represent? What types of research might this corpus support or not support?

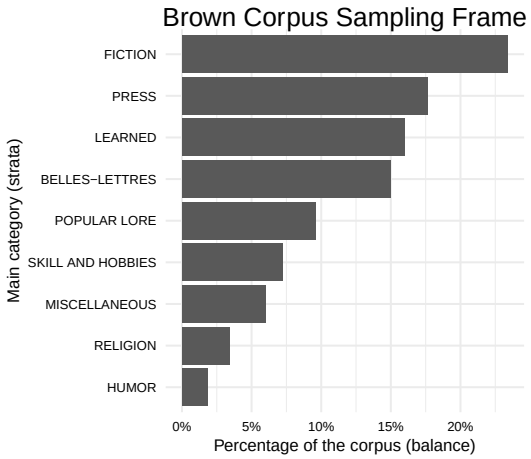


Figure 2.1: Overview of the sampling frame of the Brown Corpus.

Types

Let's take a look at some key characteristics, attributes, and features that distinguish corpora.

Reference

The least common and most ambitious corpus resources are those which aim to model the characteristics of a language population. These are known as **reference corpora**. These are projects designed with wide sampling frames, and require significant investments of time in corpus design and implementation (and continued development) that are usually undertaken by research teams (Ädel 2020).

The American National Corpus (ANC)[1] or the British National Corpus (BNC)[2] are corpora which aim to model the general characteristics of a variety of the English language, the former of American English and the later British English. Reference corpora exist for other languages as well: Spanish Reference Corpus of Present-Day Spanish (CREA)[3], German The German Reference Corpus (DeReKo)[4], Turkish Turkish National Corpus (TNC)[5], and many others.

Consider this

Of note is the fact that, at present, most of the world's languages lack reference corpus resources, or any corpus resources whatsoever. "Low-resourced" languages are often less studied, resource scarce, less available in born-digital formats, etc. (Magueresse, Carles, and Heetderks 2020).

Visit the Clarin overview[a] on reference corpora and then visit LRE Map[b]. Can you find a reference corpus for a language you speak or are interested in studying? If not, consider what can be done to address this gap in the research community.

[a] https://www.clarin.eu/resource-families/reference-corpora
[b] https://lremap.elra.info/

Specialized

Specialized corpora aim to represent more specific populations. The population may be defined either by modality, genre, time, location, or speaker-oriented characteristics, or some combination thereof. What specialized corpora lack in breadth of coverage, they make up for in depth of coverage by providing a more targeted representation of specific language populations.

The Santa Barbara Corpus of Spoken American English (SBCSAE)[6], as you can imagine from the name of the resource, aims to model spoken American English. No claim to written English is included. There are even more specific types of corpora which attempt to model other types of sub-populations such as academic writing[7], computer-mediated communication (CMC)[8], language use in specific regions of the world[9], a country[10], a region of a country[11], *etc.*

[1] https://www.anc.org/
[2] http://www.natcorp.ox.ac.uk/
[3] http://corpus.rae.es/creanet.html
[4] https://www.ids-mannheim.de/digspra/kl/projekte/korpora/
[5] https://www.tnc.org.tr/
[6] https://www.linguistics.ucsb.edu/research/santa-barbara-corpus
[7] https://www.coventry.ac.uk/research/research-directories/current-projects/2015/british-academic-written-english-corpus-bawe/
[8] https://www.clarin.eu/resource-families/cmc-corpora
[9] http://ice-corpora.net/ice/index.html
[10] https://www.wgtn.ac.nz/lals/resources/corpora-default/corpora-wsc
[11] https://cesa.arizona.edu

Consider this

Grieve, Nini, and Guo (2018) compiled a 8.9 billion-word corpus of geotagged posts from Twitter between 2013-2014 in the United States. The authors provide a search interface[a] to explore relationship between lexical usage and geographic location. Explore this corpus searching for terms related to slang ("hella", "wicked"), geographical ("mountain", "river"), meteorological ("snow", "rain"), and/ or any other term types. What types of patterns do you find? What are the benefits and/ or limitations of this type of data and/ or interface?

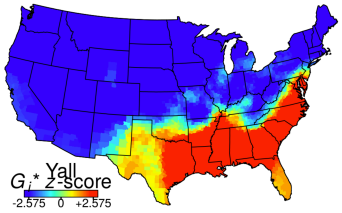


Figure 2.2: Example distribution of the term 'Ya'll' the Word Mapper project.

[a]https://isogloss.shinyapps.io/isogloss/

Another set of specialized corpora are resources which aim to compile texts from different languages or different language varieties for direct or indirect comparison. Corpora that are directly comparable, that is they include source and translated texts, are called **parallel corpora**. Parallel corpora include different languages or language varieties that are indexed and aligned at some linguistic level (*i.e.* word, phrase, sentence, paragraph, or document), see OPUS[12]. Corpora that are compiled with different languages or language varieties but are not directly aligned are called **comparable corpora**. The comparable language or language varieties are sampled with the same or similar sampling frame, for example Brown[13] and LOB[14] corpora.

The aim of the quantitative text researcher is to select the corpus, or corpora, which best align with the purpose of the research. For example, a general corpus such as the American National Corpus may be better suited to address a question dealing with the way American English works, but this general resource may lack detail in certain areas, such as medical language[15], that may be vital for a research project aimed at understanding changes in medical terminology. Furthermore, a researcher studying spoken language might collect a corpus of transcribed conversations from a particular community or region, such as the SBCSAE. While this would not include every possible spoken utterance produced by members of that group, it could be considered a representative sample of the population of speech in that context.

Sources

Published

[12]https://opus.nlpl.eu/
[13]https://ota.bodleian.ox.ac.uk/repository/xmlui/handle/20.500.12024/0402
[14]https://ota.bodleian.ox.ac.uk/repository/xmlui/handle/20.500.12024/0167
[15]https://mtsamples.com/index.asp

The most common source of data used in contemporary quantitative research is the internet. On the web an investigator can access corpora published for research purposes. Many organizations exist around the globe that provide access to published corpora in browsable catalogs, or **repositories**. There are repositories dedicated to language research, in general, such as the Language Data Consortium[16] or that specialize in specific domains, such as the spoken language repository TalkBank[17]. It is always advisable to start looking for the available language data in a repository. The advantage of beginning your data search in repositories is that a repository, especially those geared towards the linguistic community, will make identifying language corpora faster than through a general web search. Furthermore, repositories often require certain standards for corpus format and documentation for publication.

Repositories are by no means the only source of published corpora on the web. Researchers from around the world provide access to corpora and datasets on their own sites or through data sharing platforms. Corpora of various sizes and scopes will often be accessible on a dedicated homepage or appear on the homepage of a sponsoring institution. These resources may be available for download or via search inferaces. Finding these resources is often a matter of doing a web search with the word 'corpus' and a list of desired attributes, including language, modality, register, *etc.*

As part of a general movement towards reproducibility, more corpora are available on **data sharing platforms** such as GitHub[18], Zenodo[19], Re3data[20], OSF[21], *etc.* These platforms enable researchers to securely store, manage, and share data with others. Support is provided for various types of data, including documents and code, and as such they are a good place to look as they often include reproducible research projects as well.

Develop

Language corpora prepared by researchers and research groups listed on repositories or hosted by the researchers themselves is often the first place to look for data. The web, however, contains a wealth of language and language-related data that can be accessed by researcher to compile their own corpus. There are two primary ways to attain language data from the web. The first is through an **Application Programming Interface** (API). APIs are, as the title suggests, programming interfaces which allow access, under certain conditions, to information that a website or database accessible via the web contains.

The second, more involved, way to acquire data from the web is is through the process of web scraping. **Web scraping** is the process of harvesting data from the public-facing web. Language texts may be found on sites as uploaded files, such as pdf or doc (Word) documents,

[16] https://www.ldc.upenn.edu/
[17] http://talkbank.org/
[18] https://github.com/
[19] https://zenodo.org/
[20] http://www.re3data.org/
[21] https://osf.io/

or found displayed as the primary text of a site. Given the wide variety of documents uploaded and language behavior recorded daily on news sites, blogs and the like, compiling a corpus has never been easier. Having said that, how the data is structured and how much data needs to be retrieved can pose practical obstacles to collecting data from the web, particularly if the approach is to acquire the data by manually instead of automating the task.

Dive deeper
The process of corpus development is a topic in and of itself. For a more in-depth discussion of the process, see Ädel (2020).

Consider this
Explore some of the resources listed on the qtalrkit compansion site[a] and consider their sampling frames. Can you think of a research question or questions that this resource may be well-suited to support research into? What types of questions would be less-than-adequate for a given resource?

[a]https://qtalr.github.io/qtalrkit/articles/guide-5.html

Ethical considerations

Just because data is available on the web does not mean it is free to use. Repositories, APIs, and individual data resources often have licensing agreements and terms of use, ranging from public domain to proprietary licenses. Public domain licenses, such as those found in Project Gutenberg, allow anyone to use the data for any purpose. Creative Commons licenses[22], like those used by the American National Corpus, Wikipedia, and TalkBank, span from public domain to more restrictive uses, including requirements for attribution or prohibiting commercial use. Even more restrictive licenses, such as those for the Corpus of Contemporary American English and the British National Corpus, may require a fee to access and use the data, even for research purposes.

Respecting intellectual property rights is crucial when working with corpus data. Violating these rights can lead to legal and ethical issues, including lawsuits, fines, and damage to one's professional reputation. To avoid these problems, researchers must ensure they have the necessary permissions to use copyrighted works in their corpora. Obtaining permissions involves contacting the author or publisher and requesting consent to use their work for research purposes. Documenting all obtained permissions and providing attribution and/ or citation is essential respecting the intellectual property rights of others.

[22]https://creativecommons.org/about/cclicenses/

Formats

Whether you are using a published corpus or developing your own, it is important to understand how the data you want to work with is formatted. When referring to the format of a corpus, this includes the folder and file structure, the file types, the internal structure of the files themselves, and how file content is encoded electronically.

Folder and file structure

Some corpus resources are contained in a single file, such as a spreadsheet or a text file, but more often than not a corpus will be comprised of multiple files and folders. The folder and file structure will reflect the organization of the corpus and may include sub-folders for different types or groupings of data. In addition to the corpus data itself, metadata and documentation will often be included in the corpus folder structure. The corpus data may be grouped by language, modality, register, or other attributes such as types of linguistic annotation.

To illustrate, in Example 2.1 we have the file and folder structure of a toy corpus.

Example 2.1. Toy corpus structure

```
corpus/
├── documentation/
│   ├── README.md
│   ├── LICENSE
├── metadata/
│   ├── speakers.csv
├── data/
│   ├── spoken/
│   │   ├── inter-09-a.xml
│   │   ├── inter-09-b.xml
│   │   ├── convo-09-a.xml
│   │   ├── ...
│   ├── written/
│   │   ├── essay-09-a.xml
│   │   ├── essay-09-b.xml
│   │   ├── respo-09-a.xml
│   │   ├── ...
```

In this example, we have a corpus folder with three sub-folders: *documentation/*, *metadata/*, and *data/*. The *data/* folder contains two sub-folders: *spoken/* and *written/*. Each folder contains the relevant data files.

Where a single file is easy to download from the web, a corpus with a more complex folder structure can be more difficult to access. For that reason, many corpus resources are packaged into and made into a single compressed file. **File compression** has two benefits: it preserves the folder structure in a format which is contained in a single file and it also reduces the overall storage size. Common file compression formats are *.zip* and *.tar.gz*. So a compressed corpus file for the example above may be named something like *corpus.zip* or *corpus.tar.gz*. To access the original data within a compressed file, one must use a decompression tool or software to extract the contents after downloading it.

File types

In our toy corpus example, you may have noticed that each of the filenames appear with either *.md*, *.csv*, *.xml*, or nothing appended. These are examples of **file extensions**. File extensions a short sequence of characters, usually preceded by a period (.) which are used to indicate the type or format of file. File extensions help both users and software programs to identify the content and purpose of a file.

> **⚠ Warning**
> If you are working on your own desktop computer, you may not see the file extensions. This is because the file explorer is configured to hide them by default. To see the file extensions, you will need to change the settings in your file explorer. Use a search engine to find instructions for your operating system.

In addition to those listed above, other file extensions often encountered when working with data for text analysis include *.txt*, *.pdf*, *.docx*, *.xlsx*, *.json*, and *.html*. Common file extensions will often be associated with specific software programs on your computer, especially those which are directly associated with proprietary software such as *.docx* for Microsoft Word or *.xlsx* for Microsoft Excel. However, many file extensions are not directly associated with any specific software program and can be opened and edited with any text editor.

It is important to note that file extensions are helpful conventions, but they are not a guarantee of the file type or structure of the file content. Furthermore, corpus developers may create their own file extensions to signal the unique structure of their data. For example, the *.utt* file extension used in the Switchboard Dialogue Act Corpus (SWDA) or the *.cha* extension used for TalkBank resource transcripts signal project-specific structuring. In either case, it is recommended to open the file in a text editor to inspect the structure of the file content to confirm the file structure before processing the data contained therein.

File content

The internal structure of the content of corpus data files is an important aspect of any corpus both in terms of what data is included and how to approach accessing and processing the data. A corpus may include various types of linguistic (*e.g.* part of speech, syntactic structure, named

entities, *etc.*) or non-linguistic (*e.g.* source, dates, speaker information, *etc.*) attributes. These attributes are known as **metadata**, or data about data. As a general rule, files which include more metadata tend to be more internally structured. Internal file structure refers to the degree to which the content is easy to query and analyze by a computer. Let's review characteristics of the three main types of file structure types and associate common file extensions that files in each have.

Unstructured data is data which does not have a machine-readable internal structure. This is the case for plain text files (*.txt*), which are simply a sequence of characters. For example, in Example 2.2 we see a snippet of a plain text file from the the Manually Annotated Sub-Corpus of American English (MASC) (Ide et al. 2008):

Example 2.2. MASC plain text

```
>Hotel California

Fact: Sound is a vibration. Sound travels as a mechanical wave through a medium,
 ↪  and in space, there is no
medium. So when my shuttle malfunctioned and the airlocks didn't keep the air
 ↪  in, I heard nothing. After the
first whoosh of the air being sucked away, there was lightning, but no thunder.
 ↪  Eyes bulging in
panic, but no screams. Quiet and peaceful, right? Such a relief to never again
 ↪  hear my crewmate Jesse natter
about his girl back on Earth and that all-expenses-paid vacation-for-two she won
 ↪  last time he was on leave. I
swore, if I ever had to see a photo of him in a skimpy bathing suit again,
 ↪  giving the camera a cheesy thumbs-up
from a lounge chair on one of those white sandy beaches, I'd kiss a monkey.
 ↪  Metaphorically, of course.
```

Other examples of files which often contain unstructured data include *.pdf* and *.docx* files. While these file types may contain data which appears structured to the human eye, the structure is not designed to be machine-readable. As such the data would typically be read into R as a vector of **character strings**. It is possible to perform only the most rudimentary queries on this type of data, such as string matches. For anything more informative, it is necessary to further process this data.

On the other end of the spectrum, **structured data** is data which conforms to a tabular format in which elements in tables and relationships between tables are defined. This makes querying and analyzing easy and efficient. Relational databases (*e.g.* MySQL, PostgreSQL, etc.) are designed to store and query structured data. The data frame object in R is also a structured data format. In each case, the data is stored in a tabular format in which each row

represents a single observation and each column represents a single attribute whose values are of the same type.

In Example 2.3 we see an example of an R data frame object which overlaps with the data in the plain text file above in Example 2.2:

Example 2.3. MASC data frame

```
   title             date modality domain          ref_num word  lemma pos
   <chr>            <dbl> <fct>    <chr>              <dbl> <chr> <chr> <chr>
 1 Hotel California  2008 Writing  General Fiction        0 >     >     NN
 2 Hotel California  2008 Writing  General Fiction        1 Hotel hotel NNP
 3 Hotel California  2008 Writing  General Fiction        2 Cali… cali… NNP
 4 Hotel California  2008 Writing  General Fiction        3 Fact  fact  NNP
 5 Hotel California  2008 Writing  General Fiction        4 :     :     :
 6 Hotel California  2008 Writing  General Fiction        5 Sound sound NNP
 7 Hotel California  2008 Writing  General Fiction        6 is    be    VBZ
 8 Hotel California  2008 Writing  General Fiction        7 a     a     DT
 9 Hotel California  2008 Writing  General Fiction        8 vibr… vibr… NN
10 Hotel California  2008 Writing  General Fiction        9 .     .     .
11 Hotel California  2008 Writing  General Fiction       10 Sound sound NNP
```

Here we see that the data is stored in a tabular format with each row representing a single observation (`word`) and each column representing a single attribute. Internally, R applies a schema to ensure the values in each column are of the same type (*e.g.* `<chr>`, `<dbl>`, `<fct>`, *etc.*). This structured format is designed to be easy to query and analyze and as such is the primary format for data analysis in R.

Tip

It is conventional to work with column names for datasets in R using the same conventions that are used for naming objects. It is a matter of taste which convention is used, but I have adopted snake case[a] as my personal preference (*e.g* `ref_num`). There are also alternatives[b]. Regardless of the convention you choose, it is good practice to be consistent. It is also of note that the column names should be balanced for meaningfulness and brevity. This brevity is of practical concern but can be somewhat opaque. For questions into the meaning of the column and is values consult the resource's dataset documentation, consult Section 2.3.

[a] https://bookdown.org/content/d1e53ac9-28ce-472f-bc2c-f499f18264a3/names.html#snake_case
[b] https://bookdown.org/content/d1e53ac9-28ce-472f-bc2c-f499f18264a3/names.html

Semi-structured data falls between unstructured and structured data. This covers a wide range of file structuring approaches. For example, a otherwise plain text file with part-of-speech tags appended to each word is minimally structured (Example 2.4).

Example 2.4. MASC plain text with part-of-speech tags

```
>/NN Hotel/NNP California/NNP Fact/NNP :/: Sound/NNP is/VBZ a/DT vibration/NN
↪  ./. Sound/NNP travels/VBZ as/IN a/DT mechanical/JJ wave/NN through/IN a/DT
↪  medium/NN ,/, and/CC in/IN space/NN ,/, there/EX is/VBZ no/DT medium/NN ./.
↪  So/RB when/WRB my/PRP$ shuttle/NN malfunctioned/JJ and/CC the/DT
↪  airlocks/NNS did/VBD n't/RB keep/VB the/DT air/NN in/IN ,/, I/PRP heard/VBD
↪  nothing/NN ./. After/IN the/DT
```

Towards the more structured end of semi-structured data, many file formats including *.xml* and *.json* contain highly structured, hierarchical data. For example, in Example 2.5 shows a snippet from a *.xml* file from the MASC corpus.

Example 2.5. MASC XML

```
<a xml:id="penn-N65571" label="tok" ref="penn-n0" as="anc">
↪
  <fs>
    <f name="base" value="&gt;"/>
    <f name="msd" value="NN"/>
    <f name="string" value="&gt;"/>
  </fs>
</a>
<node xml:id="penn-n1">
  <link targets="seg-r1"/>
</node>
<a xml:id="penn-N65599" label="tok" ref="penn-n1" as="anc">
  <fs>
    <f name="base" value="hotel"/>
    <f name="msd" value="NNP"/>
    <f name="string" value="Hotel"/>
  </fs>
</a>
```

The format of semi-structured data is often influenced by characteristics of the data or reflect an author's individual preferences. It is sometimes the case that data will be semi-structured in a less-standard format. For example, the SWDA corpus includes a *.utt* file extension for files which contain utterances annotated with dialogue act tags.

Example 2.6. SWDA *.utt* file

```
o          A.1 utt1: Okay.  /
qw          A.1 utt2: {D So, }

qy^d          B.2 utt1: [ [ I guess, +

+         A.3 utt1: What kind of experience [ do you, + do you ] have, then with
 ↪  child care? /

+          B.4 utt1: I think, ] + {F uh, } I wonder ] if that worked. /

qy          A.5 utt1: Does it say something? /
```

Whether standard or not, semi-structured data is often designed to be machine-readable. As with unstructured data, the ultimate goal is to convert the data into a structured format and augment the data where necessary to prepare it for a particular research analysis.

File encoding

The last aspect to consider about corpus formats is **file encoding**. For a computer to display and process text characters, it must be encoded in a way that the computer can understand (*i.e.* 1's and 0's). Historically, character encoding schemes were developed to represent characters from specific character script sets (*e.g.* ASCII only includes characters from the English alphabet). However, as the need for a consistent and more inclusive way to encode characters from multiple languages and scripts became apparent, the Unicode standard, Unicode Transformation Format (UTF), was developed in the early 1990s. UTF encodings (UTF-8, UTF-16, and UTF-32) are now the most common way to encode text data and modern computers typically use them by default. Although other more script-specific encoding schemes can still be found in older data (e.g. ISO-8859, Windows-1252, Shift JIS).

When working with corpus data, it is important to know if the encoding scheme used for the data is compatible with your computing environment's default (most likely UTF). If it is not, you will need to convert the data to a compatible encoding scheme. Rest assured, there is support in R for converting between different encoding schemes if the need arises.

2.2 Information

Identifying an adequate corpus resource, in terms of content, licensing, and formatting, for the target research question is the first step in moving a quantitative text research project forward. The next step is to select the components or characteristics of this resource that are relevant

for the research and then move to organize the attributes of this data into a more informative format. This is the process of converting corpus data into a **dataset** –a tabular representation of particular attributes of the data as the basis for generating information. Once the data represented as dataset, it is often manipulated and transformed adjusting and augmenting the data such that it better aligns with the research question and the analytical approach.

2.2.1 Organization

Data alone is not informative. Only through explicit organization of the data in a way that makes relationships and meaning explicit does data become information. In this form, our data is called a dataset. This is a particularly salient hurdle in text analysis research. Many textual sources are unstructured or semi-structured, that is relationships that will be used in the analysis have yet to be purposefully drawn and organized from the data.

Tidy Data

The selection of the attributes from a corpus and the juxtaposition of these attributes in a relational format, or dataset, that converts data into information is known as **data curation**. The process of data curation minimally involves creating a base dataset, or *curated dataset*, which establishes the main informational associations according to philosophical approach outlined by Wickham (2014).

In this work, a **tidy dataset** refers both to the structural (physical) and informational (semantic) organization of the dataset. Physically, a tidy dataset is a tabular data structure, illustrated in Figure 2.3, where each *row* is an observation and each *column* is a variable that contains measures of a feature or attribute of each observation. Each cell where a given row-column intersect contains a *value* which is a particular attribute of a particular observation for the particular observation-feature pair also known as a *data point*.

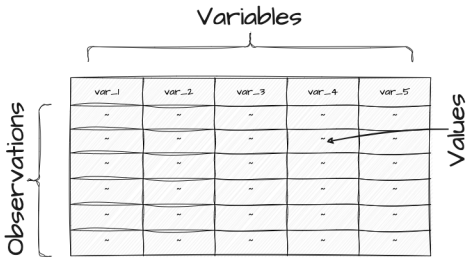


Figure 2.3: Visual summary of the tidy format.

In terms of semantics, columns and rows both contribute to the informational value of the dataset. Let's start with columns. In a tidy dataset, each column is a variable, an attribute

that can take on a number of values. Although variables vary in terms of values, they do not in type. A variable is of one and only one informational type. Statistically speaking, informational types are defined as **levels of measurement**, a classification system used to semantically distiguish between types of variables. There are four levels (or types) in this system: nominal, ordinal, interval, and ratio.

In practice, however, text analysis researchers often group these levels into three main informational types: categorical, ordinal, and numeric (S. T. Gries 2021). What do these informational types represent? **Categorical data** is for labeled data or classes that answer the question "what?" **Ordinal data** is categorical data with rank order that answers the question "what order?" **Numeric data** is ordinal data with equal intervals between values that answers the question "how much or how many?"

Let's look at an example of a tidy dataset. Using the criteria just described, let's see if we can identify the informational values (categorical, ordinal, or numeric) of the variables that appear in a snippet from the MASC corpus in dataset form in Table 2.1.

Table 2.1: MASC dataset variables.

title	modality	date	ref_num	word	pos	num_letters
Hotel California	Writing	2008	0	>	NN	1
Hotel California	Writing	2008	1	Hotel	NNP	5
Hotel California	Writing	2008	2	California	NNP	10
Hotel California	Writing	2008	3	Fact	NNP	4
Hotel California	Writing	2008	4	:	:	1
Hotel California	Writing	2008	5	Sound	NNP	5
Hotel California	Writing	2008	6	is	VBZ	2
Hotel California	Writing	2008	7	a	DT	1
Hotel California	Writing	2008	8	vibration	NN	9
Hotel California	Writing	2008	9	.	.	1

We have seven variables listed as headers for each of the columns. We could go one-by-one left-to-right but let's take another tack. Instead, let's identify all those variables that cannot be numeric –these are all the non-numeral variables: `title`, `modality`, `word`, and `pos`. The question to ask of these variables is whether they represent an order or rank. Since titles, modalities, words, and parts-of-speech are not ordered values, they are all categorical.

Now in relation to `date`, `ref_num`, and `num_letters`. All three are numerals, so they could be numeric. But they could also be numeral representations of ordinal data. Before we can move forward, we need to make sure we understand what each variable means and how it is measured, or **operationalized**. The variable name and the values can be helpful in this respect. `date` is what it sounds like, a date, and is operationalized as a year in the Gregorian calendar. And `num_letters` seems quite descriptive as well, number of letters, appearing as a

letter count. But in some cases it may be opaque as to what is being measured by the variable name alone, for example `ref_num`, and one will have to refer to the dataset documentation. In this case `ref_num` is a reference number operationalized as a unique identifier for each word per document in the corpus.

With this in mind, let's return to the question of whether `date`, `ref_num`, and `num_letters` are numeric or ordinal. Starting with the trickiest one, `date`, we can ask the question to identify numeric data: "how much or how many?". In the case of `date`, the answer is neither. A date is a point in time, not a quantity. So `date` is not numeric. But it does provide information about order. Hence, `date` is ordinal. `ref_num` is also ordinal because the question "what order?" can be asked of it. Finally, `num_letters` is numeric because it answers the question "how many?".

Let's turn to the second semantic value of a tidy dataset. In a tidy dataset, each row is an observation. But an observation of what? This depends on what the unit of observation is. That sounds circular, but its not. The **unit of observation** is simply the primary entity that is being observed. Without context, it can it can be identified in a dataset by looking at the level of specificity of the variable values and asking what each variable describes. When one variable appears to be the most individualized and other variables appear to describe that variable, then the most individualized variable is likely the unit of observation of the dataset, *i.e.* the meaning of each observation.

Applying these strategies to the Table in 2.1, we can see that each observation at its core is a word. We see that the values of each observation are the attributes of each word. `word` is the most individualized variable and the `pos` (part-of-speech), `num_letters`, and `ref_num` all describe the word.

The other variables `title`, `modality`, and `date` are not direct attributes of the word. Instead, they are attributes of the document in which the word appears. Together, however, they all provide information about the word.

> **Consider this**
> Data can be organized in many ways. It is important to make clear that data in tabular format in itself does not constitute a dataset, in the tidy sense we will be using. Can you think of examples of tabular information that would not be in a tidy format? What would be the implications of this for data analysis?

As we round out this section on data organization, it is important to stress that the purpose of curation is to represent the corpus data in an informative, tidy format. A curated dataset serves as a reference point making relationships explicit, enabling more efficient querying, and paving the way for further processing before analysis. In the subsequent section, we will highlight common approaches to modifying the curated dataset, either row-wise or column-wise, to make it more amenable to the particular aims of a given analysis.

2.2.2 Transformation

At this point have introduced the first step creating a dataset ready for analysis, data curation. However, a curated dataset is rarely the final organizational step before proceeding to statistical analysis. Many times, if not always, the curated dataset requires **data transformation** to derive or generate new data for the dataset. This process may incur row-wise (observation) or column-wise (variable) level changes, as illustrated in Figure 2.4.

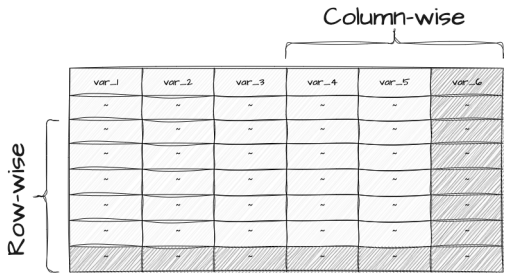


Figure 2.4: Visualization of row-wise and column-wise transformation operations on a dataset.

The results build on and manipulate the curated dataset to produce a *derived dataset*. While there is typically one curated dataset that serves as the base organizational dataset, there may be multiple derived datasets, each aligning with the informational needs of specific analyses in the research project.

In what follows, we will discuss the most common types of data transformation: text normalization, text tokenization, variable recoding, variable generation, and observation/ variable merging. Note, however, that the order in which these transformations are applied in a given research project is not fixed and will vary depending on the dataset and the research question(s) to be addressed.

Text normalization

The process of text normalization aims to prepare and standardize text. It is often a preliminary step in data transformation processes which include variables with text. The aim is to convert the text into a uniform format to reduce unwanted variation and noise.

Let's take a toy dataset, in Table 2.2, as an example starting point. In this dataset, we have two variables, `text_id` and `text`. It only has one observation.

Table 2.2: A toy dataset with two variables, `text_id` and `text`.

text_id	text
1	It's a beautiful day in the US, and our group decided to visit the famous Grand Canyon. As we reached the destination, Jane said, "I can't believe we're finally here!" The breathtaking view left us speechless; indeed, it was a sight to behold. During our trip, we encountered tourists from different countries, sharing stories and laughter. For all of us, this experience will be cherished forever.

The types of transformations we apply will depend on the specific needs of the project, but can include those found in Table 2.3.

Table 2.3: Common text normalization tasks

Task name	Relevant example	Typical purpose
Lowercasing	`"Text"` to `"text"`	Minimizing case sensitivity in subsequent analysis
Removal of Punctuation and Special Characters	`"Hello, World!"` to `"Hello World"`	Removing non-alphanumeric characters that may not carry semantic value
Adjustment of Forms	`"colour"` to `"color"`, `"it's"` to `"it is"`, `"1"` to `"one"`	Standardizing variations in spelling, contractions, and numeric forms to a common format
Stopword Removal	`"This is a sentence"` to `"This sentence"`	Discarding common words that usually do not contain meaningful semantic information

These transformations are column-wise operations, meaning they preserve the number of rows in the dataset. They also preserve the number of columns, but *do* change the values of the variables. These tasks should be applied with an understanding of how the changes will impact the analysis. For example, lowercasing can be useful for reducing differences between words that are otherwise identical, yet differ in case due to word position in a sentence ("The" versus "the"). However, lowercasing can also be problematic if the case of the word carries semantic value, such as in the case of "US" (United States) and "us" (first person plural pronoun). The same can be said for removing or adjusting particular characters and discarding stopwords.

Dive deeper
Stopwords are words that are so commonly used in a language that they tend not to contribute much to the meaning of a sentence. There are various predefined lists of stopwords for different languages available on the web and through R in the `stopwords`

package (Benoit, Muhr, and Watanabe 2021). However, it is important to note the criteria used to determine which words are considered stopwords in a particular resource may not fit a researcher's needs or the characteristics of the data. Learn more about stopwords in Kaur and Buttar (2018).

Let's be conservative and only apply lowercasing to our toy dataset as seen in Table 2.4.

Table 2.4: A toy dataset with two variables, `text_id` and `text`, where the text has been lowercased.

text_id	text
1	it's a beautiful day in the us, and our group decided to visit the famous grand canyon. as we reached the destination, jane said, "i can't believe we're finally here!" the breathtaking view left us speechless; indeed, it was a sight to behold. during our trip, we encountered tourists from different countries, sharing stories and laughter. for all of us, this experience will be cherished forever.

When text normalization steps are motivated and applied with foresight they serve to enhance the quality of the data and improves the reliability of subsequent transformation steps.

Text tokenization

Another text-oriented transformation step is **text tokenization**. This process involves adapting the text such that it reflects the target linguistic unit that will be used in the analysis. This is a row-wise operation expanding the number of rows, if the linguistic unit is smaller than the original variable, or reducing the number of rows, if the linguistic unit is larger than the original variable. At its core, tokenization is the process which enables the quantitative analysis of text.

Text variables can be tokenized at any linguistic level. To illustrate, consider our toy dataset from Table 2.4. We can tokenize the text at the sentence level, in Table 2.5, by splitting the text at the period followed by a space. This results in a dataset with four observations, one for each sentence in the original text.

Table 2.5: A toy dataset with two variables, `text_id` and `sentence`, where the text has been tokenized at the sentence level.

text_id	sentence
1	it's a beautiful day in the us, and our group decided to visit the famous grand canyon

text_id	sentence
1	as we reached the destination, jane said, “i can’t believe we’re finally here!” the breathtaking view left us speechless; indeed, it was a sight to behold
1	during our trip, we encountered tourists from different countries, sharing stories and laughter
1	for all of us, this experience will be cherished forever.

It is important to make explicit what the operationalization of our linguistic unit is as common terms such as sentence, word, *etc.* can be defined in different ways. For example, the sentence tokenization above is based on the assumption that sentences are separated by a period followed by a space. This is a suitable definition for this text, but likely will not be for other English text or for other languages/ writing scripts. For words, a very simple operationalization is to use whitespace separation (*e.g.* “I cannot believe it.” – [“I”, “cannot”, “believe”, “it.”]). However, this approach does not handle puntuation marks (*e.g.* [“it.”]) or contractions (*e.g.* [“can’t”]). A more sophisticated operationalization will be necessary for these, and possibly other, cases.

Another important token unit is the n-gram. Words or characters can be grouped into contiguous sequences with a moving window of a certain size n. Single unit windows are referred to as unigrams, two units as bigrams, three units as trigrams, and so on. Let’s tokenize our toy dataset at the bigram level for words using a simple whitespace separation for words, as seen in Table 2.6.

Table 2.6: A toy dataset with two variables, `text_id` and `bigram`, where the text has been tokenized at the bigram word level.

text_id	word
1	it’s a
1	a beautiful
1	beautiful day
1	day in
1	in the
1	the us
1	us and
1	and our
1	our group
1	group decided

In Table 2.6 we see that the first bigram is “it’s a” –the first two words (based on whitespace separation) in the text. The second bigram is “a toy” –the second and third words in the text.

This continues to the end of the text. N-gram tokenization can be useful to capture context that would otherwise would be lost from tokenizing words or characters at the unigram level.

Up to this point our tokens have been surface forms. That is, they are the actual words or characters as they appear in the text. However, we may want to reduce the tokens to their base form, removing their inflectional forms. This is known as **lemmatization**. For example, the word "run" is the lemma of the words "running", "runs", and "ran". Let's lemmatize the third sentence in our toy dataset. For comparison, `word` and `lemma` are shown side-by-side in Table 2.7.

Table 2.7: A toy dataset with two variables, `text_id` and `word`, where the text has been tokenized at the unigram word level and lemmatized.

text_id	word	lemma
1	during	during
1	our	our
1	trip	trip
1	we	we
1	encountered	encounter
1	tourists	tourist
1	from	from
1	different	different
1	countries	country
1	sharing	share
1	stories	story
1	and	and
1	laughter	laughter

Case study
Inflectional family size is the number of inflectional forms for a given word and can be calculated from a corpus by counting the number of surface forms for each lemma in the corpus (Kostić, Marković, and Baucal 2003). Baayen, Feldman, and Schreuder (2006) found that words with larger inflectional family size are associated with faster word recognition times in lexical processing tasks.

Together tokenization and lemmatization are powerful tools for transforming text. If our dataset contains more robust linguistic annotation or that annotation can be generated (see Section 2.2.2), this information can also be leveraged to tokenize language into a format that is easier to explore and quantify in an analysis.

Variable recoding

Recoding is the process of transforming the values of one or more variables into new values which are more amenable to analysis. The aim is to simplify complex variables, making it easier to identify patterns and trends relevant for the research question. This is a column-wise operation which can be applied to categorical or numeric variables.

Let's return to the MASC dataset and demonstrate recoding of categorical and numeric variables. In Table 2.1 the `pos` variable whose values represent the part-of-speech (POS) of each token in the text. The measure is a POS tag from the Penn Treebank tagset (Marcus, Santorini, and Marcinkiewicz 1993). This tagset makes twelve major and 45 minor grammatical class distinctions. In an analysis that aims to explore only major class distinctions, it would be useful to recode the `pos` variable into major classes only (*i.e.* noun, pronoun, adjective, verb, adverb, *etc.*) to facilitate queries, summaries, and visualizations.

Table 2.8: A toy dataset with three variables, `text_id`, `pos`, `major_pos`, where the `pos` variable has been recoded into major grammatical classes `major_pos`.

title	modality	date	ref_num	word	pos	major_pos	num_letters
Hotel California	Writing	2008	0	>	NN	noun	1
Hotel California	Writing	2008	1	Hotel	NNP	noun	5
Hotel California	Writing	2008	2	California	NNP	noun	10
Hotel California	Writing	2008	3	Fact	NNP	noun	4
Hotel California	Writing	2008	4	:	:	punctuation	1
Hotel California	Writing	2008	5	Sound	NNP	noun	5
Hotel California	Writing	2008	6	is	VBZ	verb	2
Hotel California	Writing	2008	7	a	DT	determiner	1
Hotel California	Writing	2008	8	vibration	NN	noun	9
Hotel California	Writing	2008	9	.	.	punctuation	1

In Table 2.8, the `pos` variable has been recoded into major grammatical classes. The `major_pos` variable is a categorical variable with 12 levels, one for each major grammatical class in the Penn Treebank tagset. While the demonstration here demonstrates the simplification of a categorical variable, recoding can also be used to transliterate categorical variables. Continuing with the theme of POS tags, the `pos` variable could be recoded into a different tagset, such as the Universal Dependencies tagset (Nivre et al. 2016).

Now, let's look at recoding the numeric variable `num_letters`. This variable represents the number of letters in each token. In the MASC dataset, the `num_letters` variable is a numeric variable with a range of values from 1 to 21. In some analyses, it may be useful to recode this variable into discrete categories, or bins, such as short, medium, and long words.

Table 2.9: The MASC dataset with the `num_letters` variable recoded into three categories: short, medium, and long words in `word_length`.

title	modality	date	ref_num	word	pos	major_pos	num_letters	word_length
Hotel California	Writing	2008	0	>	NN	noun	1	short
Hotel California	Writing	2008	1	Hotel	NNP	noun	5	medium
Hotel California	Writing	2008	2	California	NNP	noun	10	long
Hotel California	Writing	2008	3	Fact	NNP	noun	4	medium
Hotel California	Writing	2008	4	:	:	punctuation	1	short
Hotel California	Writing	2008	5	Sound	NNP	noun	5	medium
Hotel California	Writing	2008	6	is	VBZ	verb	2	short
Hotel California	Writing	2008	7	a	DT	determiner	1	short
Hotel California	Writing	2008	8	vibration	NN	noun	9	long
Hotel California	Writing	2008	9	.	.	punctuation	1	short

In Table 2.9 the variable `word_length` appears with the values `short`, `medium`, and `long`. This is now a categorical variable of type ordinal. Of note, is that the operational definition of used to create these word length bins should be made explicit in the documentation of the dataset.

In sum, recoding is a useful data transformation technique that can be used to simplify complex variables, making it easier to identify patterns and trends relevant for the research question.

Variable generation

The process of variable generation aims to augment existing variables or create new ones, and as such is a column-wise operation. Generation can include applying calculations or extracting relevant information from existing variables or enhancing text variables with linguistic annotation. Simplifying a bit, generation helps make implicit attributes explicit. The results of this process enables direct access during analysis to features that were otherwise hidden or difficult to access.

Let's highlight a some common calculation and extraction examples that generate variables. First, let's look at the calculation of measures. In text analysis, measures are often used to describe the properties of a document or linguistic unit. For example, the number of words in a corpus document, the lengths of sentences, the number of clauses in a sentence, *etc.*. In turn, these measures can be used to calculate other measures, such as lexical diversity or syntactic complexity measures.

In terms of extraction, the goal is to distill relevant information from existing variables. For example, extracting the year from a date variable, or extracting the first name from a full name variable. In text analysis, extraction is often used to extract information from text variables. Say we have a dataset with a variable containing conversation utterances. We may want to extract some characteristic from those utterances and capture their occurrence in a new variable.

But what if we want to extract linguistic information from a text variable that is not explicitly present in the text? This is where linguistic annotation comes in. Linguistic annotation is the process of enriching text with linguistic information, such as morphological features, part-of-speech tags, syntactic structure, *etc.*. This can be done manually by linguist coders and/ or done using natural language processing (NLP) tools, many of which are available in R (see Chapter 7).

To illustrate the process of generating linguistic annotation with existing tools, I will use the plain text version of the MASC. In Table 2.10, the text has been organized into a dataset and tokenized into sentences. The `text_id` variable is a unique identifier for each document, and the `sentence_id` variable is a unique identifier for each sentence.

Table 2.10: A MASC sample document in dataset tokenized into sentences.

text_id	sentence_id	sentence
1	1	>Hotel California Fact: Sound is a vibration.
1	2	Sound travels as a mechanical wave through a medium, and in space, there is no medium.
1	3	So when my shuttle malfunctioned and the airlocks didn't keep the air in, I heard nothing.
1	4	After the first whoosh of the air being sucked away, there was lightning, but no thunder.
1	5	Eyes bulging in panic, but no screams.

Applying a pre-trained model from the Universal Dependencies (UD)[23][24] project, we can generate linguistic annotation for each token in the MASC.

[23]https://universaldependencies.org/

[24]The Universal Dependency project is an effort to develop cross-linguistically consistent treebank annotation for many languages. The project has developed a set of annotation guidelines and a set of tools for generating linguistic annotation. The project has also developed a set of pre-trained models for many languages.

Table 2.11: Automatic linguistic annotation for grammatical category and syntactic structure for an example English sentence from the MASC.

doc_id	sentence_id	token_id	token	xpos	features	syntactic_relation
1	4	1	After	IN	NA	mark
1	4	2	the	DT	Definite=Def|PronType=Art	det
1	4	3	first	JJ	Degree=Pos|NumType=Ord	amod
1	4	4	whoosh	NN	Number=Sing	nsubj:pass
1	4	5	of	IN	NA	case
1	4	6	the	DT	Definite=Def|PronType=Art	det
1	4	7	air	NN	Number=Sing	nmod
1	4	8	being	VBG	VerbForm=Ger	aux:pass
1	4	9	sucked	VBN	Tense=Past|VerbForm=Part|Voice=Pass	advcl
1	4	10	away	RB	NA	advmod
1	4	11	,	,	NA	punct
1	4	12	there	EX	NA	expl
1	4	13	was	VBD	Mood=Ind|Number=Sing|Person=3|Tense=Past|VerbForm=Fin	aux
1	4	14	lightning	NN	Number=Sing	nsubj
1	4	15	,	,	NA	punct
1	4	16	but	CC	NA	cc
1	4	17	no	DT	NA	det
1	4	18	thunder	NN	Number=Sing	conj
1	4	19	.	.	NA	punct

The annotated dataset now includes the key variables `xpos` (Penn treebank tags), `features` (morphological features), and `syntactic_relation`. The results of this process can then be further transformed as need be to fit the needs of the analysis.

A word of caution: automated linguistic annotation offers rapid access to abundant and highly dependable linguistic data for numerous languages. However, linguistic annotation tools are not infallible. They are tools developed by training computational algorithms to identify patterns in previously annotated and verified datasets, resulting in a language model. This model is then employed to predict linguistic annotations for new language data (as seen in Table 2.11). The accuracy of the linguistic annotation heavily relies on the congruence between the language sampling framework of the trained data and the language data set to be automatically annotated.

Observation/ variable merging

The processing of merging datasets is a transformation step which can be row-wise or column-wise. Row-wise merging is the process of combining datasets by appending observations from

one dataset to another. Column-wise merging is the process of combining datasets by appending variables from one dataset to another. In either case, merging provides a way to enrich a dataset by incorporating additional information.

To merge in row-wise manner the datasets involved in the process must have the same variables and variable types. This process is often referred to as **concatenating datasets**. It can be thought of as stacking datasets on top of each other to create a larger dataset. Remember, having the sample variables and variable types is not the same has having the same values.

Take, for example, a case when a corpus resource contains data for two populations. In the course of curating and transforming the datasets it may make more sense to work with the datasets separately. However, when it comes time to analyze the data, it may be more convenient to work with the datasets as a single dataset. In this case, the datasets can be concatenated to create a single dataset.

To illustate, consider the toy datasets in Table 2.12 and Table 2.13.

Table 2.12: Toy dataset of written text data.

participant_id	text_id	modality	text
P1	T1	Written	Technology has revolutionized our lives in many ways. It has made communication easier, faster, and more efficient.
P3	T3	Written	Climate change is a pressing issue that affects everyone on Earth. We must take immediate action to reduce our carbon footprint.
P5	T5	Written	Education is the key to personal and societal growth. Investing in quality education will lead to a brighter future for all.

Table 2.13: Toy dataset of spoken text data.

participant_id	text_id	modality	text
P2	T2	Spoken	Hello, my name is X. I am a software engineer working at XYZ company.
P4	T4	Spoken	Hi, I'm X, and I work as a project manager. My main responsibility is to ensure that projects are completed on time and within budget.
P6	T6	Spoken	Hi, my name is X, and I'm a teacher. I teach English at a local high school.

These datasets, in Table 2.12 and Table 2.13, contain the same variables and variable types, but different observations –one in which the sample contains written language and the other spoken. Conveniently, they can be concatenated to create a single dataset that contains all of the observations, as seen in Table 2.14.

Table 2.14: Toy dataset of written and spoken text data concatenated.

participant_id	text_id	modality	text
P1	T1	Written	Technology has revolutionized our lives in many ways. It has made communication easier, faster, and more efficient.
P3	T3	Written	Climate change is a pressing issue that affects everyone on Earth. We must take immediate action to reduce our carbon footprint.
P5	T5	Written	Education is the key to personal and societal growth. Investing in quality education will lead to a brighter future for all.
P2	T2	Spoken	Hello, my name is X. I am a software engineer working at XYZ company.
P4	T4	Spoken	Hi, I'm X, and I work as a project manager. My main responsibility is to ensure that projects are completed on time and within budget.
P6	T6	Spoken	Hi, my name is X, and I'm a teacher. I teach English at a local high school.

Merging datasets can be performed in a column-wise manner as well. In this process, the datasets need not have the exact same variables and variable types, rather it is required that the datasets share a common variable of the same informational type that can be used to index the datasets. This process is often referred to as **joining datasets**.

Corpus resources often include metadata in stand-off annotation format. That is, the metadata is not embedded in the corpus files, but rather is stored in a separate file. The metatdata and corpus files will share a common variable which is used to join the metadata with the corpus files.

To exemplify, here's another toy dataset that shares the `participant_id` index with the previous dataset in Table 2.14 and includes the variables `native_speaker_eng`, `age`, and `gender`:

Table 2.15: Toy dataset of participant data with a shared variable `participant_id` to index the datasets.

participant_id	native_speaker_eng	age	gender
P1	Yes	28	M
P2	No	35	M
P3	Yes	42	F
P4	No	26	F
P5	Yes	31	M
P6	No	39	F

This dataset provides additional information about each participant, such as their English native speaker status, age, and gender.

Since the two datasets share the `participant_id` variable, we can merge them to create a new dataset that combines the information from both datasets, as we see in Table 2.16.

Table 2.16: Joining variables from two datasets based on a shared index variable.

participant_id	text_id	modality	text	native_speaker_eng	age	gender
P1	T1	Written	Technology has revolutionized our lives in many ways. It has made communication easier, faster, and more efficient.	Yes	28	M
P3	T3	Written	Climate change is a pressing issue that affects everyone on Earth. We must take immediate action to reduce our carbon footprint.	Yes	42	F
P5	T5	Written	Education is the key to personal and societal growth. Investing in quality education will lead to a brighter future for all.	Yes	31	M
P2	T2	Spoken	Hello, my name is X. I am a software engineer working at XYZ company.	No	35	M
P4	T4	Spoken	Hi, I'm X, and I work as a project manager. My main responsibility is to ensure that projects are completed on time and within budget.	No	26	F
P6	T6	Spoken	Hi, my name is X, and I'm a teacher. I teach English at a local high school.	No	39	F

Another common case where joining datasets is useful is when there are external resources that can be used to enrich the dataset. For example, a dataset of text data may include a variable that identifies the language of each document. This variable can be used to join the dataset with a dataset of language metadata, such as the number of speakers of each language, as long as the language metadata dataset includes a variable that identifies the language of each document in the same format as the language variable in the text dataset.

In sum, the transformation steps described here collectively aim to produce higher quality datasets that are relevant in content and structure to submit to analysis. The process may include one or more of the previous transformations but is rarely linear and is most often iterative. It is typical to do some normalization then generation, then recoding, and then return to normalizing, and so forth. This process is highly idiosyncratic given the characteristics of the curated dataset and the ultimate goal for the derived dataset(s).

2.3 Documentation

As we have seen in this chapter, acquiring corpus data and converting that data into information involves a number of conscious decisions and implementation steps. As a favor to ourselves, as researchers, and to the research community, it is crucial to document these decisions and steps. This makes it both possible to retrace our own steps and also provides a guide for future researchers that want to reproduce and/ or build on your research. A programmatic approach to quantitative research helps ensure that the implementation steps are documented and reproducible but it is also vital that the decisions that are made are documented as well. This includes data origin information for the acquired corpus data and data dictionaries for the curated and derived datasets.

2.3.1 Data origin

Data acquired from corpus resources should be accompanied by information about the **data origin**. Table 2.17 provides a list of the types of information that should be included in the data origin information.

Table 2.17: Data origin information.

Information	Description
Resource name	Name of the corpus resource.
Data source	URL, DOI, *etc.*
Data sampling frame	Language, language variety, modality, genre, *etc.*

Information	Description
Data collection date(s)	The date or date range of the data collection.
Data format	Plain text, XML, HTML, *etc.*
Data schema	Relationships between data elements: files, folders, *etc.*
License	CC BY, CC BY-NC, *etc.*
Attribution	Citation information for the data source.

For many corpus resources, the corpus documentation will include all or most of this information as part of the resource download or documented online. If this information is not present in the corpus resource or you compile your own, it is important to document this information yourself. This information can be documented in file, such as a plain text file or spreadsheet, that is included with the corpus resource.

2.3.2 Data dictionaries

The process of organizing the data into a dataset, curation, and modifications to the dataset in preparation for analysis, transformation, each include a number of project-specific decisions. These decisions should be documented.

On the one hand each dataset that is created should have a **data dictionary** file. A data dictionary is a document, usually in a spreadsheet format, that describes the variables in a dataset. The key information that should be included in a data dictionary is provided in Table 2.18.

Table 2.18: Data dictionary information.

Information	Description
Variable name	The name of the variable as it appears in the dataset, *e.g.* `participant_id`, `modality`, *etc.*
Readable variable name	A human-readable name for the variable, *e.g.* 'Participant ID', 'Language modality', *etc.*
Variable type	The type of information that the variable contains, *e.g.* 'categorical', 'ordinal', *etc.*
Variable description	A prose description expanding on the readable name and can include measurement units, allowed values, *etc.*

Organizing this information in a tabular format, such as a spreadsheet, can make it easy for others to read and understand your data dictionary.

On the other hand, the data curation and transformation steps should be documented in the code that is used to create the dataset. This is one of the valuable features of a programmatic

approach to quantitative research. The transparency of this documentation is enhanced by using **literate programming** strategies to intermingling prose descriptions and code the steps in the same, reproducible document.

By providing a comprehensive data dictionary and using a programmatic approach to data curation and transformation, you ensure that others can easily understand and work with your dataset, facilitating collaboration and reproducibility.

Summary

In this chapter we have focused on data and information –the first two components of DIKI Hierarchy. This process is visualized in Figure 2.5.

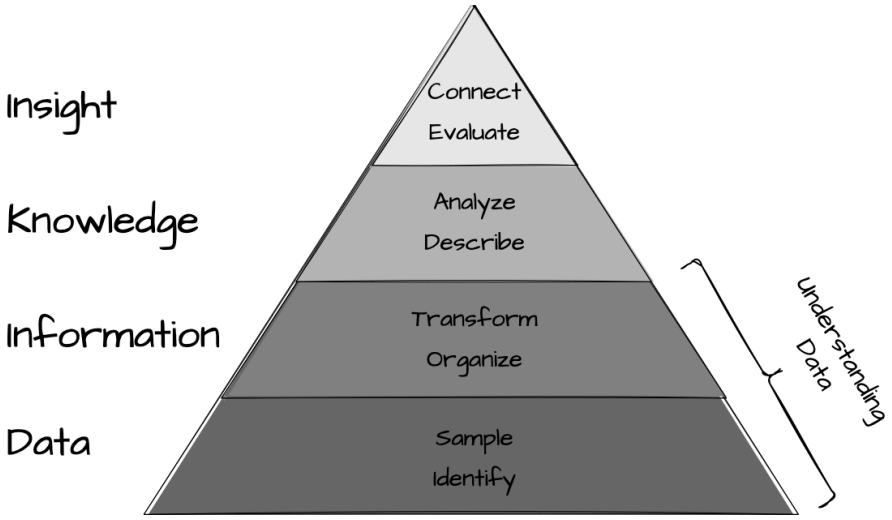


Figure 2.5: Understanding data: visual summary

First a distinction is made between populations and samples, the latter being a intentional and subjective selection of observations from the world which attempt to represent the population of interest. The result of this process is known as a corpus. Whether developing a corpus or selecting an existing a corpus it is important to vet the sampling frame for its applicability and viability as a resource for a given research project.

Once a viable corpus is identified, then that corpus is converted into a curated dataset which adopts the tidy dataset format where each column is a variable, each row is an observation, and the intersection of columns and rows contain values. This curated dataset serves to establish the base informational relationships from which your research will stem.

The curated dataset will most likely require transformations which may include normalization, tokenization, recoding, generation, and/ or merging to enhance the usefulness of the information to analysis. A derived dataset or set of datasets will the result from this process.

Finally, documentation should be implemented at the acquisition, curation, and transformation stages of the analysis project process. The combination of data origin, data dictionary, and literate programming files establishes documentation of the data and implementation steps to ensure transparent and reproducible research.

Activities

In the following activities you will learn how to read, inspect, and write data and datasets in R using reproducible strategies.

Recipe
What: Reading, inspecting, and writing data[a]
How: Read Recipe 2 and participate in the Hypothes.is online social annotation.
Why: To use literate programming in Quarto to work with R coding strategies for reading, inspecting, and writing datasets.

[a]https://qtalr.github.io/qtalrkit/articles/recipe-2.html

Lab
What: Reading, inspecting, and writing data[a]
How: Clone, fork, and complete the steps in Lab 2.
Why: To read datasets from packages and from plain-text files, inspect and report characteristics of datasets, and write datasets to plain-text files.

[a]https://github.com/qtalr/lab-2

Questions

Conceptual questions

- What is the difference between a population and a sample?
- Why is it important to vet a corpus before using it in a research project?
- What is a curated dataset in the context of linguistic research?
- What is the difference between a variable, an observation, and a value?
- Why is it important to identify the informationasl types of variables in a dataset?

- What kinds of transformations may be performed on a curated dataset to enhance its usefulness for analysis?
- What is an transformed dataset and why is it important in linguistic research?
- Why is documentation important in the process of conducting linguistic analysis?
- How does a programmatic approach enhance documentation in linguistic research?
- How does documenting the corpus data and the curated and derived datasets contribute to transparent and reproducible research in linguistics?

Technical questions

- Creating a sample corpus.
- Writing a corpus documentation.
- Converting a corpus to a derived dataset.
- Writing a data dictionary.
- Transforming a derived dataset.
- Merging datasets.
- Writing a dataset to disk.
- Consider (an example dataset) and its data dictionary, write a script to read the dataset, inspect it, and write it to disk.
- Consider a dataset and its data dictionary what appears to be the unit of analysis and the unit of observation?

3 Approaching analysis

Draft

Ready for review.

> Statistical thinking will one day be as necessary for efficient citizenship as the ability to read and write.
>
> — H.G. Wells

Outcomes

- Recall the fundamental concepts and principles of statistics in data analysis.
- Articulate the roles of diagnostic, analytic, and interpretive statistics in quantitative analysis.
- Compare the similarities and differences between analytic approaches to data analysis.

In this chapter I will build on the notions of data and information from the previous chapter. The aim of analysis is to derive knowledge from information, the next step in the DIKI Hierarchy. Where the creation of information from data involves human intervention and conscious decisions, as we have seen, deriving knowledge from information involves another level of intervention. The goal is to break down complex information into simpler components which are more readily interpretable. In what follows, we will cover the main steps in the process of analysis. The first is to inspect the data to ensure its quality and understand its characteristics. The second is to interrogate the data to uncover patterns and relationships and interpret the findings. To conclude this chapter I will outline methods to and the importance of communicating the analysis results and procedure in a transparent and reproducible manner.

Swirl lesson
What: Data visualization[a]
How: In the R Console pane load `swirl`, run `swirl()`, and follow prompts to select the lesson.
Why: To explore data visually in text and in graphics.

[a]https://github.com/qtalr/swirl

3.1 Diagnose

The purpose of diagnostic measures is to inspect your data to ensure its quality and understand its characteristics. There are two primary types of diagnostic measures: verfication and description. Verification methods are applied to catch missing or erroneous data while descriptive methods are used to gain a better understanding of the data. Although treated in two separate sections, in practice these methods are complementary and are often addressed in tandem.

To ground this discussion I will introduce a new dataset. This dataset is drawn from the Barcelona English Language Corpus (BELC) (Muñoz 2006), which is found in the TalkBank repository. I've selected the "Written composition" task from this corpus which contains 80 writing samples from 36 second language learners of English at different ages. Participants were given the task of writing for 15 minutes on the topic of "Me: my past, present and future". Data was collected for participants from one to three times over the course of seven years (at 10, 12, 16, and 17 years of age).

In Table 3.1 we see the data dictionary for the BELC dataset which reflects structural and transformational steps I've done so we start with a tidy dataset with `word` as the unit of observation.

Table 3.1: Data dictionary for the BELC dataset.

variable	name	description	variable_type
part_id	Participant ID	Unique identifier for each participant	categorical
sex	Participant's sex	Sex of the participant	categorical
group	Time group	Longitudinal group to which the participant belongs	ordinal
month_age	Participant's age in months	Age of the participant in months	numeric
utt_id	Utterance ID	Unique identifier for each utterance	numeric
word_id	Word ID	Unique identifier for each word within an utterance	numeric
word	Word	The word spoken by the participant	categorical
lemma	Word lemma	Base form of the word	categorical
pos	Part of speech	Grammatical category of the word	categorical

The data dictionary provides a easily accessible overview of the dataset. This includes a human-readable mapping from variable names to variable descriptions. Further, it provides

information about the type of variable (e.g., categorical, ordinal, numeric). As we will see the informational type of variables is key to diagnostic measures, as well as all other components of analysis.

3.1.1 Verify

Although a dataset has undergone curation and transformation, it is still important to verify the data. This is a process of checking the data to ensure that it is accurate and complete. In the case that it is not, consideration should be given to how to address the issues.

The most basic and usually the first step is to check for **missing data** to ensure that all necessary data points are present. In Table 3.2, there are missing values for the `lemma` and `pos` variables in the BELC dataset.

Table 3.2: Summary output for missing values in the BELC dataset.

variable	type	n_missing	complete_rate
part_id	character	0	1.000
sex	character	0	1.000
group	character	0	1.000
word	character	0	1.000
lemma	character	79	0.985
pos	character	23	0.996
month_age	numeric	0	1.000
utt_id	numeric	0	1.000
word_id	numeric	0	1.000

There are two primary approaches to dealing with missing data: deletion and recoding. Since these missing values account for only 1.5% and 0.4% of the data respectively, we might be safe to remove these observations. Another approach is to recode the missing values by either applying a unique value for missing values (e.g., `NULL`) or by imputing values. Imputing values is usually done by replacing missing values with some middle-of-the-road value (e.g., mean, median, mode), but other, more nuanced approaches are possible.

> **Dive deeper**
> For more information on missing data, see the 🔧 in this book.

In either case, it is important to consider the implications of missing data for the analysis. For example, if the missing data is not at random or include a sizeable portion of the values of interest, then the analysis may be biased.

Value coding schemes, annotation errors, or other issues may result in **anomalies** in the data. These are values that are unusual, unexpected, or inconsistent with the rest of the data or effect the treatment of the data for the particular analysis to be performed.

For categorical variables, this may include values that are not expected or are not in the set of values that are expected. A summary of the values for a given variable can be used as a first step to identify anomalies. In Table 3.3, we see the minimum and maximum number of characters and the number of unique values for each categorical variable in the BELC dataset.

Table 3.3: Summary output for categorical variables in the BELC dataset.

variable	min_chars	max_chars	num_unique
part_id	3	3	36
sex	4	6	2
group	2	2	4
word	1	20	913
lemma	1	20	774
pos	1	9	38

From our knowledge of the data, we can gauge whether these values are expected. For example, `sex` has two values; likely corresponding to some coding of 'male' and 'female'. The variable `part_id` has 36 distinct values, which is expected since there are 36 participants and `group` has four, corresponding to the longitudinal time groups. It is also possible to gauge the expected values for `lemma` as we know that these are the base words and should be less than the number of words in the dataset.

Further verfication of the categorical variables is need, of course. This may include aggregating the data to see the distribution of values and/ or checking the values against the documentation.

Let's now consider numeric variables. Numeric variables, by their very nature, do not lend themselves to the same type of summary used for categorical variables (*i.e.* character lengths, number of unique values, or aggregation) to detect anomalies. For numeric variables there are two types of anomalies that we will consider: outliers and errors in coding. **Outliers** are anomalies that are extreme values that are not representative of the great majority of the data points. To determine what is extreme, we need to consider the distribution of the data, that is, the range of values and the frequency of values. It is rarely the case that we can eyeball the distribution of the data based on raw values. Instead, a combination of summary statistics and visualizations are used to determine the distribution of the data. For this reason, the detection of outliers is often carried out as part of the descriptive assessment of the data, as we will see in Section 3.1.2.

On the other hand, coding anomalies are values that are not expected or are not in the set of values that are expected. These can sometimes be detected by visual inspection of the data. For example, in Table 3.4, we see the first 10 observations for each variable in the BELC dataset.

Table 3.4: First 10 observations for variables in the BELC dataset.

part_id	sex	group	month_age	utt_id	word_id	word	lemma	pos
L01	female	T2	153	0	0	I	I	pro:sub
L01	female	T2	153	0	1	was	be	cop
L01	female	T2	153	0	2	born	born	adj
L01	female	T2	153	0	3	in	in	prep
L01	female	T2	153	0	4	Barcelona	Barcelona	n:prop
L01	female	T2	153	0	5	and	and	coord
L01	female	T2	153	0	6	I	I	pro:sub
L01	female	T2	153	0	7	live	live	v
L01	female	T2	153	0	8	in	in	prep
L01	female	T2	153	0	9	Barcelona	Barcelona	n:prop

Leaving `month_age` aside, we see that the other two numeric variables `utt_id` and `word_id` index utterances and words respectively. However, in contrast to `part_id` which is a categorical variable as it serves as a unique identifier for each participant, these variables are numeric as they serve to not only index utterances and words but also to provide a measure of how many utterances or words have been produced. Seen in this light, `0` for the first value of `utt_id` and `word_id` is unexpected. To adjust for this, we can add `1` to each value of these variables.

3.1.2 Describe

The goal of descriptive statistics is to summarize the data in order to understand and prepare the data for the analysis approach to be performed. This is accomplished through a combination of statistic measures and/ or tabular or graphic summaries. The choice of descriptive statistics is guided by the type of data, as well as the question(s) being asked of the data.

To that end, let's consider a reconfiguration of the BELC dataset, in Table 3.5, which will provide a more illustrative dataset.

Table 3.5: First 10 observations of the reconfigured BELC dataset.

essay_id	part_id	sex	group	tokens	types	ttr	prop_l2
E1	L01	female	T2	79	46	0.582	0.987
E2	L02	female	T1	18	18	1.000	0.667

essay_id	part_id	sex	group	tokens	types	ttr	prop_l2
E3	L02	female	T3	101	53	0.525	1.000
E4	L05	female	T1	20	17	0.850	0.900
E5	L05	female	T3	158	80	0.506	0.987
E6	L05	female	T4	184	94	0.511	0.995
E7	L07	male	T3	98	60	0.612	1.000
E8	L07	male	T4	134	84	0.627	0.978
E9	L10	female	T1	38	28	0.737	0.974
E10	L10	female	T3	118	74	0.627	1.000

In this new configuration, the unit of observation is now `essay_id`. Each of the following variable are attributes or measures of this variable. The new variables in this dataset are aggregates of the previous BELC dataset: `tokens` is the number of total words, `types` is the number of unique words, `ttr` is the ratio of unique words to total words. This is known as the Type-Token Ratio and it is a standard metric for measuring lexical diversity. Finally, the proportion of L2 words (English) to the total words (tokens) is provided in `prop_l2`.

In descriptive statistics, there are four basic questions that are asked of each of the variables in the dataset. Each correspond to a different type of descriptive measure.

1. Central Tendency: Where do the data points tend to be located?
2. Dispersion: How spread out are the data points?
3. Distribution: What is the overall shape of of the data points?
4. Interdependence: How are these data points related to other data?

Central tendency

The central tendency is measure which aims to summarize the data points in a variable as the most representative, middle or most typical value. There are three common measures of central tendency: the mode, mean and median. Each differ in how they summarize the data points.

The **mode** is the value, or values, that appears most frequently in a set of values. If there are multiple values with the highest frequency, then the variable is said to be multimodal. The most versatile of the central tendency measures as it can be applied to all levels of measurement, although the mode is not often used for numeric variables as it is not as informative as other measures.

The more common measures for numeric variables are the mean and the median. The **mean** is a summary statistic calculated by summing all the values and dividing by the number of values. The **median** is calculated by sorting all the values in the variable and then selecting the middle value. Given that the mean and median are calculated differently, they will not

Table 3.6: Central tendency and dispersion of the variables in the BELC dataset

(a) Categorical variables

variable	top_counts	norm_entropy
essay_id	E1: 1, E10: 1, E11: 1, E12: 1	1.00
part_id	L05: 3, L10: 3, L11: 3, L12: 3	1.00
sex	fem: 48, mal: 32	0.97
group	T1: 25, T3: 24, T2: 16, T4: 15	0.98

(b) Numeric variables

variable	mean	median	sd	iqr
tokens	67.62	56.50	44.20	61.25
types	41.85	38.50	23.03	31.50
ttr	0.68	0.66	0.13	0.15
prop_12	0.96	0.99	0.10	0.03

always yield the same result. Differences that appear between the mean and median will be of interest to us later in this chapter.

Dispersion

The mean, median, and mode provide summary information where data points tend to be located. However, they do not provide us with any understanding as to how representative this value is. To provide this context, the spread of the values around the central tendency, or **dispersion**, is calculated.

For categorical variables, the spread is framed in terms of how balanced the values are across the levels. One way to do this is to calculate the (normalized) entropy. **Entropy** is a measure of uncertainty. The more balanced the values are across the levels, the higher the entropy. The less balanced the values are across the levels, the lower the entropy. Normalized entropy scores range from 0 to 1, with 0 indicating that all the values are the same and 1 indicating that all the values are different.

The most common measure of dispersion for numeric variables is the **standard deviation**. The standard deviation is calculated by taking the square root of the variance. The **variance** is the average of the squared differences from the mean. So, more succinctly, the standard deviation is a measure of the spread of the values around the mean. Where the standard deviation is anchored to the mean, the **interquartile range** (IQR) is tied to the median. The median represents the sorted middle of the values, in other words the 50th percentile. The IQR is the difference between the 75th percentile and the 25th percentile. Again, just as the mean and the median, the standard deviation and the IQR are calculated in different ways, they are not always the same.

Let's now consider the relevant central tendency and dispersion of the variables in the BELC dataset in Table 3.6.

In Table 3.6a we see the measures for categorical variables. The `top_counts` variable gives us a short list of the most frequent levels of the variable. From `top_count` we can gather whether the variable has one mode or is multimodel. Both `essay_id` and `part_id` have the same most

frequent value for the levels listed. On the other hand, `sex` and `group` have a single mode. We can also appreciate the dispersion of these variables based on the `norm_entropy` of each variable. `essay_id` is completely balanced across the levels, so it has a normalized entropy of 1. the other variables are not as balanced, but still quite balanced as the normalized entropy is close to 1.

In Table 3.6b the numeric variables have a column for the mean, median, standard deviation, and IQR for each. The variable `tokens` has a larger difference between the mean and median than the other variables and the standard deviation is relatively large suggesting that the values are more spread out around the mean. In the case of `ttr` the mean and median are quite close and the standard deviation is relatively small suggesting that the values are more tightly clustered around the mean.

When interpreting these summary values, it is important to only directly compare column-wise. That is, focusing only on a single variable, not across variables. Each variable, as is, is measured on a different scale and only relative to itself can we make sense of the values.

However, we can transform the central tendency and dispersion scores for numeric variables to make them more comparable by standardizing the scale of the values. **Standardization** is a scale-based transformation that changes the scale of the values to a common scale, or *z-scores*. It involves two separate transformations: centering and scaling. **Centering** is a transformation that subtracts the mean or median from each value. The result is a mean and median of zero. **Scaling** is a transformation that divides each value by the standard deviation or IQR.

In Table 3.7, we see the same summary statistics as in Table 3.6b, but the values have been standardized for the mean and standard deviation. The mean is now zero and the standard deviation is one. This allows us to compare the median and IQR of the variables more directly.

Table 3.7: Standardized central tendency and dispersion of numeric variables

variable	mean	median	sd	iqr
tokens	0	-0.25	1	1.39
types	0	-0.15	1	1.37
ttr	0	-0.19	1	1.14
prop_l2	0	0.25	1	0.27

One more caveat to keep in mind is that we need to be mindful of the nature of the data being standardized and what the standardized values mean. For example, the variables `tokens` and `types` were originally counts. But the standardized values are not interpretable as counts, they are now on a different scale –specifically a z-score scale. In the same way since the `ttr` and `prop_l2` variables were originally proportions, the standardized values are also not interpretable as proportions. One additional twist, however, is that the original scales for

these pairs of variables were not the same: `tokens` and `types` were counts, but `ttr` and `prop_l2` were proportions. So, even though the standardized values are on the same scale, they are not directly comparable.

Beyond comparing central tendency and dispersion across variables, standarization is useful for analytic statistics to mitigate the influence of variables with large values. In some cases, the statistical method will require standardization of variables before analysis.

Distributions

Summary statistics of the central tendency and dispersion of a variable provide a sense of the most representative value and how spread out the data is around this value. However, to gain a more comprehensive understanding of the variable, it is key to consider the frequencies of all the data points. The **distribution** of a variable is the pattern or shape of the data that emerges when the frequencies of all data points are considered. This can reveal patterns that might not be immediately apparent from summary statistics alone. Understanding the frequency and distribution of data points is vital as it informs subsequent choices of statistical analysis and evaluative methods, ensuring they are appropriate for the specific characteristics of the data.

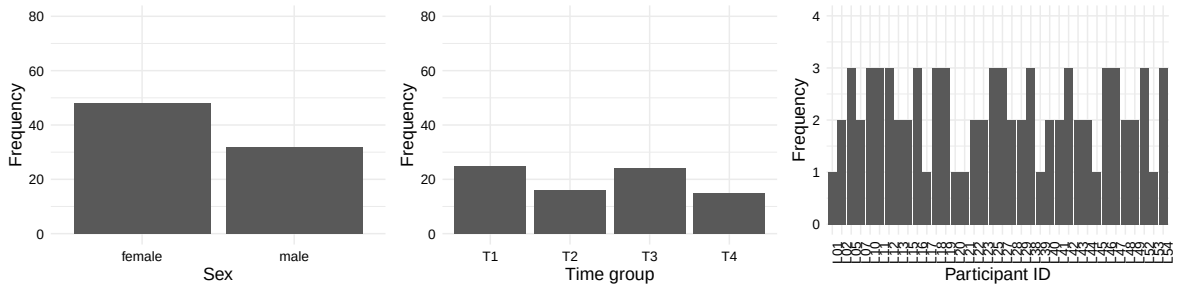
When assessing the distribution of categorical variables, we can use a frequency table or bar plot. A **frequency table** is a useful method to display the frequency and proportion of each level in a categorical variable in a clear and concise manner. In Table 3.8 we see the frequency table for the variable `sex`.

Table 3.8: Frequency table for the variable `sex`.

sex	frequency	proportion
female	48	0.6
male	32	0.4

A **bar plot** is a type of plot where the x-axis is a categorical variable and the y-axis is the frequency of the values. The frequency is represented by the height of the bar. The variables can be ordered by frequency, alphabetically, or some other order. Figure 3.1 is a bar chart for the variables `sex`, `group`, and `part_id`, ordered alphabetically.

So for a frequency table or barplot, we can see the frequency of each level of a categorical variable. This gives us some knowledge about the BELC dataset: there are more girls in the dataset, more essays appear in first and third time groups, and the number of essays written by each participant is scattered from one to three. If we were to see any clearly loopsided categories, this would be a sign of imbalance in the data and we would need to consider how this might impact our analysis.



(a) Sex (b) Time group (c) Participant ID

Figure 3.1: Bar plots for categorical variables `sex`, `group`, `part_id` in the BELC dataset.

Consider this
The goal of descriptive statistics is to summarize the data in a way that is meaningful and interpretable. With this in mind, compare the frequency table in 3.8 and bar plot in 3.1a. Does one provide a more interpretable summary of the data? Why or why not? Are there any other ways you might communicate this distribution more effectively?

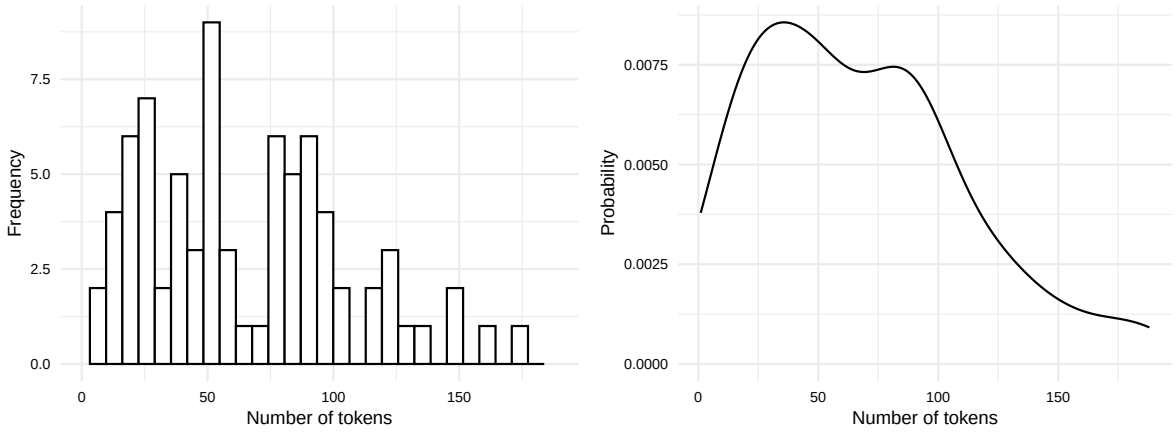
For numeric variables, understanding the distribution is more complex, and also more important. In essence, however, we are assessing two things: the appearance of outliers in relation to and the overall shape of the distribution.

Now, a frequency table, as in Table 3.8, does not summarize the distribution of a numeric variable in a concise, readily human-consumable format. Instead, the distribution of a numeric variable is best understood visually.

The most common visualizations of the distribution of a numeric variable are histograms and density plots. **Histograms** are a type of bar plot where the x-axis is a numeric variable and the y-axis is the frequency of the values falling within a determined range of values, or bins. The frequency of values within each bin is represented by the height of the bars. **Density plots** are a smoothed version of histograms. The y-axis of a density plot is the probability of the values. When frequent values appear closely together, the plot line is higher. When the frequency of values is lower or more spread out, the plot line is lower. An example of these plots is show in Figure 3.2 for the variable `tokens`.

Both the histogram in Figure 3.2a and the density plot in Figure 3.2b show the distribution of the variable `tokens` in slightly different ways which translate into trade-offs in terms of interpretability.

The histogram shows the frequency of the values in bins. The number of bins and/ or binwidth can be changed for more or less granularity. A rough grain histogram shows the general shape of the distribution, but it is difficult to see the details of the distribution. A fine grain histogram shows the details of the distribution, but it is difficult to see the general shape of

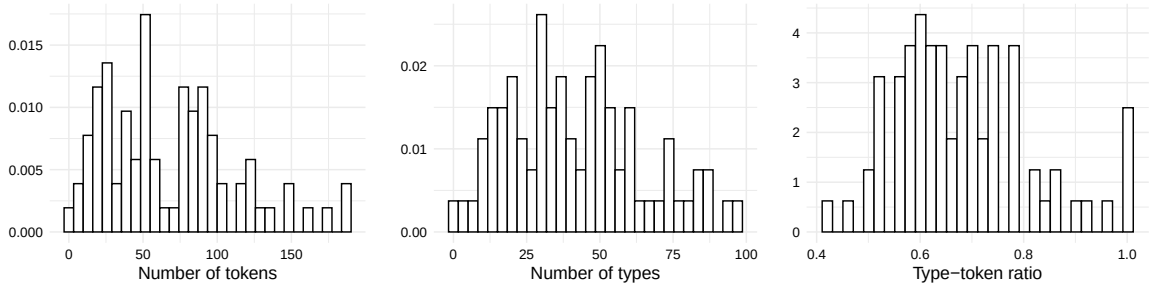


(a) Histogram (b) Density plot

Figure 3.2: Distribution plots for the variable `tokens`.

the distribution. The density plot shows the general shape of the distribution, but it hides the details of the distribution. Given this trade-off, it is often useful explore outliers with histograms and the overall shape of the distribution with density plots.

In Figure 3.3 we see histograms for the variables `tokens`, `types`, and `ttr`.



(a) Number of tokens (b) Number of types (c) Type-token ratio score

Figure 3.3: Histograms for numeric variables `tokens`, `types`, and `ttr`.

Focusing on the details captured in the histogram we are better able to detect potential outliers. Outliers can reflect valid values that are simply extreme or they can reflect something erroneous in the data. To distinguish between these two possibilities, it is important to know the context of the data. Take, for example, Figure 3.3c. We see that there is a bin near the value 1.0. Given that the type-token ratio is a ratio of the number of types to the number of tokens, it is unlikely that the type-token ratio would be exactly 1.0 as this would mean that every word in an essay is unique. Another, less dramatic, example is the bin to the far right of Figure 3.3a. In this case, the bin represents the number of tokens in an essay. An uptick in the number of

essays with a large number of tokens is not surprising and would not typically be considered an outlier. On the other hand, consider the bin near the value 0 in the same plot. It is unlikely that a true essay would have 0, or near 0, words and therefore a closer look at the data is warranted.

It is important to recognize that outliers contribute undue influence to overall measures of central tendency and dispersion. To appreciate this, let's consider another helpful visualization called a **boxplot**. A boxplot is a visual representation which aims to represent the central tendency, dispersion, and distribution of a numeric variable in one plot.

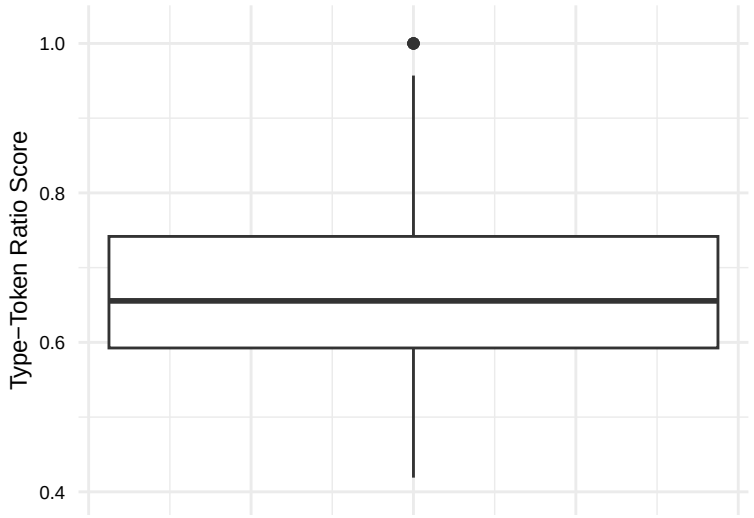
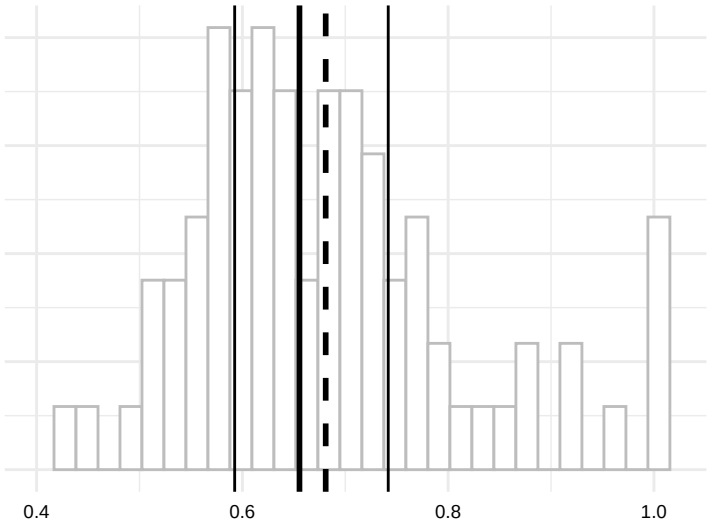


Figure 3.4: Boxplot for the variable `ttr`.

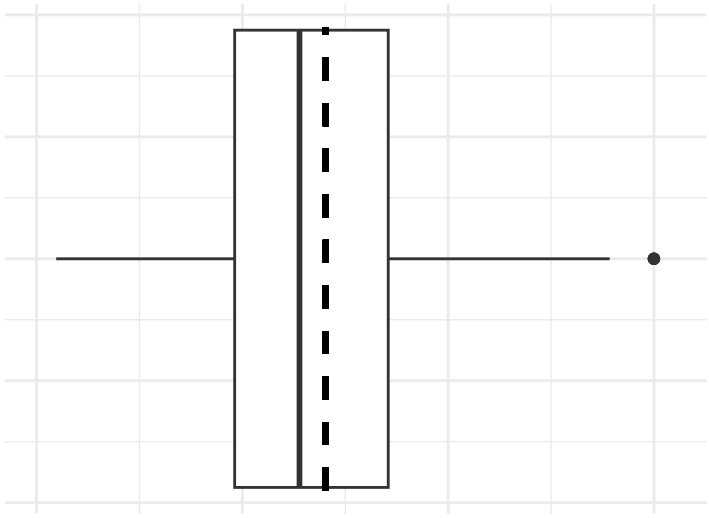
In Figure 3.4 we see a boxplot for `ttr` variable. The box in the middle of the plot represents the interquartile range (IQR) which is the range of values between the first quartile and the third quartile. The solid line in the middle of the box represents the median. The lines extending from the box are called 'whiskers' and provide the range of values which are within 1.5 times the IQR. Values outside of this range are plotted as individual points.

Now let's consider boxplots from another angle. In Figure 3.5b I've plotted the boxplot horizontally, right below the histogram in Figure 3.5a. In this view, we can see that a boxplot is a simplifed histogram augmented with central tendency and dispersion statistics. While histograms focus on the frequency distribution of data points, boxplots focus on the data's quartiles and potential outliers.

I've added a dashed line in Figure 3.5a and Figure 3.5b to signal the mean in this set of plots, but it is not typically included. I include the dashed line to make a point: the mean is more sensitive to outliers than the median. As I pointed out in Section 3.1.2, the mean is the sum



(a) Histogram



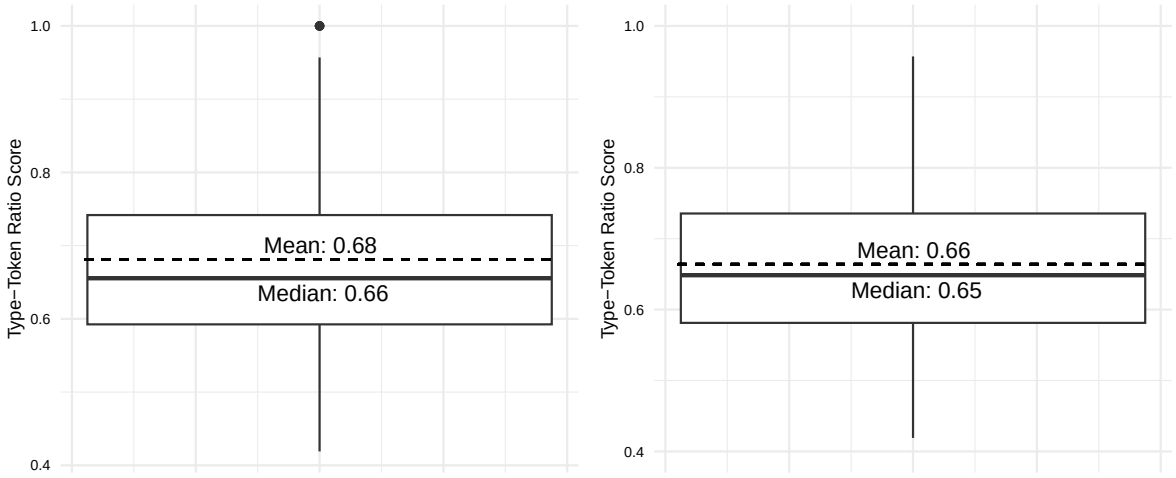
(b) Boxplot (horizontal)

Figure 3.5: Histogram and boxplot for the variable `ttr`.

of all values divided by the number of values. If there are extreme values, the mean will be pulled in the direction of the extreme values. The median, however, is the middle value and a few extreme values have less effect. So, when central tendency is reported, if there is a sizeable difference between the mean and the median, measures of dispersion will be larger and the direction of the difference can be used to infer the presence of outliers.

Returning to outliers, it is important to address them to safeguard the accuracy of the analysis. There are two main ways to address outliers: 1) transform the data and 2) eliminate observations with outliers (**trimming**). Trimming is more extreme as it removes data but can be the best approach for true outliers. Transforming the data is an approach to mitigating the influence of extreme but valid values. **Transformation** involves applying a mathematical function to the data which changes the scale and/ or shape of the distribution, but does not remove data nor does it change the relative order of the values.

In Figure 3.8, we see two boxplots. Figure 3.6a is the original `ttr` data and Figure 3.6b reflects the data trimmed to remove outliers. In this case, we have removed essays with a type-token ratio of 1.



(a) Type-token ratio score

(b) Type-token ratio score (trimmed)

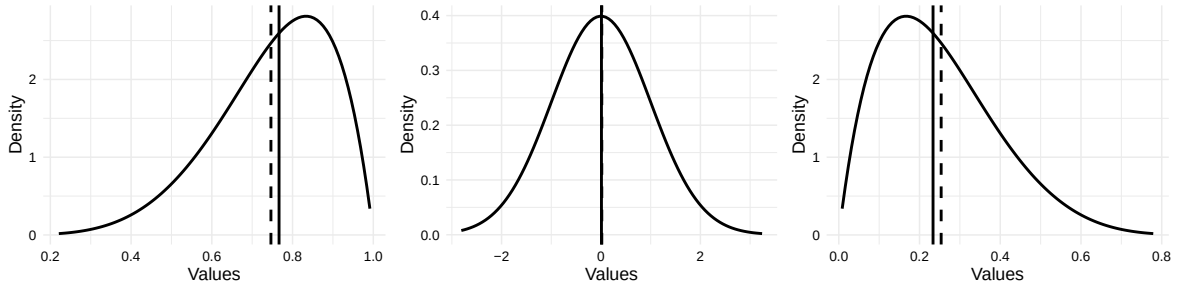
Figure 3.6: Boxplots for `ttr` before and after trimming.

We can now appreciate the relatively larger effect that the outliers had on the mean value of the `ttr` variable. As outliers are removed as the difference between the mean and median will become smaller.

The exploration the data points with histograms and boxplots has helped us to identify outliers. Now we turn to the question of the overall shape of the distribution. The key question is whether the observed distribution of each variable approximates the Normal Distribution, or not.

The **Normal Distribution** is a theoretical distribution where the values are symmetrically dispersed around the central tendency (mean/ median). In terms we can now understand, this means that the mean and median are the same. The Normal Distribution is important because many statistical tests assume that the data distribution is normal or near normal.

Stepping away from our BELC dataset, I've created simulated data that fit normal and non-normal, or skewed, distributions. I present each of these distributions as density plots with mean and median line overlays in Figure 3.7.



(a) Left skewed distribution (b) Normal distribution (c) Right skewed distribution

Figure 3.7: Mean and median for normal and skewed distributions.

A **Normal Distribution**, illustrated in Figure 3.7b, is a distribution where the values are symmetrically dispersed around the central tendency (mean/ median). This means that in a theoretical distribution that the mean and median are the same. The Normal Distribution is also known as the **Gaussian Distribution** or the **Bell Curve**, for the hallmark bell shape of the distribution. In this distribution, extreme values are less likely than values near the center.

A **skewed distribution** is not a specific type of distribution but rather a characteristic than many distributions can exhibit where the values are not symmetrically dispersed around the central tendency. A distribution in which values tend to disperse to the left of the central tendency is **left skewed** as in Figure 3.7a and dispersion to the right is **right skewed** as in Figure 3.7c.

Data that are normally, or near-normally distributed are often analyzed using parametric tests while data that exhibit a skewed distributed are often analyzed using non-parametric tests. Divergence from normality is not a binary distinction. Rather, it is a matter of degree. A visual inspection is usually sufficient for experienced researchers to determine whether a distribution is normal or skewed. However, for those who are less experienced or if you want to be more precise, there are two primary measures which can help ascertain the degree to which a distribution is normal: skewness and kurtosis. **Skewness** is a measure of the degree to which a distribution is asymmetrical. **Kurtosis** is a measure of the degree to which a distribution is peaked.

Table 3.10: Rules of thumb for skewness and kurtosis scores.

(a) Skewness scores

Score Range	Evaluation
-0.5 to 0.5	Approximately symmetric
-1 to -0.5 or 0.5 to 1	Moderately skewed
$<$ -1 or $>$ 1	Highly skewed

(b) Kurtosis scores

Score Range	Evaluation
$<$ 3	Less peaked than normal
Equal to 3	Normal peak
$>$ 3	More peaked than normal

In Table 3.9 I provide the skewness and kurtosis scores for our simulated distributions along with central tendency measures for context.

Table 3.9: Skewness and kurtosis for normal and skewed distributions.

distribution	mean	median	histogram	skewness	kurtosis
Left skew	0.746	0.767		-0.711	3.27
Normal	0.016	0.009		0.065	2.93
Right skew	0.254	0.233		0.711	3.27

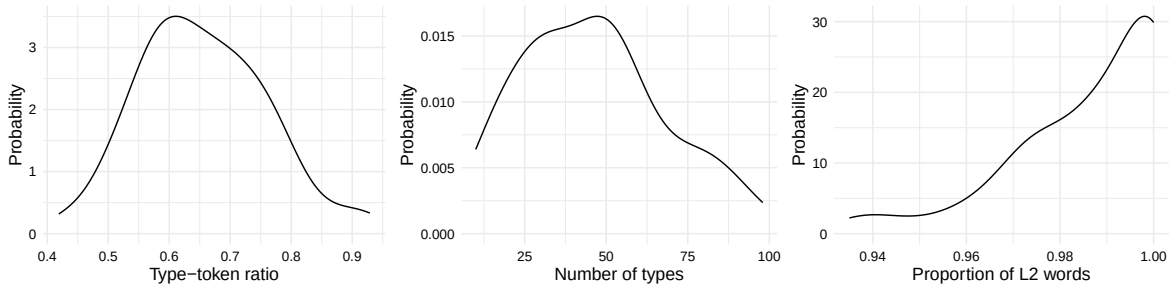
All things distribution are matters of degree, so there are no hard and fast rules for determining whether a distribution is normal or skewed. However, there are some general guidelines that can be used to determine the degree to which a distribution is normal or skewed, as shown in Table 3.9.

Dive deeper
Another approach for visually summarizing a single numeric variable is the Empirical Cumulative Distribution Function, or *ECDF*. An ECDF plot is a summary of the cummulative proportion of each of the values of a numeric variable. In addition to providing insight into the distribution of a variable, ECDF plots can be useful in determing what proportion of the values fall above or below a certain percentage of the data.

The question is which type of distribution does each numeric variable in the BELC dataset fit? Comparing the variables `ttr`, `types` and `prop_l2` in Figure 3.8 to the three distributions in Figure 3.7, we see that all three numeric variables in the BELC dataset are skewed to some degree.

Figure 3.8a for `ttr` has some right skewing but not as much as `types` in Figure 3.8b. `prop_l2` in Figure 3.8c is the most skewed of the three variables. As mentioned earlier, skewed distributions can take many forms, some are more skewed than others.

To view statistics on our three variables in Figure 3.8, we can calculate the skewness and kurtosis.



(a) Type-token ratio score (b) Number of types (c) Proportion of L2 words

Figure 3.8: Histogram/ Density plots for numeric variables in the BELC dataset.

Table 3.11: Skewness and kurtosis for numeric variables in the BELC dataset.

distribution	mean	median	histogram	skewness	kurtosis
ttr	0.655	0.648		0.319	2.90
types	46.044	46.000		0.407	2.45
tokens	75.338	77.000		0.669	2.98
prop_l2	0.986	0.990		-1.273	4.13

Given the characteristics of the numeric variables in the BELC dataset, although none of them are perfectly normal, but only `prop_l2` is highly skewed. Therefore, if we intend to use these variables 'as-is' in statistical measures or tests, we now know whether to choose parametric or non-parametric alternatives.

In the case that a variable is highly skewed, it is often useful to attempt transform the variable to reduce the skewness. In contrast to scale-based transformations (*e.g.* centering and scaling), shape-based transformations change the scale and the shape of the distribution. The most common shape-based transformation is the logarithmic transformation. The **logarithmic transformation** (log-transformation) takes the log (typically base 10) of each value in a variable. The log-transformation is useful for reducing the skewness of a variable as it compresses large values and expands small values. If the skewness is due to these factors, the log-transformation can help.

It is important to note, however, that if scale-based transformations are to be applied to a variable, they should be applied after the log-transformation as the log of negative values is undefined.

Interdependence

We have covered the first three of the four questions we are interested in asking in a descriptive analysis. The fourth, and last, question is whether there is mutual dependence between variables. If so, what is the directionality and how strong is the dependence? Knowing the answers to these questions will help frame our approach to analysis.

To assess interdependence, the number and information types of the variables under consideration are important. Let's start by considering two variables. If we are working with two variables, we are dealing with a **bivariate** relationship. Given there are three informational types (categorical, ordinal, and numeric), there are six logical bivariate combinations: categorical-categorical, categorical-ordinal, categorical-numeric, ordinal-ordinal, ordinal-numeric, and numeric-numeric.

The directionality of a relationship will take the form of a tabular or graphic summary depending on the informational value of the variables involved. In Table 3.12, we see the appropriate summary types for each of the six bivariate combinations.

Table 3.12: Appropriate summary types for different combinations of variable types.

	Categorical	Ordinal	Numeric
Categorical	Contingency table	Contingency table/ Bar plot	Pivot table/ Boxplot
Ordinal	-	Contingency table/ Bar plot	Pivot table/ Boxplot
Numeric	-	-	Scatterplot

Let's first start with the combinations that include a categorical or ordinal variable. Categorical and ordinal variables reflect measures of class-type information, with add meaningful ranks to ordinal variables. To assess a relationship with these variable types, a table is always a good place to start. When combined together, a contingency table is the appropriate table. A **contingency table** is a cross-tabulation of two class-type variables, basically a two-way frequency table. This means that three of the six bivariate combinations are assessed with a contingency table: categorical-categorical, categorical-ordinal, and ordinal-ordinal.

In Table 3.13 we see contingency tables for the categorical variable `sex` and ordinal variable `group` in the BELC dataset.

A contingency table may include only counts, as in Table 3.13a, or may include proportions or percentages in an effort to normalize the counts and make them more comparable, as in Table 3.13b.

It is sometimes helpful to visualize a contingency table as a bar plot when there are a larger number of levels in either or both of the variables. Again, looking at the relationship between

Table 3.13: Contingency tables for categorical variable `sex` and ordinal variable `group` in the BELC dataset.

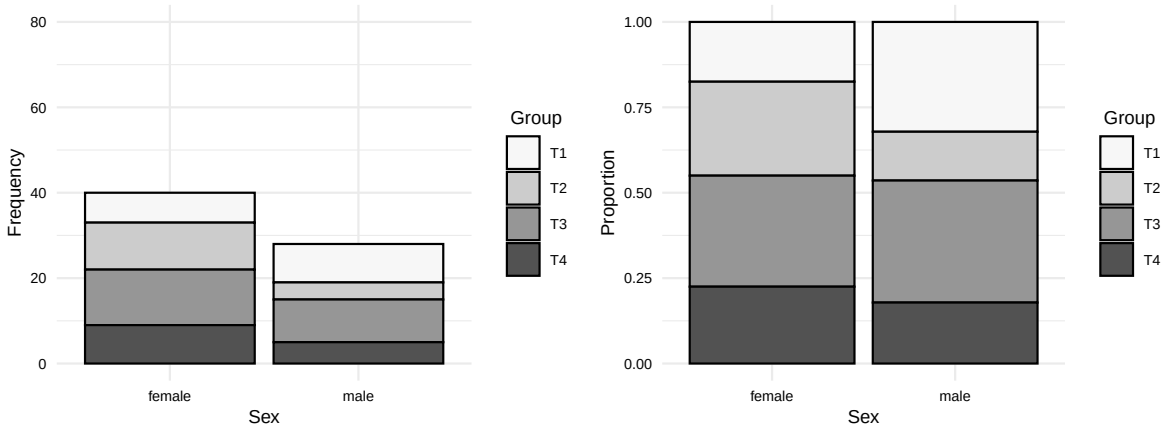
(a) Counts

group	female	male	Total
T1	7	9	16
T2	11	4	15
T3	13	10	23
T4	9	5	14
Total	40	28	68

(b) Percentages

group	female	male	Total
T1	43.75%	56.25%	100.00%
T2	73.33%	26.67%	100.00%
T3	56.52%	43.48%	100.00%
T4	64.29%	35.71%	100.00%
Total	58.82%	41.18%	100.00%

`sex` and `group`, we see that we can plot the counts or the proportions. In Figure 3.9, we see both.



(a) Counts

(b) Proportions

Figure 3.9: Bar plots for the relationship between `sex` and `group` in the BELC dataset.

To summarize and assess the relationship between a categorical or an ordinal variable and a numeric variable, we cannot use a contingency table. Instead, this type of relationship is best summarized in a table using a summary statistic in a **pivot table**. A pivot table is a table in which a class-type variable is used to group a numeric variable by some summary statistic appropriate for numeric variables, *e.g.* mean, median, standard deviation, *etc.*

In Table 3.14, we see a pivot table for the relationship between `group` and `tokens` in the BELC dataset. Specifically, we see the mean number of tokens by group.

Table 3.14: Pivot table for the relationship between `group` and `tokens` in the BELC dataset.

group	mean_tokens
T1	35.4
T2	62.5
T3	85.0
T4	118.9

We see that the mean number of tokens increases from Group T1 to T4, which is consistent with the idea that the students in the higher groups are writing longer essays.

Although a pivot table may be appropriate for targeted numeric summaries, a visualization is often more informative for assessing the dispersion and distribution of a numeric variable by a categorical or ordinal variable. There are two main types of visualizations for this type of relationship: a boxplot and a **violin plot**. A violin plot is a visualization that summarizes the distribution of a numeric variable by a categorical or ordinal variable, adding the overall shape of the distribution, much as a density plot does for histograms.

In Figure 3.10, we see both a boxplot and a violin plot for the relationship between `group` and `tokens` in the BELC dataset.

(a) Boxplot

(b) Violin plot

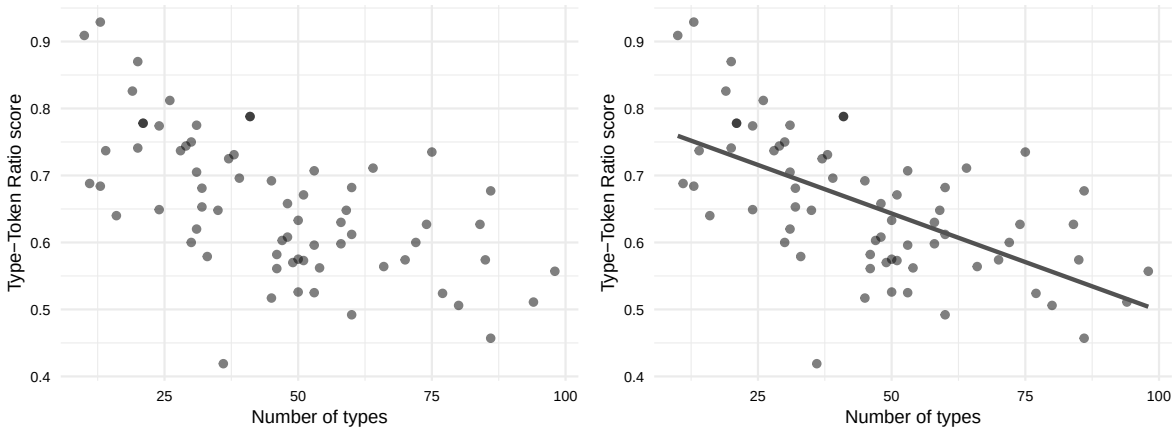
Figure 3.10: Boxplot and violin plot for the relationship between `group` and `tokens` in the BELC dataset.

From the boxplot in Figure 3.10a, we see that the general trend towards more tokens used by students in higher groups. But we can also appreciate the dispersion of the data within each group looking at the boxes and whiskers. On the surface it appears that the data for groups T1 and T3 are closer to each other than groups T2 and T4, in which there is more variability

within these groups. Furthermore, we can see outliers in groups T1 and T3, but not in groups T2 and T4. From the violin plot in Figure 3.10b, we can see the same information, but we can also see the overall shape of the distribution of tokens within each group. In this plot, it is very clear that group T4 includes a wide range of token counts.

The last bivariate combination is numeric-numeric. To summarize this type of relationship a scatterplot is used. A **scatterplot** is a visualization that plots each data point as a point in a two-dimensional space, with one numeric variable on the x-axis and the other numeric variable on the y-axis. Depending on the type of relationship you are trying to assess, you may want to add a trend line to the scatterplot. A trend line is a line that summarizes the overall trend in the relationship between the two numeric variables. To assess the extent to which the relationship is linear, a straight line is drawn which minimizes the distance between the line and the points.

In Figure 3.11, we see a scatterplot and a scatterplot with a trend line for the relationship between `ttr` and `types` in the BELC dataset.



(a) Points

(b) Points with a linear trend line

Figure 3.11: Scatter plot for the relationship between `ttr` and `types` in the BELC dataset.

We see that there is an apparent positive relationship between these two variables, which is consistent with the idea that as the number of types increases, the type-token ratio increases. In other words, as the number of unique words increases, so does the lexical diversity of the text. Since we are evaluating a linear relationship, we are assessing the extent to which there is a **correlation** between `ttr` and `types`. A correlation simply means that as the values of one variable change, the values of the other variable change in a consistent manner.

Once a sense of the directionality of a relationship can be established, the next step is to gauge the relative strength, or association. **Association** refers to any relationship in which there is a dependency between two variables. Quantitative measures of association, in combination

with tabular and visual summaries, can provide a more complete picture of the relationship between two variables.

There are a number of measures of assocation, depending on the types of variables being assessed and, for numeric variables, whether the distribution is normal (parametric) or non-normal (non-parametric), as seen in Table 3.15.

Table 3.15: Measures of association or correlation strength for different combinations of variable types.

	Categorical	Ordinal	Numeric*Non-parametric*	Numeric*Parametric*
Categorical	Chi-square χ^2, Cramér's V	Goodman and Kruskal's γ	Rank biserial Correlation	Point-biseral Correlation
Ordinal	-	Kendall's τ	Kendall's τ	Pearson's r
Numeric*Non-parametric*	-	-	Kendall's τ	Pearson's r
Numeric*Parametric*	-	-	-	Pearson's r

Association measures often are expressed as a number between -1 and 1, where 0 indicates no association, -1 indicates a perfect negative association, and 1 indicates a perfect positive association. The closer the number is to 0, the weaker the association. The closer the number is to -1 or 1, the stronger the association. Association statistics are often accompanied by a **confidence interval** (CI), which is a range of values that is likely to contain the true value of the association in the population. The confidence interval is expressed as a percentage, such as 95%, which means that if we were to repeat the study 100 times, 95 of those studies would produce a confidence interval that contains the true value of the association in the population. If the range between the lower and higher bounds of the confidence interval contains 0, then the association is likely no different than chance.

Given these measures and interpretations, let's consider the different types of bivariate relationships we have seen so far in the BELC dataset. The first interdependence we explored involved the categorical variable `sex` and the ordinal variable `group`. This relationship may not be of primary interest to a study on L2 writing, but it is a good example of how to assess the strength of an association between a categorical and ordinal variable. Furthermore, it could be the case that we want to assess whether we have widely unbalanced female/ male proportions in our time groups.

Using Table 3.15, we see that we can use Goodman and Kruskal's γ (gamma) to assess the strength of the association between these two variables. The measures of association in Table 3.16 suggest that the proportion of male participants is higher in group T1 and lower in group T2. However, these associations are moderately strong, as the gamma value is near $\pm$ 0.4.

Table 3.16: Gamma for the relationship between `sex` and `group` in the BELC dataset.

Parameter1	Parameter2	r	CI	CI_low	CI_high
sex.female	group.T1	-0.381	0.95	-0.568	-0.157
sex.female	group.T2	0.389	0.95	0.167	0.575
sex.male	group.T1	0.381	0.95	0.157	0.568
sex.male	group.T2	-0.389	0.95	-0.575	-0.167

When paired with Figure 3.9 we can appreciate that groups T1 and T2 have contrasting proportions of females to males and that groups T3 and T4 are more closely proportioned. This observation should be considered when approaching statistical analyses in which categorical variables required (near) equal proportions of categories.

Now let's take a look at a more interesting relationship, the one between the ordinal variable `group` and the numeric variable `tokens`. Since we determined that `tokens` was near normally distributed, we can choose the parametric version of our association measure, Pearson's r. The measures of association in Table 3.17 suggest that there is a negative association between group T1 and a positive one betwen group T4 and `tokens`, which is consistent with the idea that as the group number increases, the number of tokens increases. These associations are moderate to strong, as the Pearson's r values are near $\pm$ 0.5. However, the other groups (T2 and T3) have very weak assocations with `tokens` and the CI includes 0, which means that the association is likely no different than chance.

Table 3.17: Pearson's r for the relationship between `group` and `tokens` in the BELC dataset.

Parameter1	Parameter2	r	CI	CI_low	CI_high
group.T1	tokens	-0.520	0.95	-0.675	-0.322
group.T2	tokens	-0.160	0.95	-0.384	0.082
group.T3	tokens	0.161	0.95	-0.080	0.385
group.T4	tokens	0.521	0.95	0.322	0.675

These association measures suggest that there is a relationship between `group` and `tokens`, but that the relationship is not the same for all groups. This may due to a number of factors, such as the number of participants in each group, the effect of outliers within particular levels, etc. or may simply underscore that the relationship between `group` and `tokens` is not linear. What we do with this information will depend on our research aims. Whatever the case, we can use these measures to inform our next steps, as we will see in the next section.

Finally, let's look at the relationship between the numeric variables `ttr` and `types`. Since we determined both `ttr` and `types` are normally distributed, we can choose the parametric version of our association measure, Pearson's r. The measures of association in Table 3.18 suggest that

there is a negative association between `ttr` and `types`, which is consistent with the idea that as the number of types increases, the type-token ratio decreases. This association is strong, as Pearson's r value is near 0.6.

Table 3.18: Pearson's r for the relationship between `ttr` and `types` in the BELC dataset.

Parameter1	Parameter2	r	CI	CI_low	CI_high
ttr	types	-0.606	0.95	-0.738	-0.43

Before moving on to the next section, it is important to remember than through the process of diagnostic measures, we gain a thorough understanding of our data's characteristics and quality, preparing us for the next step in our analysis. However, remember that these measures do not exist in isolation. The decisions we make at this stage, from handling missing data to understanding the distribution of our variables, can have significant implications on our subsequent analysis. So, this initial step of data analysis deserves our careful attention and scrutiny.

3.2 Analyze

Having ensured that our dataset is clean, valid, and thoroughly understood, we can proceed to the next key stage of our data analysis process - employing analytic methods. The goal of analysis, generally, is to generate knowledge from information. The type of knowledge generated and the process by which it is generated, however, differ and can be broadly grouped into three analysis types: exploratory, predictive, and inferential.

In this section I will provide an overview of how each of these analysis types are tied to research aims and how the general purpose of each type affect: (1) how to *identify* the variables of interest, (2) how to *interrogate* these variables, and (3) how to *interpret* the results. I will structure the discussion of these analysis types moving from the least structured (inductive) to most structured (deductive) approach to deriving knowledge from information with the aim to provide enough information for you to identify these research approaches in the literature and to make appropriate decisions as to which approach your research should adopt.

3.2.1 Explore

In **Exploratory Data Analysis (EDA)**, we use a variety of methods to identify patterns, trends, and relations within and between variables. The goal of EDA is uncover insights in an inductive, data-driven manner. That is to say, that we do not enter into EDA with a fixed hypothesis in mind, but rather we explore intuition, probe anecdote, and follow hunches to identify patterns and relationships and to evaluate whether and why they are meaningful. We

are admittedly treading new or unfamiliar terrain letting the data guide our analysis. This means that we can use and reuse the same data to explore different angles and approaches adjusting our methods and measures as we go. In this way, EDA is an iterative, meaning generating process.

In line with the investigative nature of EDA, the identification of variables of interest is a discovery process. We most likely have a intution about the variables we would like to explore, but we are able to adjust our variables as need be to suit our research aims. When the identification and selection of variables is open, the process is known as **feature engineering**. A process that is much an art as a science, feature engineering leverages a mixture of relevant domain knowledge, intuition, and trial and error to identify features that serve to best represent the data and to best serve the research aims. Furthermore, the roles of features in EDA are fluid –no variable has a special status, as seen in Figure 3.12. We will see that in other types of analysis, some or all the roles of the variables are fixed.

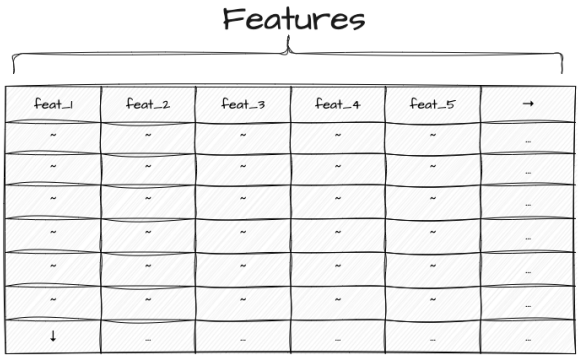


Figure 3.12: Roles of variables in EDA.

For illustrative purposes let's consider the State of the Union Corpus (SOTU) (Benoit 2020). The presidential addresses and a set of metadata variables are included in the corpus. I've subsetted this corpus to only include U.S. presidents since 1946. A tabular preview of the first 10 addresses (truncated for display) can be found in Table 3.19.

Table 3.19: First ten addresses from the SOTU Corpus.

president	date	delivery	party	addresses
Truman	1946-01-21	written	Democratic	To the Congress of the United States: A quarter…
Truman	1947-01-06	spoken	Democratic	Mr. President, Mr. Speaker, Members of the Cong…
Truman	1948-01-07	spoken	Democratic	Mr. President, Mr. Speaker, and Members of the …

president	date	delivery	party	addresses
Truman	1949-01-05	spoken	Democratic	Mr. President, Mr. Speaker, Members of the Cong…
Truman	1950-01-04	spoken	Democratic	Mr. President, Mr. Speaker, Members of the Cong…
Truman	1951-01-08	spoken	Democratic	Mr. President, Mr. Speaker, Members of the Cong…
Truman	1952-01-09	spoken	Democratic	Mr. President, Mr. Speaker, Members of the Cong…
Truman	1953-01-07	written	Democratic	To the Congress of the United States: I have th…
Eisenhower	1953-02-02	spoken	Republican	Mr. President, Mr. Speaker, Members of the Eigh…
Eisenhower	1954-01-07	spoken	Republican	Mr. President, Mr. Speaker, Members of the Eigh…

A dataset such as this one could serve as a starting point to explore many different types of research questions. In order to maintain research coherence so our efforts to not careen into a free-for-all, we need to tether our feature engineering to a unit of analysis that is relevant to the research question. A **unit of analysis** is the entity that we are interested in studying. Not to be confused with the unit of observation, which is the entity that we are able to observe and measure.

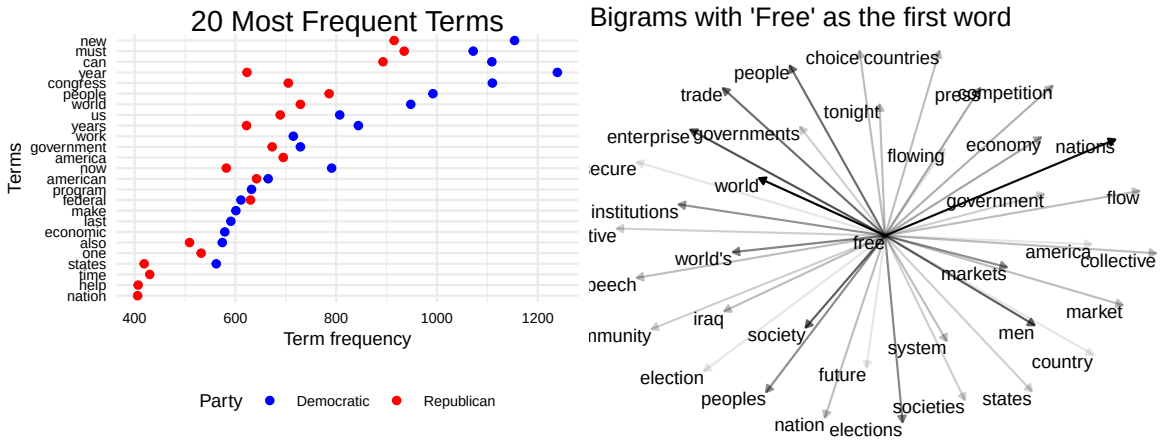
To demonstrate the distinction, let's look consider different approaches to analyzing the SOTU dataset. For example, the unit of analysis could be the language of particular presidents, party ideology, or political rhetoric in general and the unit of observation could be individual words, phrases, sentences, etc. In some cases the unit of analysis and the unit of observation are the same. For example, if we were interested in potential changes use of the word "terrorist" over time in SOTU addresses, the unit of analysis and the unit of observation would be the same –individual addresses. So, depending on the perspective we are interested in investigating, the choice of how to approach engineering features to gain insight will vary.

By the same token, approaches for interrogating the dataset can differ significantly, between research projects and within the same project, but for instructive purposes, let's draw a distinction between descriptive methods and unsupervised learning methods, as seen in Table 3.20.

Table 3.20: Some common EDA methods

Descriptive methods	Unsupervised learning methods
Frequency analysis	Cluster analysis
Keyness analysis	Topic Modeling
Co-occurence analysis	Vector Space Models

The first group, **descriptive methods** can be seen as a (more robust) extenstion of the descriptive statistics covered earlier in this chapter including statistic, tabular, and visual techniques. For example, a frequency analysis of the SOTU dataset could be used to identify the most common words used by U.S. political parties in their addresses, in Figure 3.13a, or a co-occurence analysis could be used to identify the most common words the appear after the term "free", in Figure 3.13b, in the dataset.



(a) Frequency analysis of the 20 most frequent terms by party.

(b) Co-occurence analysis of the terms that appear after the term 'free'.

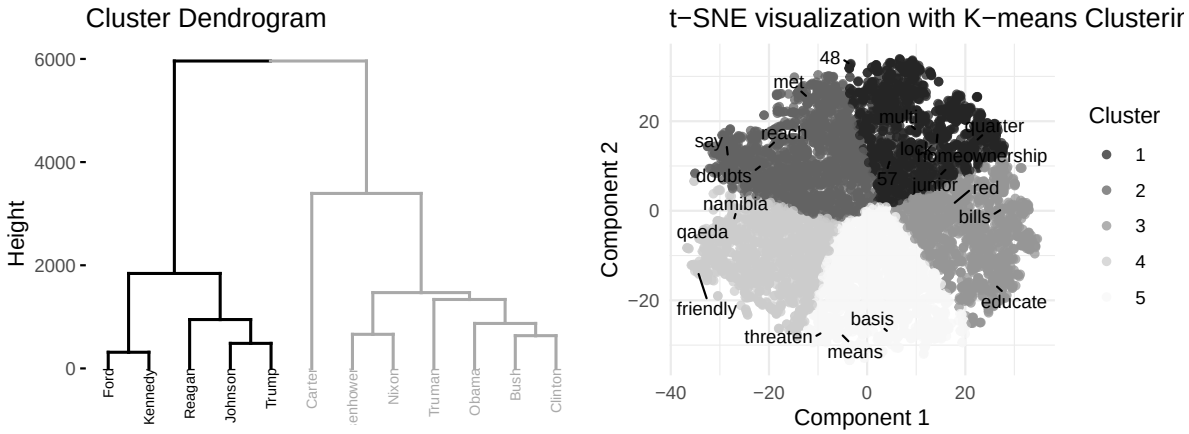
Figure 3.13: Example of descriptive methods applied to the SOTU dataset.

The second group, **unsupervised learning** is a subtype of machine learning in which an algorithm is used to find patterns within and between variables in the data without any guidance (supervision). In this way, the algorithm, or machine learner, is left to make connections and associations wherever they may appear in the input data. If we were interested in finding word-use continuities and discontinuities between presidents, we could use a clustering algorithm, seen in Figure 3.14a. Or if we wanted to uncover themes ... 🔧 [ADD: modify plot] we could use a vector space model, as in Figure 3.14b.

Either through descriptive, unsupervised learning methods, or a combination of both, EDA employs quantitative methods to summarize, reduce, and sort complex datasets in order to provide the researcher novel perspective to be qualitatively assessed. Exploratory methods produce results that require associative thinking and pattern detection. Speculative as they are, the results from exploratory methods can be highly informative and lead to new insight and inspire further study in directions that may not have been expected.

3.2.2 Predict

Predictive Data Analysis (PDA) employs a variety of techniques to examine and evaluate the association strength between a variable or set of variables, with a specific focus on pre-



(a) Hierarchical clustering of the SOTU corpus. (b) Word embedding space in the SOTU corpus.

Figure 3.14: Example of unsupervised learning methods applied to the SOTU dataset.

dicting a target variable. The aim of PDA is to construct models that can accurately forecast future outcomes, using either data-driven or theory-driven approaches. In this process, **supervised learning** methods, where the machine learning algorithm is guided (supervised) by a target outcome variable, are used. This means we don't begin PDA with a completely open-ended exploration, but rather with an objective - accurate predictions. However, the path to achieving this objective can be flexible, allowing us freedom to adjust our models and methods. Unlike EDA, where the entire dataset can be reused for different approaches, PDA requires a portion of the data to be reserved for evaluation, enhancing the validity of our predictive models. Thus, PDA is an iterative process that combines the flexibility of exploratory analysis with the rigor of confirmatory analysis.

There are two types of variables in PDA: the outcome variable and the predictor variables, or features. The **outcome variable** is the variable that the researcher is trying to predict. It is the only variable that is necessarily fixed as part of the research question. The features are the variables that are used to predict the outcome variable. An overview of the roles of these variables in PDA is shown in Figure 3.15.

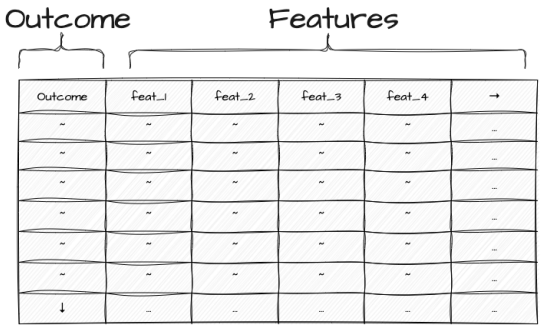


Figure 3.15: Roles of variables in PDA.

Feature selection can be either data-driven or theory-driven. Data-driven features are those that are engineered to enhance predictive power, while theory-driven features are those that are selected based on theoretical relevance.

Let's consider the Europarl corpus of native, non-native and translated texts (ENNTT) (Nisioi et al. 23-28, 2016-05). This is a monolingual English corpus of translated and non-translated texts from the European Parliament.

Table 3.21: Data dictionary of the ENNTT corpus.

variable	name	variable_type	description
session_id	Session ID	categorical	Unique identifier for each session
seq_speaker_id	Sequential Speaker ID	ordinal	Unique numeric identifier for each speaker
state	State	categorical	Country of the session speaker
language	Language	categorical	Original language in which the sentence was uttered
type	Type	categorical	Category of the speaker: natives, nonnatives, or translations
text	Text	categorical	Text spoken in the session

Now depending on our research question, we will have a different outcome variable. If we want to examine the potential linguistic differences between native and non-native speakers, we will select our outcome variable to be the `type` (natives/ nonnatives). The features selected to use to predict `type` depend on our research question. If our research is guided by data, we will choose features that are specifically designed to boost the ability to predict. On the other hand, if our research is steered by theory, we will opt for features that are chosen due to their theoretical significance. In either case, the original dataset will likely need to be transformed.

The approach to interrogating the dataset includes three main steps: feature engineering, model selection, and model evaluation. We've discussed feature engineering, so what is model selection and model evaluation? And how do we go about performing these steps?

Model selection is the process of choosing a machine learning algorithm and set of features that produces the best prediction accuracy for the outcome variable. To refine our approach such that we arrive at the best combination of algorithm and features, we need to train our machine learner on a variety of combinations and evaluate the accuracy of each. We don't want to train and evaluate on the same data, as this would be cheating, and likely would not produce a model that generalizes well to new data. Instead, we split our data into two sets: a training set and a test set. The **training set** is used to train the machine learner, while the **test set** is used to evaluate the accuracy of the model[1]. The larger portion of the data, from 60% to 80%, is used for training, while the remaining portion is used for testing.

The elephant in the room is, what type of machine learning algorithm do I use? Well, there are many different types of machine learning algorithms, each with their own strengths and weaknesses. The first rough cut is to decide what type of outcome variable we are predicting: categorical or numeric. If the outcome variable is categorical, we are performing a **classification** task, and if the outcome variable is numeric, we are performing a **regression** task. As we see in Table 3.22, there are various algorithms that can be used for each task.

Table 3.22: Some common supervised learning algorithms used in PDA.

Classification	Regression	Learner type
Logistic Regression	Linear Regression	Interpretable
Decision Tree	Regression Tree	Interpretable
Support Vector Machine	Support Vector Regression	Black box
Multilayer Perceptron	Multilayer Perceptron	Black box

I've included a column in Table 3.22 that charaterizes a second consideration which is whether we want an interpretable model or a black box model. When talking about whether a model is interpretable or not, we are not referring to the evaluation of the accuracy of the model. Rather, we are referring to the inner workings of the model itself that allow us to understand how the model is making its predictions. An **interpretable model** is one that can be understood and explored by humans, while a **black box model** is one whose inner workings are not trivially unraveled. The advantage of an interpretable model is that it researchers can go beyond evaluating prediction accuracy and probe feature-outcome associations. On the other hand, if the goal is to simply boost prediction accuracy, interpretability may not be a concern.

[1]Depending on the application and the amount of available data, a third *development set* is sometimes created as a pseudo test set to facilitate the testing of multiple approaches on data outside the training set before the final evaluation on the test set is performed.

Finally, there are a number of algorithm-specific strengths and weaknesses to be considered in the process of model selection. These hinge on characteristics of the data, such as the size of the dataset, the number of features, the type of features, and the expected type of relationships between features or on computing resources, such as the amount of time available to train the model or the amount of memory available to store the model.

Model evaluation is the process of assessing the accuracy of the model on the test set, which is a proxy for how well the model will generalize to new data. Model evaluation is performed quantitatively by calculating the accuracy of the model on the training, to develop the model, and ultimately, the test set. The accuracy of a model is calculated by comparing the predicted values to the actual values. For the results of classification tasks, this results in a contingency table, known as a confusion matrix. A **confusion matrix** juxtaposes predicted and actual values allowing various metrics to be calculated, for example in Table 3.23.

Table 3.23: Confusion matrix for the utterance type classification task.

	Predicted: natives	Predicted: nonnatives
Actual: natives	26294 (90% of 29215)	2921 (10% of 29215)
Actual: nonnatives	730 (10% of 7304)	6574 (90% of 7304)

Since regression tasks predict numeric values, the accuracy of the model is calculated by comparing the difference between the predicted and actual values.

It is important to note that whether the accuracy metrics are good is to some degree qualitative judgment. For example, classification accuracy overall may be relatively high, but the model may be performing poorly on one of the classes. In this case, the model may not be useful for the task at hand, despite the overall accuracy.

In the end, PDA offers a versitle path to discover data-driven insights, to probe theory-driven associations, or even simply to perform tasks that are too complex or time-consuming for humans to perform.

3.2.3 Infer

The most commonly recognized of the three data analysis approaches, **Inferential data analysis (IDA)** is the bread-and-butter of science. IDA is a deductive, theory-driven approach in which all aspects of analysis stem from a pre-determined premise, or hypothesis, about the nature of a relationship in the world and then aims to test whether this relationship is statistically supported given the evidence. Since the goal is to infer conclusions about a certain relationship in the population based on a statistical evaluation of a (corpus) sample, the representativeness of the sample is of utmost importance. Furthermore, the use of the data is limited to the scope of the hypothesis –that is, the data cannot be used for exploratory purposes.

The selection of variables and the roles they play in the analysis are determined by the hypothesis. In a nutshell, a **hypothesis** is a formal statement about the state of the world. This statement is theory-driven meaning that it is predicated on previous research. We are not exploring or examining relationships, rather we are testing a specific relationship. In practice, however, we are in fact proposing two mutally exclusive hypotheses. The first is the **Alternative Hypothesis**, or H_1. This is the hypothesis I just described –the statement grounded in the previous literature outlining a predicted relationship. The second is the **Null Hypothesis**, or H_0. This is the flip-side of the hypothesis testing coin and states that there is no difference or relationship. Together H_1 and H_0 cover all logical outcomes.

To connect hypotheses to variable selection and variable roles, let's consider a study in which a researcher is investigating the claim that men and women differ in terms of the number of questions they use in spontaneous conversations. The unit of analysis is individuals (i.e. men and women) and the unit of observation is (sponteanous) conversations.

A dataset based on the Switchboard Dialog Act Corpus (SWDA) (University of Colorado Boulder 2008), seen in Table 3.24, aligns well with this investigation. It is a large collection of transcribed telephone conversations between strangers. The dataset includes gender information for each participant and dialog act annotation for each utterance, including a range of question types.

Table 3.24: Data dictionary of the SWDA dataset.

variable	name	variable_type	description
doc_id	Document ID	numeric	The unique identifier for each document
speaker_id	Speaker ID	numeric	The unique identifier for each speaker
sex	Sex	categorical	The sex or gender of the speaker, either 'Male' or 'Female'
damsl_tag	DAMSL Tag	categorical	The Dialogue Act Markup in Several Layers (DAMSL) tag classification
utterance_text	Utterance Text	categorical	The transcription of what the speaker said in the document

The Alternative Hypothesis may be formulated in this way:

H_1: Men and women differ in the frequency of the use of questions in spontaneous conversations.

The Null Hypothesis, then, would be a statement describing the remaining logical outcomes. Specifically:

H_0: Men and women do *not* differ in the frequency of the use of questions in spontaneous conversations.

Now, in standard IDA one variable is the dependent variable and one or more variables are predictor variables. The **dependent variable**, sometimes referred to as the outcome or response variable, is the variable which contains the information which is hypothesized to depend on the information in the predictor variable(s). It is the variable whose variation a research study seeks to explain. A **predictor variable**, sometimes referred to as a independent or explanatory variable, is a variable whose variation is hypothesized to explain the variation in the dependent variable.

Returning to our hypothetical study and the hypotheses presented, we can identify the variables in our study and map them to their roles. The frequency of questions used by each speaker would be our dependent variable and the biological sex of the speakers our predictor variable. This is so because H_1 states the proposition that a speaker's sex will predict the frequency of questions used. The next step would be to operationalize what we mean by 'frequency of questions' and then transform the dataset to reflect this definition.

In our hypothetical study we've identified two variables, one dependent and one predictor. It is important keep in mind that there can be multiple predictor variables in cases where the dependent variable's variation is predicted to be related to multiple variables. This relationship would need to be explicitly part of the original hypothesis, however. Due to the increasing difficulty for interpretation, in practice, IDA studies rarely include more than two or three predictor variables in the same analysis.

Predictor variables add to the complexity of a study because they are part of our research focus, specifically our hypothesis. It is, however, common to include other variables which are not of central focus, but are commonly assumed to contribute to the explanation of the variation of the dependent variable. Let's assume that the background literature suggests that the age of speakers also plays a role in the number of questions that men and women use in spontaneous conversation. Let's also assume that the data we have collected includes information about the age of speakers. If we would like to factor out the potential influence of age on the use of questions and focus on the particular predictor variables we've defined in our hypothesis, we can include the age of speakers as a **control variable**. A control variable will be added to the statistical analysis and documented in our report but it will not be included in the hypothesis nor interpreted in our results.

We can now see in Figure 3.16 the variables roles assigned to variables in a hypothesis-driven study.

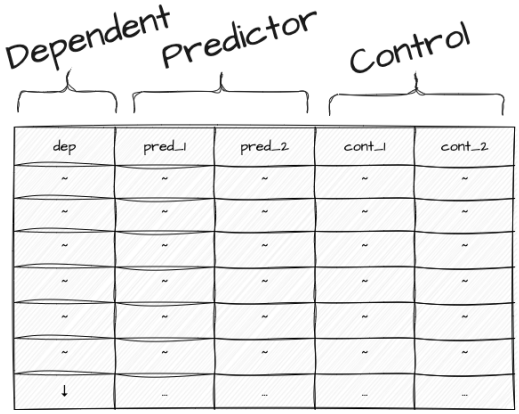


Figure 3.16: Roles of variables in IDA.

At this point let's look at the main characteristics that need to be taken into account to statistically interrogate the variables we have chosen to test our hypothesis. The type of statistical test that one chooses is based on (1) the informational value of the dependent variable and (2) the number of predictor variables included in the analysis. Together these two characteristics go a long way in determining the appropriate class of statistical test, but other considerations about the distribution of particular variables (i.e. normality), relationships between variables (i.e. independence), and expected directionality of the predicted effect may condition the appropriate method to be applied.

As you can imagine, there are a host of combinations and statistical tests that apply in particular scenarios, too many to consider in given the scope of this coursebook (see S. Th. Gries (2013) and Paquot and Gries (2020) for a more exhaustive description). Below I've summarized some common statistical scenarios and their associated tests which focus on the juxtaposition of informational values and the number of variables, leaving aside alternative tests which deal with non-normal distributions, ordinal variables, *etc.*

In Table 3.25 we see **monofactorial tests**, tests with only one predictor variable.

Table 3.25: Common monofactorial tests used in IDA.

Dependent	Predictor	Test
Categorical	Categorical	Pearson's Chi-squared test
Numeric	Categorical	Student's t-Test
Numeric	Numeric	Pearson's correlation test

Table 3.26 includes a listing of **multifactorial tests**, tests with more than one predictor and/ or control variables.

Table 3.26: Common multifactorial tests used in IDA.

Dependent	Predictor	Control	Test
Categorical	varied	varied	Logistic regression
Numeric	varied	varied	Linear regression

IDA relies heavily on quantitative evaluation methods to draw conclusions that can be generalized to the target population. It is key to understand that our goal in hypothesis testing is not to find evidence in support of H_1, but rather to assess the likelihood that we can reliably reject H_0. The metric used to determine if there is sufficient evidence is based on the probability that given the nature of the relationship and the characteristics of the data, the likelihood of there being no difference or relationship is low. The threshold for likelihood has traditionally been summarized in the p-value statistic. In the Social Sciences, a p-value lower than .05 is considered *statistically significant* which when interpreted correctly means that there is more than a 95% chance that the observed relationship would not be predicted by H_0. Note that we are working in the realm of probability, not in absolutes, therefore an analysis that produces a significant result does not prove H_1 is correct or that H_0 is incorrect, for that matter. A margin of error is always present. For this reason, other metrics such as effect size and confidence intervals are also used to interpret the results of statistical tests.

3.3 Communicate

Conducting research should be enjoyable and personally rewarding but the effort you have invested and knowledge you have generated should be shared with others. Whether part of a blog, presentation, journal article, or for your own purposes it is important to document your analysis results and process in a way that is informative and interpretable. This enhances the value of your work, allowing others to learn from your experience and build on your findings.

3.3.1 Report

The most widely recognized form of communicating research is through a report. A report is a narrative of your analysis, including the research question, the data you used, the methods you applied, and the results you obtained. We are both reporting our findings and documenting our process to inform others of what we did and why we did it but also to invite readers to evaluate our findings for themselves. The scientific process is a collaborative one and evaluation by peers is a key component of the process.

The audience for your report will determine the level of detail and the type of information you will need to include in your report but there are some common elements to reference in any report. First, the research question and/ or hypothesis should be clearly stated and the

motivation for the question should be explained. This will help the reader understand the context of the analysis and the importance of the results. Second, diagnostic procedures to verifiy or describe the data should be explained. This may include anomaly correction, missing data, data transformation, etc. and/ or descriptive summaries of the data including assessments of individual variables (central tendency, dispersion, distribution) and/ or relationships between variables (association strength). Third, a blueprint of the methods used will describe the variable selection process, how the variables are operationalized, what analysis methods are employed, and how the variables are used in the statistical analysis. Fourth, the results from the analysis are reported. Reporting details will depend on the type of analysis and the particular method(s) employed. For inferential analyses this will include the test statistic(s) and some measure of confidence. In predictive analyses, accuracy results will be reported. For exploratory analyses, the reporting of results will vary and often include visualizations and metrics that require more human interpretation than the other analysis types. Finally, the results are interpreted in light of the research question and/ or hypothesis. This will include a discussion of the limitations of the analysis and a discussion of the implications of the results for future research.

3.3.2 Document

While a good report will include the most vital information to understand the procedures, results, and findings of an analysis, there is much more information generated in the course of an analysis which does not traditionally appear in prose. If a research project is conducted programmatically, however, data, code, and documentation can be made available to others as part of the communication process. Increasingly, researchers are sharing their data and code as part of the publication process. This allows others to reproduce the analysis and verify the results contributing to the collaborative nature of the scientific process. 🔧 [CITATION]

Together, data, code, and documentation form a **research compendium**. As you can imagine the research process can quickly become complex and unwieldy as the number of files and folders grows. If not organized properly, it can be difficult to find the information you need. Furthermore, if not documented, decisions made in the course of the analysis can be difficult or impossible to trace. For this reason it is recommendable to follow a set of best practices for organizing and documenting your research compendium.

We will have more to say about this in the next chapter but for now it will suffice to point to some key elements in a research compendium. First, the data used in the analysis should be saved as a separate file(s). As a given research project progresses to analysis, the data may be transformed and manipulated to best fit the needs of the analysis. Preserving the data at each stage adds to the complete picture of the data from collection to analysis. Second, since you are working programmatically, you can share your precise analysis step-by-step in code form. This allows others to reproduce your analysis and verify your results. Including code comments provides additional information to communicate the steps taken and your thought process. Finally, a codebook documents any additional information that helps understand the

research better. This will often include guides for installing software and running the code to reproduce the analysis and an overview of the aims of the scripts and the contents of the data and datasets.

Summary

In this chapter we have focused on description and analysis –the third component of DIKI Hierarchy. This process is visually summarized in Figure 3.17.

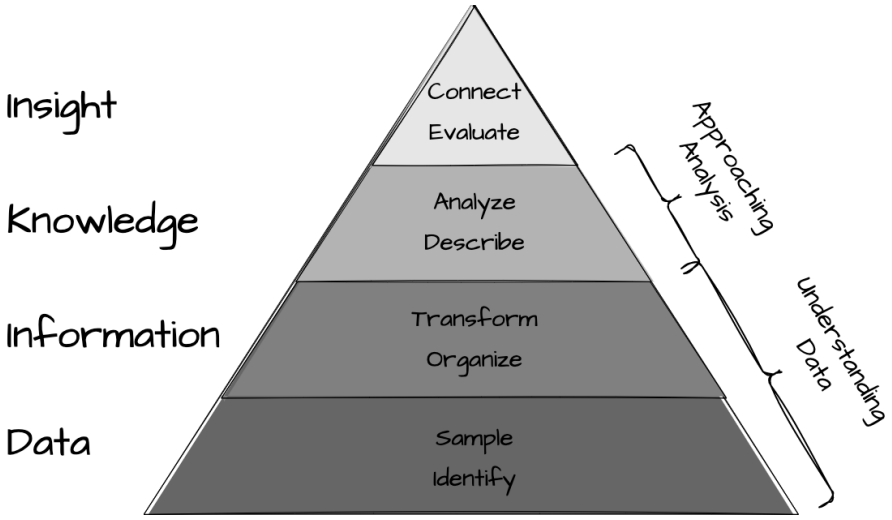


Figure 3.17: Approaching analysis: visual summary

Building on the strategies covered in Chapter 2 "Understanding data" to derive a rich relational dataset, in this chapter we outlined key points in approaching analysis. The first key step in any analysis is to perform a diagnostic assessment of the individual variables and relationships between variables. To select the appropriate descriptive measures we covered the various informational values that a variable can take. In addition to providing key information for reporting purposes, descriptive measures are important to explore so the researcher can get a better feel for the dataset before conducting an analysis.

We outlined three data analysis types in this chapter: exploratory, predictive, and inferential. Each of these embodies distinct approaches to deriving knowledge from data. Ultimately the choice of analysis type is highly dependent on the goals of the research. Inferential analysis is centered around the goal of testing a hypothesis, and for this reason it is the most highly structured approach to analysis. This structure is aimed at providing the mechanisms to draw conclusions from the results that can be generalized to the target population. Predictive analysis has a less-ambitious but at times more relevant goal of examining the extent to which

a given relationship can be established from the data to provide a model of language that can accurately predict an outcome using new data. This methodology is highly effective for applying different algorithmic approaches and examining relationships between an outcome variable and various configurations of variables. The ability to explore the data in multiple ways, is also a key strength of employing an exploratory analysis. The least structured and most variable of the analysis types, exploratory analyses are a powerful approach to generating knowledge from data in an area where clear predictions cannot be made.

I rounded out this chapter with a short description of the importance of communicating the analysis process and results. Reporting, in its traditional form, is documented in prose in an article. This reporting aims to provide the key information that a reader will need to understand what was done, how it was done, and why it was done. This information also provides the necessary information for reader's with a critical eye to understand the analysis in more detail. Yet even the most detailed reporting in a write-up still leaves many practical, but key, points of the analysis obscured. A programming approach provides the procedural steps taken that when shared provide the exact methods applied. Together with the write-up, a research compendium which provides the scripts to run the analysis and documentation on how to run the analysis forms an integral part of creating reproducible research.

Activities

☐ Add description of outcomes …

Recipe
What: Descriptive assessment of datasets[a]
How: Read Recipe 3 and participate in the Hypothes.is online social annotation.
Why: To explore appropriate methods for summarizing variables in datasets given the number and informational values of the variable(s).

[a]https://qtalr.github.io/qtalrkit/articles/recipe-3.html

Lab
What: Descriptive assessment of datasets[a]
How: Clone, fork, and complete the steps in Lab 3.
Why: To identify and apply the appropriate descriptive methods for a vector's informational value and to assess both single variables and multiple variables with the appropriate statistical, tabular, and/ or graphical summaries.

[a]https://github.com/qtalr/lab-3

Questions

Conceptual questions

- What are the key differences between assessment and analysis?
- What are the potential measures of central tendency and dispersion for a variable? Does it depend on the informational value of the variable?
- Consider the following variables: X = number of children, Y = number of siblings, Z = number of siblings who are older than the participant. Which of these variables are categorical, ordinal, numeric? What are the measures of central tendency and dispersion for each variable?
- What type(s) of tables or plots are appropriate for summarizing a variable? What type(s) of tables or plots are appropriate for summarizing the relationship between two variables?
- In the following variables and informational values, identify if the plots are appropriate for summarizing the relationship.
  🔧 …
- What are the key differences between exploratory, predictive, and inferential analysis?
- How do the goals of the research influence the choice of analysis type?
- Given the following research questions, identify which type of analysis is most appropriate and why.
  🔧 …
- How are the results of inferential, predictive, and exploratory analysis evaluated?
- Research compendia are an important part of reproducible research. What are the key components of a research compendium? What are the benefits of sharing a research compendium?

🔧 **Technical exercises**

- Create a contingency table for the following variables:
- Create a plot for the following variables:
- Report these tables and plots with a short interpretation of what they show.
- …

4 Framing research

Draft

Ready for review.

> Thus, the task is, not so much to see what no one has seen yet; but to think what nobody has thought yet, about that what everybody sees.
>
> — Arthur Schopenhauer

Outcomes

- Identify a research area and problem by listing key strategies and describing their contribution towards research identification.
- Explain the significance of a well-framed research question in guiding the overall research project.
- Comprehend how the conceptual and practical steps involved in developing a research blueprint aid not only the researcher but also the broader scientific community.

At this point in this part of the textbook, we have covered Data, Information, and Knowledge from the Data to Insight Hierarchy. The goal has been to provide an orientation to the main building blocks of doing text analysis. Insight is the last component of the hierarchy. However, in practical terms, it is the first step to address in an research project as goals of a research project influence all subsequent steps.

In this chapter we discuss how to frame research, that is how to position your research project's findings to contribute insight to understanding of the world. We will cover how to connect with the literature, selecting a research area and identifying a research problem, and how to design research best positioned to return relevant findings that will connect with this literature, establishing a research aim and research question. We will round out this chapter with a guide on developing a research blueprint –a working plan to organize the conceptual and practical steps to implement the research effectively and in a way that supports communicating the research findings and the process by which the findings were obtained.

>_ Swirl lesson
What: Version control[a]
How: In the R Console pane load `swirl`, run `swirl()`, and follow prompts to select the lesson.
Why: 🔧 To ….

[a]https://github.com/qtalr/lessons

4.1 Frame

Together a research area, problem, aim and question and the research blueprint that forms the conceptual and practical scaffolding of the project ensure from the outset that the project is solidly grounded in the main characteristics of good research. These characteristics, summarized by Cross (2006), are found in Table 4.1.

Table 4.1: Characteristics of research (Cross, 2006).

Characteristic	Description
Purposive	Based on identification of an issue or problem worthy and capable of investigation
Inquisitive	Seeking to acquire new knowledge
Informed	Conducted from an awareness of previous, related research
Methodical	Planned and carried out in a disciplined manner
Communicable	Generating and reporting results which are feasible and accessible by others

With these characteristics in mind, let's get started with the first component to address – connecting with the literature.

4.2 Connect

4.2.1 Research area

The first decision to make in the research process is to identify a research area. A **research area** is a general area of interest where a researcher wants to derive insight and make a contribution to understanding. For those with an established research trajectory in language, the area of research to address through text analysis will likely be an extension of their prior work. For others, which include new researchers or researcher's that want to explore new areas of language research or approach an area through a language-based lens, the choice of area

may be less obvious. In either case, the choice of a research area should be guided by a desire to contribute something relevant to a theoretical, social, and/ or practical matter of personal interest. Personal relevance goes a long way to developing and carrying out **purposive** and **inquisitive** research.

So how do we get started? The first step is to reflect on your own areas of interest and knowledge, be it academic, professional, or personal. Language is at the heart of the human experience and therefore found in some fashion anywhere one seeks to find it. But it is a big world and more often than not the general question about what area to explore language use is sometimes the most difficult. To get the ball rolling, it is helpful to peruse disciplinary encyclopedias or handbooks of linguistics and language-related an academic fields (e.g. Encyclopedia of Language and Linguistics[1] (Brown 2005), A Practical Guide to Electronic Resources in the Humanities[2] (Dubnjakovic and Tomlin 2010), Routledge encyclopedia of translation technology[3] (Chan 2014))

A more personal, less academic, approach is to consult online forums, blogs, *etc.* that one already frequents or can be accessed via an online search. For example, Reddit[4] has a wide variety of active subreddits (r/LanguageTechnology[5], r/Linguistics[6], r/corpuslinguistics[7], r/DigitalHumanities[8], *etc.*). Twitter and Facebook also have interesting posts on linguistics and language-related fields worth following. Through one of these social media site you may find particular people that maintain a blog worth browsing. For example, I follow Julia Silge[9], Rachel Tatman[10], and Ted Underwood[11], *inter alia.* Perusing these resources can help spark ideas and highlight the kinds of questions that interest you.

Regardless of whether your inquiry stems from academic, professional, or personal interest, try to connect these findings to academic areas of research. Academic research is highly structured and well-documented and making associations with this network will aid in subsequent steps in developing a research project.

4.2.2 Research problem

Once you've made a rough-cut decision about the area of research, it is now time to take a deeper dive into the subject area and jump into the literature. This is where the rich structure

[1] https://www.sciencedirect.com/referencework/9780080448541/encyclopedia-of-language-and-linguistics
[2] https://www.sciencedirect.com/book/9781843345978/a-practical-guide-to-electronic-resources-in-the-humanities
[3] https://www.routledgehandbooks.com/doi/10.4324/9781315749129
[4] https://www.reddit.com/
[5] https://www.reddit.com/r/LanguageTechnology/
[6] https://www.reddit.com/r/linguistics/
[7] https://www.reddit.com/r/corpuslinguistics/
[8] https://www.reddit.com/r/DigitalHumanities/
[9] https://juliasilge.com/
[10] http://www.rctatman.com/
[11] https://tedunderwood.com/

of disciplinary research will provide aid to traverse the vast world of academic knowledge and identify a research problem. A **research problem** highlights a particular topic of debate or uncertainty in existing knowledge which is worthy of study.

Surveying the relevant literature is key to ensuring that your research is **informed**, that is, connected to previous work. Identifying relevant research to consult can be a bit of a 'chicken or the egg' problem –some knowledge of the area is necessary to find relevant topics, some knowledge of the topics is necessary to narrow the area of research. Many times the only way forward is to jump into conducting searches. These can be world-accessible resources (*e.g.* Google Scholar[12]) or limited-access resources that are provided through an academic institution (*e.g.* Linguistics and Language Behavior Abstracts[13]), ERIC[14], PsycINFO[15], *etc.*). Some organizations and academic institutions provide research guides[16] to help researcher's access the primary literature.

Another avenue to explore are journals dedicated to areas in which linguistics and language-related research is published. In Table 4.2, Table 4.3, and Table 4.4, I've listed a number of highly visible journals in linguistics, digital humanities, and computational linguistics.

Table 4.2: A list of some linguistics journals.

Resource	Description
Corpora	An international, peer-reviewed journal of corpus linguistics focusing on the many and varied uses of corpora both in linguistics and beyond.
Corpus Linguistics and Linguistic Theory	Corpus Linguistics and Linguistic Theory (CLLT) is a peer-reviewed journal publishing high-quality original corpus-based research focusing on theoretically relevant issues in all core areas of linguistic research, or other recognized topic areas.
International Journal of Corpus Linguistics	The International Journal of Corpus Linguistics (IJCL) publishes original research covering methodological, applied and theoretical work in any area of corpus linguistics.
International Journal of Language Studies	It is a refereed international journal publishing articles and reports dealing with theoretical as well as practical issues focusing on language, communication, society and culture.
Journal of Child Language	A key publication in the field, Journal of Child Language publishes articles on all aspects of the scientific study of language behaviour in children, the principles which underlie it, and the theories which may account for it.

[12]https://scholar.google.com/
[13]https://about.proquest.com/en/products-services/llba-set-c
[14]https://eric.ed.gov/
[15]https://www.ebsco.com/products/research-databases/apa-psycinfo
[16]https://guides.zsr.wfu.edu/linguistics

Resource	Description
Journal of Linguistic Geography	The Journal of Linguistic Geography focuses on dialect geography and the spatial distribution of language relative to questions of variation and change.
Journal of Quantitative Linguistics	Publishes research on the quantitative characteristics of language and text in mathematical form, introducing methods of advanced scientific disciplines.

Table 4.3: A list of some humanities journals.

Resource	Description
Digital Humanities Quarterly	Digital Humanities Quarterly (DHQ), an open-access, peer-reviewed, digital journal covering all aspects of digital media in the humanities.
Digital Scholarship in the Humanities	DSH or Digital Scholarship in the Humanities is an international, peer reviewed journal which publishes original contributions on all aspects of digital scholarship in the Humanities including, but not limited to, the field of what is currently called the Digital Humanities.
Journal of Cultural Analytics	Cultural Analytics is an open-access journal dedicated to the computational study of culture. Its aim is to promote high quality scholarship that applies computational and quantitative methods to the study of cultural objects (sound, image, text), cultural processes (reading, listening, searching, sorting, hierarchizing) and cultural agents (artists, editors, producers, composers).

Table 4.4: A list of some computational linguistics journals.

Resource	Description
Computational Linguistics	Computational Linguistics is the longest-running publication devoted exclusively to the computational and mathematical properties of language and the design and analysis of natural language processing systems.
LREC Conferences	The International Conference on Language Resources and Evaluation is organised by ELRA biennially with the support of institutions and organisations involved in HLT.

Resource	Description
Transactions of the Association for Computational Linguistics	Transactions of the Association for Computational Linguistics (TACL) is an ACL-sponsored journal published by MIT Press that publishes papers in all areas of computational linguistics and natural language processing.

To explore research related to text analysis it is helpful to start with the (sub)discipline name(s) you identified in when selecting your research area, more specific terms that occur to you or key terms from the literature, and terms such as 'corpus study' or 'corpus-based'. The results from first searches may not turn out to be sources that end up figuring explicitly in your research, but it is important to skim these results and the publications themselves to mine information that can be useful to formulate better and more targeted searches. Relevant information for honing your searches can be found throughout an academic publication (article or book). However, pay particular attention to the abstract, in articles, and the table of contents, in books, and the cited references. Abstracts and tables of contents often include discipline-specific jargon that is commonly used in the field. In some articles there is even a short list of key terms listed below the abstract which can be extremely useful to seed better and more precise search results. The references section will contain relevant and influential research. Scan these references for publications which appear to narrowing in on topic of interest and treat it like a search in its own right.

Once your searches begin to show promising results it is time to keep track and organize these references. Whether you plan to collect thousands of references over a lifetime of academic research or your aim is centered around one project, software such as Zotero[17][18], Mendeley[20], or BibDesk[21] provide powerful, flexible, and easy-to-use tools to collect, organize, annotate, search, and export references. Citation management software is indispensable for modern research –and often free!

As your list of relevant references grows, you will want to start the investigation process in earnest. Begin skimming (not reading) the contents of each of these publications, starting with the most relevant first[22]. Annotate these publications using highlighting features of the citation management software to identify: (1) the stated goal(s) of the research, (2) the data source(s) used, (3) the information drawn from the data source(s), (4) the analysis approach employed, and (5) the main finding(s) of the research as they pertain to the stated goal(s). Next, in your own words, summarize these five key areas in prose adding your summary to the notes feature of the citation management software. This process will allow you to efficiently gather and document references with the relevant information to guide the identification of

[17]https://www.zotero.org/
[18]Zotero Guide[19]
[20]https://www.mendeley.com/reference-management/reference-manager
[21]https://bibdesk.sourceforge.io/
[22]Or what appears to be most relevant. This may change as you start to take a closer look.

a research problem, to guide the formation of your problem statement, and ultimately, to support the literature review that will figure in your project write-up.

From your preliminary annotated summaries you will undoubtedly start to recognize overlapping and contrasting aspects in the research literature. These aspects may be topical, theoretical, methodological, or appear along other lines. Note these aspects and continue to conduct more refine searches, annotate new references, and monitor for any emerging patterns of uncertainty or debate (gaps) which align with your research interest(s). When a promising pattern takes shape, it is time to engage with a more detailed reading of those references which appear most relevant highlighting the potential gap(s) in the literature.

At this point you can focus energy on more nuanced aspects of a particular gap in the literature with the goal to formulate a problem statement. A **problem statement** directly acknowledges a gap in the literature and puts a finer point on the nature and relevance of this gap for understanding. This statement reflects your first deliberate attempt to establish a line of inquiry. It will be a targeted, but still somewhat general, statement framing the gap in the literature that will guide subsequent research design decisions.

4.3 Define

4.3.1 Research aim

With a problem statement in hand, it is now time to consider the goal(s) of the research. A **research aim** frames the type of inquiry to be conducted. Will the research aim to explore, examine, or explain? In other words, will the research seek to uncover novel relationships, assess the potential strength of a particular relationship, or test a particular relationship? As you can appreciate, the research aim is directly related to the analysis methods we touched upon in Chapter 3.

To gauge how to frame your research aim, reflect on the literature that led you to your problem statement and the nature of the problem statement itself. If the gap at the center of the problem statement is a lack of knowledge, your research aim may be exploratory. If the gap concerns a conjecture about a relationship, then your research may take a predictive approach. When the gap points to the validation of a relationship, then your research will likely be inferential in nature. Before selecting your research aim it is also helpful to consult the research aims of the primary literature that led you to your research statement.

Typically, a problem statement addressing a subtle, specific issue tends to adopt research objectives similar to prior studies. In contrast, a statement focusing on a broader, more distinct issue is likely to have unique research goals. Yet, this is more of a guideline than a strict rule.

It's crucial to understand both the existing literature and the nature of various types of analyses. Being clear about your research goals is important to ensure that your study is

well-placed to produce results that add value to the current understanding in an informed manner.

4.3.2 Research question

The next step in research design is to craft the research question. A **research question** is clearly defined statement which identifies an aspect of uncertainty and the particular relationships that this uncertainty concerns. The research question extends and narrows the line of inquiry established in the research statement and research aim. To craft a research question, we can use the research statment for the content and the research aim for the form.

Form

The form of a research question will vary based on the research aim, which as I mentioned, is inimately connected to the analysis approach. For inferential-based research, the research question will actually be a statement, not a question. This statement makes a testable claim about the nature of a particular relationship –*i.e.* asserts a hypothesis.

For illustration, let's return to the hypothesis (H_1) we previously sketched out in Chapter 3, leaving aside the implicit null hypothesis, seen in Example 4.1.

Example 4.1. Women use more questions than men in spontaneous conversations.

For predictive- and exploratory-based research, the research question is in fact a question. A reframing of the example hypothesis for a predictive-based research question might take the form seen in Example 4.2.

Example 4.2. Can the number of questions used in spontaneous conversations predict if a speaker is male or female?

And a similar exploratory-based research question might take the form seen in Example 4.3.

Example 4.3. Do men and women differ in terms of the number of questions they use in spontaneous conversations?

The central research interest behind these hypothetical research questions is, admittedly, quite basic. But from these simplified examples, we are able to appreciate the similarities and differences between the forms of research statements that correspond to distinct research aims.

Content

In terms of content, the research question will make reference to two key components. First, is the unit of analysis. The **unit of analysis** is the entity which the research aims to investigate. For our three example research aims, the unit of analysis is the same, namely *speakers*. Note, however, that the current unit of analysis is somewhat vague in the example research questions. A more precise unit of analysis would include more information about the population from which the speakers are drawn (*i.e.* English speakers, American English speakers, American English speakers of the Southeast, *etc.*).

The second key component is the unit of observation. The **unit of observation** is the primary element on which the insight into the unit of analysis is derived and in this way constitutes the essential organizational unit of the dataset to be analyzed. In our examples, the unit of observation, again, is unchanged and is *spontaneous conversations*. Note that while the unit of observation is key to identify as it forms the organizational backbone of the research, it is very common for the research to derive variables from this unit to provide evidence to investigate the research question.

In examples 4.1, 4.2, and 4.3, we identified the number of conversations as part of the research question. Later in the research process it will be key to operationalize this variable. For example, will the number of conversations be the total number of conversations in the dataset or will it be the average number of conversations per speaker? These are important questions to consider as they will influence variable selection, statistical choices, and ultimately the interpretation of the results.

4.4 Blueprint

Efforts to craft a research question are a very important aspect of developing purposive, inquisitive, and informed research (returning to Cross's characteristics of research). Moving beyond the research question in the project means developing and laying out the research design in a way such that the research is **Methodical** and **Communicable**. In this textbook, the method to achieve these goals is through the development of a research blueprint. The blueprint includes two components: (1) laying out a conceptual plan and (2) deriving the organizational scaffolding that will support the implementation of the research.

As Ignatow and Mihalcea (2017) point out:

> Research design is essentially concerned with the basic architecture of research projects, with designing projects as systems that allow theory, data, and research methods to interface in such a way as to maximize a project's ability to achieve its goals [...]. Research design involves a sequence of decisions that have to be taken in a project's early stages, when one oversight or poor decision can lead to results that are ultimately trivial or untrustworthy. Thus, it is critically important to

> think carefully and systematically about research design before committing time and resources to acquiring texts or mastering software packages or programming languages for your text mining project.

In what follows, I will cover the main aspects of developing a research blueprint. I will start with the conceptual plan and then move on to the organizational scaffolding.

4.4.1 Plan

Importance of establishing a feasible research design from the outset and documenting the key aspects required to conduct the research cannot be understated. On the one hand this process links a conceptual plan to a tangible implementation. In doing so, a researcher is better-positioned to conduct research with a clear view of what will be entailed. On the other hand, a promising research question may present unexpected challenges once a researcher sets about to implement the research. This is not uncommon to encounter issues that require modification or reevaluation of the viability of the project. However, a well-documented research plan will help a researcher to identify and address many of these challenges at the conceptual level before expending effort on the implementation.

Let's now consider the main aspects of developing a research plan: identifying data source(s), key variables, and analysis methods. Before we do, however, it is important to reiterate the importance of a research question or hypothesis before moving forward in research planning. The research question or hypothesis is the central component of the research plan. It guides every step which follows from data selection to interpretation of the analysis results. Furthermore, a well-founded research question is based on a solid literature review from which can provide helpful guidance at key choice points in the research process.

First, **identify a viable data source**. Viability includes the accessibility of the data, availability of the data, and the content of the data. If a purported data source is not accessible and/ or is has stringent restrictions on its use, then it is not a viable data source. If a data source is accessible and available, but does not contain the building blocks needed to address the research question, then it is not a viable data source. In the case that research is inferential in nature, the sampling frame of the corpus is of primary importance as the goal is to generalize the findings to a target population. A corpus resource should align, to the extent feasible, with this target population. For predictive and exploratory research, the goal to generalize a claim is not central and for this reason the there is some freedom in terms of how representative a corpus sample is of a target population. Ideally, a researcher will find and be able to model a language population of target interest. Since the goal, however, is not to test a hypothesis, but rather to explore particular or evaluate potential relationships, either in an exploratory or predictive fashion, the research can often continue with the stipulation that the results are interpreted in the light of the characteristics of the available corpus sample.

The second step is to **identify the key variables** need to conduct the research are and then ensure that this information can be derived from the corpus data. The research question will

reference the unit of analysis and the unit of observation, but it is important at this point to then pinpoint what the key variables will be. If the unit of observation is spontaneous conversations. The question as to what aspects of these conversations will be used in the analysis. In the research questions presented in this chapter, we will want to envision what needs to be done to derive a variable which measures the number of questions in each of the conversations. In other research, their may be features that need to be extracted, recoded, and/ or generated to address the research question. Other variables of importance may be non-linguistic in nature. In cases where there the metadata is incomplete for the goals of the research, it is sometimes possible to merge metadata from other sources.

The third step is to **identify a method of analysis** to interrogate the dataset. The selection of the analysis approach that was part of the research aim (*i.e.* explore, predict, or infer) and then the research question goes a long way to narrowing the methods that a researcher must consider. But there are a number of factors which will make some methods more appropriate than others.

Exploratory research is the least restricted of the three types of analysis approaches. Although it may be the case that a research will not be able to specify from the outset of a project what the exact analysis methods will be, an attempt to consider what types of analysis methods will be most promising to provide results to address the research question goes a long way to steering a project in the right direction and grounding the research. As with the other analysis approaches, it is important to be aware of what the analysis methods available and what type of information they produce in light of the research question.

For predictive-based research, the informational value of the target variable is key to deciding whether the prediction will be a classification task or a regression task. This has downstream effects when it comes time to evaluate and interpret the results. Although the feature engineering process in predictive analyses means that the features do not need to be specified from the outset and can be tweaked and changed as needed during an analysis, it is a good idea to start with a basic sense of what features most likely will be helpful in developing a robust predictive model. Furthermore, while the number and informational values of the features (predictor variables) are not as important to selecting a prediction method (algorithm) as they are in inferential analysis methods, it is important to recognize that particular algorithms have strengths and shortcomings when working large numbers and/ or types of features (Lantz 2013).

In inferential research, the number and information values of the variables to be analyzed will be of key importance (S. Th. Gries 2013). The informational value of the dependent variable will again narrow the search for the appropriate method. The number of predictor variables also plays an important role. For example, a study with a categorical dependent variable with a single categorical predictor variable will lead the researcher to the Chi-squared test. A study with a numeric dependent variable with multiple predictor variables will lead to linear regression. Another aspect of note for inference studies is the consideration of the distribution of numeric variables –a normal distribution will use a parametric test where a non-normal distribution will use a non-parametric test. These details need not be nailed down at this

point, but it is helpful to have them on your radar to ensure that when the time comes to analyze the data, the appropriate steps are taken to test for normality and then apply the correct test.

The last of the main components of the research plan concerns the **interpretation and evaluation of the results**. This step brings the research plan full circle connecting the research question to the methods employed. It is important to establish from the outset what the criteria will be to evaluate the results. This is in large part a function of the relationship between the research question and the analysis method. For example, in exploratory research, the results will be evaluated qualitatively in terms of the associative patterns that emerge. Predictive and inferential research leans more heavily on quantitative metrics in particular the accuracy of the prediction or the strength of the relationship between the dependent and predictor variable(s), respectively. However, these quantitative metrics require qualitative interpretation to determine whether the results are meaningful in light of the research question.

To summarize these planning steps, I've created a checklist in Table 4.5.

Table 4.5: Research Plan Checklist

Steps	Description
Research Question or Hypothesis	Formulate a research question or hypothesis based on a thorough review of existing literature including references. This will guide every subsequent step from data selection to interpretation of results.
Data Source(s)	Identify viable data source(s) and vet the sample data in light of the research question. Consider to what extent the goal is to generalize findings to a target population, and ensure that the corpus aligns as much as feasible with this target.
Key Variables	Determine the key variables needed for the research, define how they will be operationalized, and ensure they can be derived from the corpus data. Additionally, identify any features that need to be extracted, recoded or generated.
Analysis Method	Choose an appropriate method of analysis to interrogate the dataset. This choice should be in line with your research aim (*e.g.*, exploratory, predictive, inferential). Be aware of what each method can offer and how it addresses your research question.
Interpretation & Evaluation	Establish criteria to interpret and evaluate the results. This will be a function of the relationship between the research question and the analysis method.

In addition to addressing the steps outlined in Table 4.5, it is also important to document the strengths and shortcomings of the research plan including the data source(s), the information to be extracted from the data, and the analysis methods. If there are potential shortcomings,

which there most often are, sketch out contingency plans to address these shortcomings. This will help buttress your research and ensure that your time and effort is well-spent.

Together the information collected from this process will serve to guide the research and provide a solid foundation for the research write-up. Furthermore, you may consider pre-registering your research project to ensure that your plans are well-documented and to provide a timestamp for your research. Pre-registration can also be a helpful way to get feedback on your research plan from colleagues and experts in the field. Popular pre-registration platforms include Open Science Framework[23] and Center for Open Science[24].

4.4.2 Scaffold

The next step in creating a research blueprint is to consider how to physically implement your project. This includes how to organize files and directories in a fashion that both provides the researcher a logical and predictable structure to work with but also ensures that the research is **Communicable**. On the one hand, communicable research includes a strong write-up of the research, but, on the other hand, it is also important that the research is reproducible. Reproducibility strategies are a benefit to the researcher (in the moment and in the future) as it leads to better work habits and to better teamwork and it makes changes to the project easier. Reproducibility is also of benefit to the scientific community as shared reproducible research enhances replicability and encourages cumulative knowledge development (Gandrud 2015).

There are a set of guiding principles to accomplish these goals (Gentleman and Temple Lang 2007; Marwick, Boettiger, and Mullen 2018), seen in Example 4.4.

Example 4.4. Reproducible Research Principles

1. All files should be plain text which means they contain no formatting information other than whitespace.
2. There should be a clear separation between the data, method, and output of research. This should be apparent from the directory structure.
3. A separation between original, derived, and analysis data should be made. Original data should be treated as 'read-only'. Any changes to the original data should be justified, generated by the code, and documented (see point 6).
4. Each project file (script) should represent a particular, well-defined step in the research process.
5. Each project script should be modular –that is, each file should correspond to a specific goal in the analysis procedure with input and output only corresponding to this step.
6. All project scripts should be tied together by a 'main' script that is used to coordinate the execution of all the project steps.

[23] https://osf.io/
[24] https://www.cos.io/initiatives/prereg

7. Everything should be documented. This includes data collection, data preprocessing, analysis steps, script code comments, data description in data dictionaries, information about the computing environment and packages used to conduct the analysis, and detailed instructions on how to reproduce the research.

These seven principles can be physically implemented in numerous ways. In recent years, there has been a growing number of efforts to create R packages and templates to quickly generate the scaffolding and tools to facilitate reproducible research. Some notable R packages include workflowr[25] (Blischak, Carbonetto, and Stephens 2021), ProjectTemplate[26] (White 2023), and targets[27] (Landau 2023), but there are many other resources for R included on the CRAN Task View for Reproducible Research[28]. There are many advantages to working with pre-existing frameworks for the savvy R programmer including the ability to quickly generate a project scaffold, to efficiently manage changes to the project, and to buy in to a common framework that is supported by a community of developers.

On the other hand, these frameworks can be a bit daunting for the novice R programmer. At the most basic level, a project can implement the seven principles outlined above by creating a directory structure and a set of files manually.

Example 4.5. Minimal Project Framework

```
project/
├── data/
│   ├── analysis/
│   ├── derived/
│   └── original/
├── output/
│   ├── figures/
│   ├── reports/
│   ├── results/
│   └── tables/
├── code/
│   └── ...
├── _main.R
└── README
```

In Example 4.5, I provide a minimal framework that aligns with the reproducible research principles. Let me now make the connections between the principles and this project structure.

[25]https://jdblischak.github.io/workflowr/
[26]http://projecttemplate.net/
[27]https://github.com/ropensci/targets
[28]https://cran.r-project.org/web/views/ReproducibleResearch.html

The *project/* directory is composed of three main sections: *data/*, *output/*, and *code/* corresponding to the input, output, and code, respectively.

The *data/* section is divided into three subsections:

- *analysis/* for storing data used to perform analysis
- *derived/* for housing data produced in curation and transformation steps
- *original/* for keeping the original 'read-only' data

The *output/* section contains four subsections:

- *figures/* for visualizations produced as part of the project
- *reports/* for the resulting reports (*e.g.* article, presentation, blog post, *etc.*)
- *results/* for statistical results from the analysis
- *tables/* for summary tables

The *code/* directory houses the code, with the *_main.R* file at the root of the project orchestrating the execution of all project steps.

Lastly, the *README* file provides a description of the project and instructions on how to reproduce the research.

The project structure in Example 4.5 meets the minimal structural requirements for reproducible research but can be augmented in more sophisticated ways to support more functionality, as we will see in Chapter 12. One enhancement that I highly recommend is the use of literate programming, in the form of Quarto documents, to serve as the main project scripts. This facilitates the combination of executable code and prose documentation for each project step in a single, modular file.

Summary

The aim of this chapter is to provide the key conceptual and practical points to guide the development of a viable research project. Good research is purposive, inquisitive, informed, methodological, and communicable. It is not, however, always a linear process. Exploring your area(s) of interest and connecting with existing work will help couch and refine your research. But practical considerations, such as the existence of viable data, technical skills, and/ or time constrains, sometimes pose challenges and require a researcher to rethink and/ or redirect the research in sometimes small and other times more significant ways. The process of formulating a research question and developing a viable research plan is key to supporting viable, successful, and insightful research. To ensure that the effort to derive insight from data is of most value to the researcher and the research community, the research should strive to be methodological and communicable adopting best practices for reproducible research.

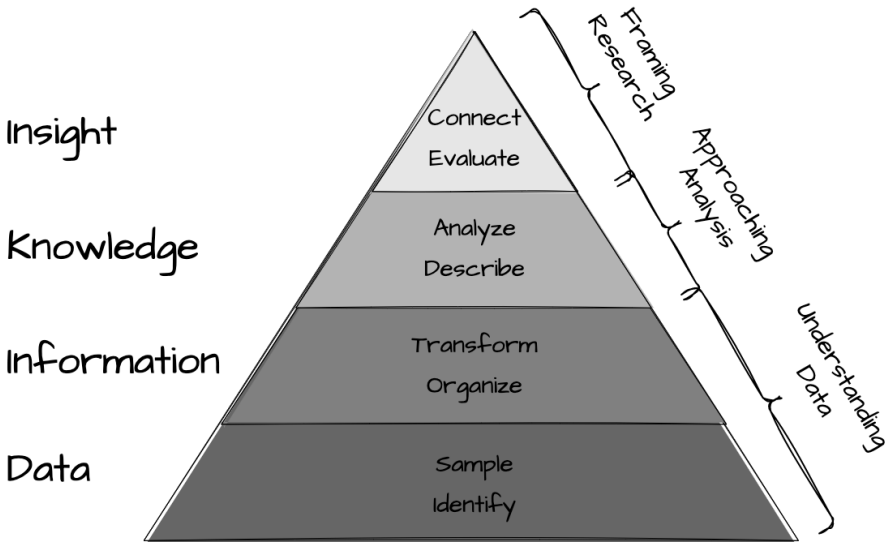


Figure 4.1: Framing research: visual summary

This chapter concludes the Foundations part of this textbook. At this stage our overview of fundamental characteristics of research are in place to move a project towards implementation, as seen in Figure 4.1. From this point forward we will integrate your conceptual knowledge and emerging R programming skills as we cover common scenarios encountered when conducting reproducible research with real-world data.

The next part, Preparation, aims to cover R coding strategies to acquire, curate, and transform data in preparation for analysis. These are the first steps in putting a research blueprint into action and by no coincidence the first components in the Data to Insight Hierarchy. Without further ado, let's get started!

Activities

☐ 🔧 Add description of outcomes …

Recipe
What: Project management[a]
How: Read Recipe 4 and participate in the Hypothes.is online social annotation.
Why: 🔧 To learn how to use … reproducible research projects.

[a] https://qtalr.github.io/qtalrkit/articles/recipe-4.html

Lab
What: Project management[a]
How: Clone, fork, and complete the steps in Lab 4.
Why: To … a reproducible research project.

[a]https://github.com/qtalr/lab-4

Questions

Conceptual questions

- What are the key characteristics of good research as described in this chapter?
- What are some strategies researchers can use to identify potential research areas and problems to investigate?
- For each strategy, describe how it contributes to research that is purposive, inquisitive, and informed.
- Why is framing a clear, focused research question or hypothesis important? Briefly explain how the research question guides the overall research process.
- What is the difference between the unit of analysis and the unit of observation? How do these concepts relate to the research question?
- What does it mean to operationalize a variable? Why is this important?
- The process of developing a research blueprint involves both conceptual planning and practical implementation steps. Explain how going through this process not only aids the individual researcher, but also the research community.
- Describe the main aspects of developing a research plan.
- Explain why it is important for research to be methodological and reproducible. What are some challenges researchers may face in achieving this?

Technical exercises

- Matching research questions with data sources
- Matching research questions with research plans
- Preregistering a research project (?)
- Propose a quantitative research topic (or question if possible). Support your topic with supporting literature. (?)
- …

Part III

Preparation

At this point we begin our journey to implement the research blueprint. As such, the content will be more focused on the practical steps to bring a plan to fruition integrating our conceptual understanding of the research process from the previous chapters with our emerging programming skills developed in lessons, recipes, and labs.

This part, Preparation, will address data acquistion, curation, and transformation steps. The goal of data preparation is to create a dataset which is ready for analysis. In each of these three upcoming chapters, I will outline some of the main characteristics to consider in each of these research steps and provide authentic examples of working with R to implement these steps. In Chapter 5 this includes downloads, working with APIs, and webscraping. In Chapter 6 we turn to organize data into rectangular, or 'tidy', format. Depending on the data or dataset acquired for the research project, the steps necessary to shape our data into a base dataset will vary, as we will see. In Chapter 7 we will work to manipulate curated datasets to create datasets which are aligned with the research aim and research question. This often includes normalizing values, recoding variables, and generating new variables as well as and sourcing and merging information from other datasets with the dataset to be submitted for analysis.

Each of these chapters will cover the necessary documentation to trace our steps and provide a record of the data preparation process. Documentation serves to inform the analysis and interpretation of the results and also forms the cornerstone of reproducible research.

5 Acquire data

Draft

Ready for review.

> The scariest moment is always just before you start.
>
> – Stephen King

Outcomes

- Identify common strategies for acquiring corpus data.
- Describe how to organize and document data acquisition to support reproducibility.
- Recall R programming concepts and strategies relevant to acquiring data.

As we start down the path to executing our research blueprint, our first step is to acquire the primary data that will be employed in the project. This chapter covers three widely-used strategies for acquiring corpus data: downloads, APIs (Application Programming Interfaces), and web scraping. We get started with the most straighforward approaches from a conceptual standpoint, gradually escalating to more nuanced methods. We will encounter various file formats and folder structures in the process and we will address how to effectively organize our data for subsequent processing. Crucial to our efforts is the process of documenting our data. We will learn to provide data origin information to ensure key characteristics of the data and its source are documented. Along the way we will explore R coding concepts including control statements and custom functions relevant to the task of acquiring data. By the end of this chapter, you will not only be adept at acquiring data from diverse sources but also capable of documenting it comprehensively, enabling you to replicate the process in the future.

Swirl lesson
What: Control Statements, Custom Functions[a]
How: In the R Console pane load `swirl`, run `swirl()`, and follow prompts to select the lesson.
Why: To recognize the logic behind code that can make dynamic choices and to recall how functions serve to produce efficient, reusable, and more legible code.

[a]https://github.com/qtalr/lessons

5.1 Downloads

The most common and straightforward method for acquiring corpus data is through direct downloads. In a nutshell, this method involves navigating to a website, locating the data, and downloading it to your computing environment. In some cases access to the data requires manual intervention and in others the process can be implemented programmatically. The data may be contained in a single file or multiple files. The files may be compressed or uncompressed. The data may be hierarchically organized or not. Each resource will have its own unique characteristics that will influence the process of acquiring the data. In this section we will work through a few examples to demonstrate the general process of acquiring data through downloads.

5.1.1 Manual

In contrast to the other data acquisition methods we will cover in this chapter, **manual downloads** require human intervention. This means that manual downloads are non-reproducible in a strict sense and require that we keep track of and document our procedure. It is a very common for research projects to acquire data through manual downloads as many data resources require some legwork before they are accessible for downloading. These can be resources that require institutional or private licensing and fees (Language Data Consortium[1], International Corpus of English[2], BYU Corpora[3], *etc.*), require authorization/ registration (The Language Archive[4], COW Corpora[5], *etc.*), and/ or are only accessible via resource search interfaces (Corpus of Spanish in Southern Arizona[6], Corpus Escrito del Español como L2 (CEDEL2)[7], *etc.*).

Let's take a look at how to acquire data from a resource that requires manual intervention. The resource we will use is the Corpus Escrito del Español como L2 (CEDEL2)[8] (Lozano 2009), a corpus of Spanish learner writing. It includes L2 writing from students with a variety of L1 backgrounds. For comparative puposes it also includes native writing for Spanish, English, and several other languages.

[1]https://www.ldc.upenn.edu/
[2]http://ice-corpora.net/ice/
[3]https://www.corpusdata.org/
[4]https://archive.mpi.nl/tla/
[5]https://www.webcorpora.org/
[6]https://cesa.arizona.edu/
[7]http://cedel2.learnercorpora.com/
[8]http://cedel2.learnercorpora.com/

The CEDEL2 corpus is a freely available resource, but to access the data you must first use a search interface to select the relevant characteristics of the data of interest. Following the search/ download link you can find a search interface that allows the user to select the subcorpus and filter the results by a set of attributes, seen in Figure 5.1.

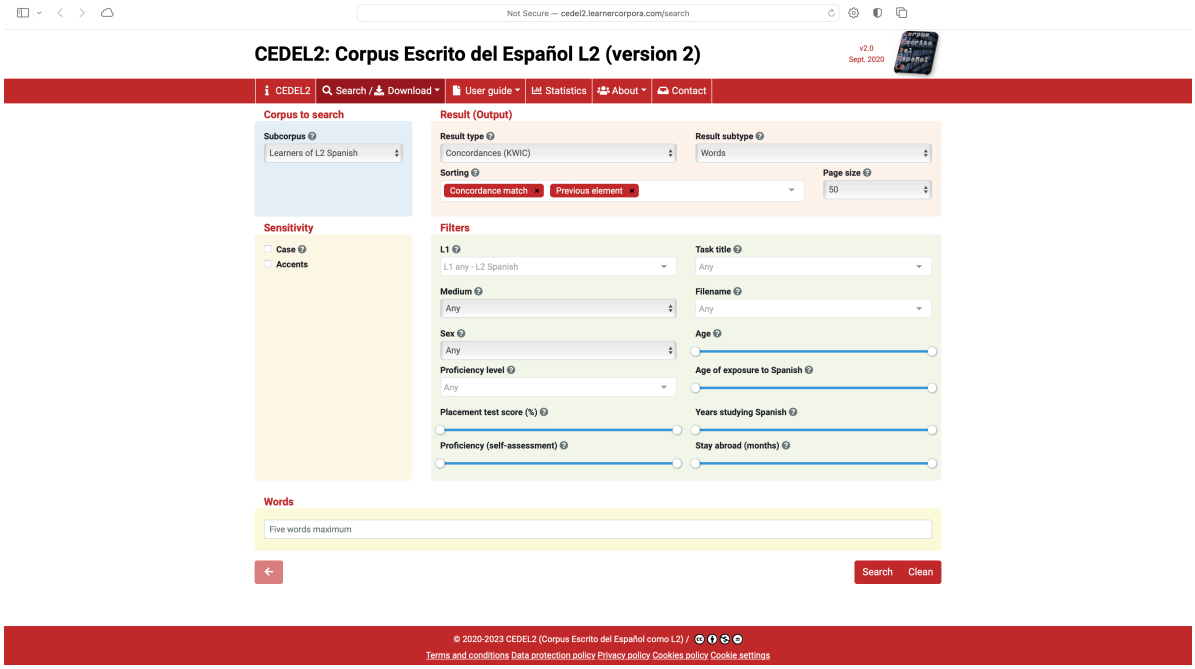


Figure 5.1: Search and download interface for the CEDEL2 Corpus

For this example let's assume that we want to acquire data to use in a study comparing the use of the Spanish preterite and imperfect past tense aspect in written texts by English L1 learners of Spanish to native Spanish speakers. To acquire data for such a project, we will first select the subcorpus "Learners of L2 Spanish". We will set the results to provide full texts and filter the results to "L1 English - L2 Spanish". Additionally, we will set the medium to "Written". This will provide us with a set of texts for the L2 learners that we can use for our study. The search parameters and results are shown in Figure 5.2.

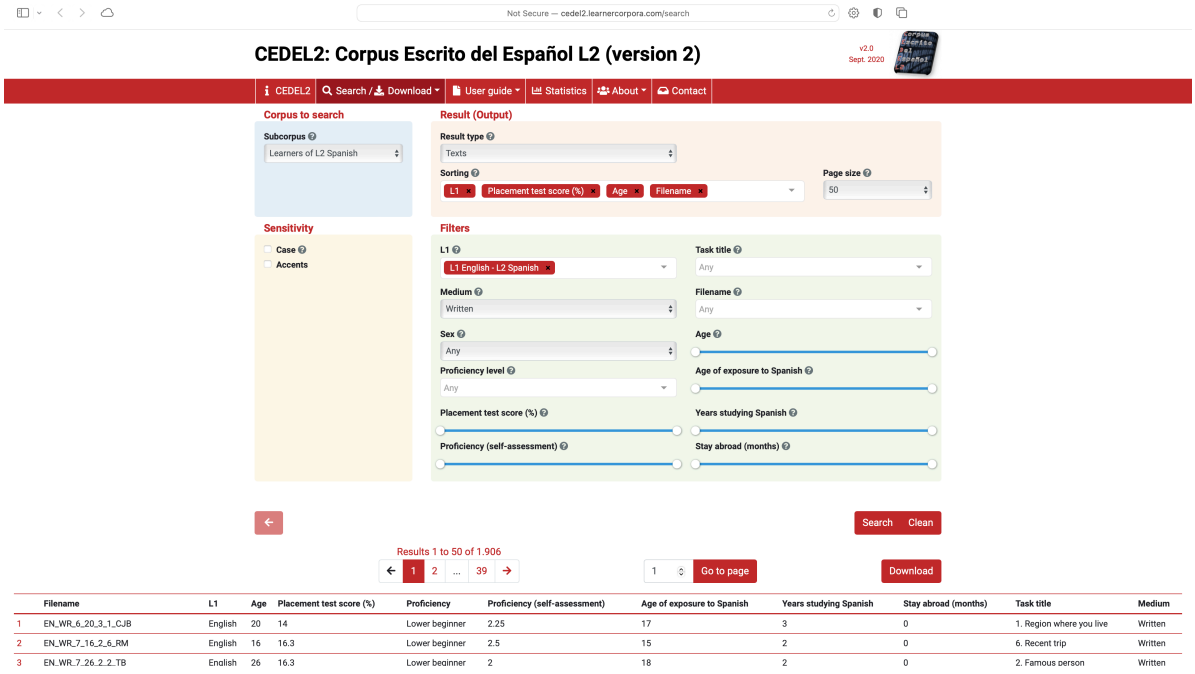


Figure 5.2: Search results for the CEDEL2 Corpus

The 'Download' link now appears for this search criteria. Following this link will provide the user a form to fill out. This particular resource allows for access to different formats to download (Texts only, Texts with metadata, CSV (Excel), CSV (Others)). I will select the 'CSV (Others)' option so that the data is structured for easier processing downstream when we work to curate the data in our next processing step. Then I will choose to save the CSV in the *data/original/* directory of my project and create a sub-directory named *cedel2/*, as seen in Example 5.1.

Example 5.1. Download CEDEL2 L2 Spanish Learners data

```
data/
├── analysis/
├── derived/
└── original/
    └── cedel2/
        └── cedel2-l1-english-learners.csv
```

Note that the file is named *cedel2-l1-english-learners.csv* to reflect the search criteria used to acquire the data. In combination with other data documentation, this will help us to maintain transparency.

Now, after downloading the L2 learner and the native speaker data into the appropriate directory, we move on to the next processing step, right? Not so fast! Imagine we are working on a project with a collaborator. How will they know where the data came from? What if we need to come back to this data in the future? How will we know what characteristics we used to filter the data? The directory and filenames may not be enough. To address these questions we need to document the origin of the data, and in the case of data acquired through manual downloads, we need to document the procedures we took to acquire the data to the best of our ability.

As discussed in Section 2.3.1, all acquired data should be accompanied by a data origin file. The majority of this information can typically be identified on the resource's website and/ or the resource's documentation. In the case of the CEDEL2 corpus, the corpus homepage provides most of the information we need.

Structurally, data documentation files should be stored close to the data they describe. So for our data origin file this means adding it to the *data/original/* directory. Naming the file in a transparent way is also important. I've named the file *cedel2_do.csv* to reflect the name of the corpus, the meaning of the file as data origin with *_do*, *and the file extension* .csv* to reflect the file format. CSV files reflect tabular content. It is not required that data origin files are tabular, but it makes it easier to read and display them in literate programming documents.

> **Tip**
> There are many ways to create and edit CSV files. You can use a spreadsheet program like MS Excel or Google Sheets, a text editor like Notepad or TextEdit, or a code editor like RStudio or VS Code. The `qtalrkit` package provides a convenient function, `create_data_origin()` to create a CSV file with the data origin boilerplate structure. This CSV file then can be edited to add the relevant information in any of the above mentioned programs.
> Using a spreadsheet program is the easiest method for editing tabular data. The key is to save the file as a CSV file, and not as an Excel file, to maintain our adherence to the principle of using open formats for reproducible research.

In Table 5.1, I've created a data origin file for the CEDEL2 corpus.

Given this is a manual download we also need to document the procedure used to retrieve the data in prose. The script in the *code/* directory that is typically used to acquire the data is not used to programmatically retrieve data in this case. However, to keep things predictable we will use this file to document the download procedure. I've created a Quarto file named *1_acquire_data.qmd* in the *code/* directory of my project.

A glimpse at the directory structure of the project at this point is seen in Example 5.2.

Example 5.2. Project structure for the CEDEL2 corpus data acquisition

Table 5.1: Data origin file for the CEDEL2 corpus

attribute	description
Resource name	CEDEL2: Corpus Escrito del Español como L2.
Data source	http://cedel2.learnercorpora.com/, https://doi.org/10.1177/02676583211050522
Data sampling frame	Corpus that contains samples of the language produced from learners of Spanish as
Data collection date(s)	2006-2020.
Data format	CSV file. Each row corresponds to a writing sample. Each column is an attribute of
Data schema	A CSV file for L2 learners and a CSV file for native speakers.
License	CC BY-NC-ND 3.0 ES
Attribution	Lozano, C. (2022). CEDEL2: Design, compilation and web interface of an online co

```
project/
├── code/
│   ├── 1_acquire_data.qmd
│   └── ...
├── data/
│   ├── analysis/
│   ├── derived/
│   └── original/
│       ├── cedel2_do.csv
│       └── cedel2/
│           ├── cedel2-l1-english-learners.csv
│           └── cedel2-native-spanish-speakers.csv
├── output/
│   ├── figures/
│   ├── reports/
│   ├── results/
│   └── tables/
├── README.md
└── _main.R
```

In the *1_acquire_data.qmd* file I've added example sections to display the data origin CSV file as a table and to document the data download procedures, as seen in File 5.1.

The output from *1_acquire_data.qmd* will contain a table displaying the data origin file and a prose section documenting the data acquisition process. This will provide a transparent record of the data acquisition process for future reference.

Manually downloading other resources will inevitably include unique processes for obtaining the data, but in the end the data should be archived in the research structure in the *data/o-*

riginal/ directory and documented in the appropriate places. The acquired data is treated as 'read-only', meaning it is not modified in any way. This gives us a transparent starting point for subsequent steps in the data preparation process.

5.1.2 Programmatic

There are many resources that provide corpus data that is directly accessible for which programmatic downloads can be applied. A **programmatic download** is a download in which the process can be automated through code. Thus, this is a reproducible process. The data can be acquired by anyone with access to the necessary code.

In this case, and subsquent data acquisition procedures in this chapter, we use the *1_acquire_data.qmd* Quarto file to its full potential intermingling prose, code, and code comments to execute and document the download procedure. In File 5.2, I've added example sections to display example boilerplate structure for a programmatic data acquisition and documentation.

To illustrate how this works to conduct a programmatic download, we will work with the Switchboard Dialog Act Corpus (SWDA) (University of Colorado Boulder 2008). The version that we will use is found on the Linguistic Data Consortium under the Switchboard-1 Release 2 Corpus[9]. The corpus and related documentation are linked on the catalog page https://catalog.ldc.upenn.edu/docs/LDC97S62/.

From the documentation we learn that the corpus contains transcripts for 1155 5-minute two-way telephone conversations among English speakers for all areas of the United States. The speakers were given a topic to discuss and the conversations were recorded. The corpus metadata and annotations for sociolinguistic and discourse features.

The SWDA was referred to in Section 3.2.3 to support our toy hypothesis that men and women differ in the frequency of the use of questions in spontaneous conversations. This corpus, as you can image, could support a wide range of interesting reseach questions. Let's assume we are following research conducted by Tottie (2011) to explore the use of filled pauses such as "um" and "uh" and traditional sociolinguistic variables such as sex, age, and education in spontaneous speech by American English speakers.

With this goal in mind, let's get started writing the code to download and organize the data in our project directory. First we need to identify the URL (Uniform Resource Locator) for the data that we want to download. More often than not this file will be some type of compressed archive file with an extension such as *.zip* (Zipped file), *.tar* (Tarball file), or *tar.gz* (Gzipped tarball file), which is the case for the SWDA corpus. Compressed files make downloading multiple files easy by grouping files and directories into one file.

[9] https://catalog.ldc.upenn.edu/LDC97S62

Consider this
You may be wondering what the difference betwen *.zip*, *.tar*, and *.tar.gz* files are. The *.zip* file format is the most common. It groups file and directories into one file (archives) and compresses them to reduce the size of the file in one step when the file is created.
The *.tar* file format is used archive files and folders, it does not perform compression. Gzipping peforms the compression to the *.tar* file resulting in a file with the *.tar.gz* extension. Notably the *.gz* compression is highly efficient for large files. Take the *swda.tar.gz* file for example. It has a compressed file size of 4.6 MB, but when uncompressed it is 16.9 MB. This is a 73% reduction in file size.

In R we can use the `download.file()` function from base R. The `download.file()` function minimally requires two arguments: `url` and `destfile`. These correspond to the file to download and the location where it is to be saved to disk. To break out the process a bit, I will assign the URL and destination file path to variables and then use the `download.file()` function to download the file.

Example 5.3.

```
# URL to SWDA corpus compressed file
file_url <-
  "https://catalog.ldc.upenn.edu/docs/LDC97S62/swb1_dialogact_annot.tar.gz"

# Relative path to project/data/original directory
file_path <- "../data/original/swda.tar.gz"

# Download SWDA corpus compressed file
download.file(url = file_url, destfile = file_path)
```

Warning
Note that the `file_path` variable in Example 5.3 is a relative path to the *data/original/* directory. This is because the *1_acquire_data.qmd* file that we are using for this code is located in the *code/* directory and the *data/* directory is a sibling directory to the *code/* directory.
It is also possible to use an absolute path to the *data/original/* directory. I will have more to say about the advantages and disadvantages of relative and absolute paths in reproducible research in Chapter 12.

As we can see looking at the directory structure, in Example 5.4, the *swda.tar.zip* file has been added to the *data/original/* directory.

Example 5.4. Downloaded SWDA corpus compressed file

```
data/
├── analysis/
├── derived/
└── original/
    └── swda.tar.zip
```

Once an compressed file is downloaded, however, the file needs to be 'decompressed' to reveal the directory structure and files. To decompress this *.tar.gz* file we use the `untar()` function with the arguments `tarfile` pointing to the *.tar.gz* file and `exdir` specifying the directory where we want the files to be extracted to. Again, I will assign the arguments to variables. Then we can decompress the file using the `untar()` function.

Example 5.5.

```
# Relative path to the compressed file
tar_file <- "../data/original/swda.tar.gz"

# Relative path to the directory to extract to
extract_to_dir <- "../data/original/swda/"

# Decompress .zip file and extract to our target directory
untar(tar_file, extract_to_dir)
```

The directory structure of *data/* in Example 5.6 now shows the *swda.tar.gz* file and the *swda* directory that contains the decompressed directories and files.

Example 5.6.

```
data/
├── analysis/
├── derived/
└── original/
    ├── swda/
    │   ├── README
    │   ├── doc/
    │   ├── sw00utt/
    │   ├── sw01utt/
    │   ├── sw02utt/
    │   ├── sw03utt/
    │   ├── sw04utt/
```

```
    |   ├── sw05utt/
    |   ├── sw06utt/
    |   ├── sw07utt/
    |   ├── sw08utt/
    |   ├── sw09utt/
    |   ├── sw10utt/
    |   ├── sw11utt/
    |   ├── sw12utt/
    |   └── sw13utt/
    └── swda.tar.gz
```

At this point we have acquired the data programmatically and with this code as part of our workflow anyone could run this code and reproduce the same results. The code as it is, however, is not ideally efficient. Firstly the *swda.tar.gz* file is not strictly needed after we decompress it and it occupies disk space if we keep it. And second, each time we run this code the file will be downloaded from the remote server. This leads to unnecessary data transfer and server traffic and will overwrite the data if it already exists in our project directory which could be problematic if the data changes on the remote server. Let's tackle each of these issues in turn.

To avoid writing the *swda.tar.gz* file to disk (long-term) we can use the `tempfile()` function to open a temporary holding space for the file in the computing environment. This space can then be used to store the file, decompress it, and then the temporary file will automatically be deleted. We assign the temporary space to an R object we will name `temp_file` with the `tempfile()` function. This object can now be used as the value of the argument `destfile` in the `download.file()` function.

Example 5.7.

```
# URL to SWDA corpus compressed file
file_url <-
  "https://catalog.ldc.upenn.edu/docs/LDC97S62/swb1_dialogact_annot.tar.gz"

# Create a temporary file space for our .tar.gz file
temp_file <- tempfile()

# Download SWDA corpus compressed file
download.file(file_url, temp_file)
```

Tip
In Example 5.7, I've used the values stored in the objects `file_url` and `temp_file` in the `download.file()` function without specifying the argument names –only providing the names of the objects. R will assume that values of a function map to the ordering of the arguments. If your values do not map to ordering of the arguments you are required to specify the argument name and the value. To view the ordering of objects hit `tab` after entering the function name or consult the function documentation by prefixing the function name with `?` and hitting `enter`.

At this point our downloaded file is stored temporarily on disk and can be accessed and decompressed to our target directory using *temp_file* as the value for the argument `tarfile` from the `untar()` function. I've assigned our target directory path to `extract_to_dir` and used it as the value for the argument `exdir`.

Example 5.8.

```
# Assign our target directory to `extract_to_dir`
extract_to_dir <- "../data/original/swda/"

# Decompress .tar.gz file and extract to our target directory
untar(tarfile = temp_file, exdir = target_dir)
```

Our directory structure in Example 5.8 is the same as in Example 5.6, minus the *swda.tar.gz* file.

The second issue I raised concerns the fact that running this code as part of our project will repeat the download each time. Since we would like to be good citizens and avoid unnecessary traffic on the web and avoid potential issues in overwriting data, it would be nice if our code checked to see if we already have the data on disk and if it exists, then skip the download, if not then download it.

The desired functionality we've described can be implemented using the `if()` function. The `if()` function is one of a class of functions known as control statements. **Control statments** allow us to control the flow of our code by evaluating logical statements and processing subsequent code based on the logical value it is passed as an argument.

So in this case we want to evaluate whether the data directory exists on disk. If it does then skip the download, if not, proceed with the download. In combination with `else` which provides the 'if not' part of the statement, we have the following logical flow in Example 5.9.

Example 5.9.

```
if (DIRECTORY_EXISTS) {
  # Do nothing
} else {
  # Download data
}
```

We can simplify this statement by using the `!` operator which negates the logical value of the statement it precedes. So if the directory exists, `!DIRECTORY_EXISTS` will return `FALSE` and if the directory does not exist, `!DIRECTORY_EXISTS` will return `TRUE`. In other words, if the directory does not exist, download the data. This is shown in Example 5.10.

Example 5.10.

```
if (!DIRECTORY_EXISTS) {
  # Download data
}
```

Now, to determine if a directory exists in our project directory we will turn to the `fs` package (Hester, Wickham, and Csárdi 2023). The `fs` package provides a set of functions for interacting with the file system, including `dir_exists()`. `dir_exists()` takes a path to a directory as an argument and returns the logical value, `TRUE`, if that directory exists, and `FALSE` if it does not.

We can use this function to evaluate whether the directory exists and then use the `if()` function to process the subsequent code based on the logical flow we set out in Example 5.10. Applied to our project, the code will look like Example 5.11.

Example 5.11.

```
# Load the `fs` package
library(fs)

# URL to SWDA corpus compressed file
file_url <-
  "https://catalog.ldc.upenn.edu/docs/LDC97S62/swb1_dialogact_annot.tar.gz"

# Create a temporary file space for our .tar.gz file
temp_file <- tempfile()

# Assign our target directory to `extract_to_dir`
```

```
extract_to_dir <- "../data/original/swda/"

# Check if our target directory exists
# If it does not exist, download the file and extract it
if (!dir_exists(extract_to_dir)) {
  # Download SWDA corpus compressed file
  download.file(file_url, temp_file)

  # Decompress .tar.gz file and extract to our target directory
  untar(tarfile = temp_file, exdir = extract_to_dir)
}
```

The code in Example 5.11 is added to the *1_acquire_data.qmd* file we introduced in File 5.2. When this file is run, the SWDA corpus data will be downloaded and extracted to our project directory. If the data already exists, the download will be skipped, just as we wanted.

Before we move on, we need to make sure to create and add the appropriate information to the data origin file. To make this easier, the `qtalrkit` package includes a function, `create_data_origin()`, to create a data origin file template in CSV format. This function takes the path for the desired file. In the SWDA Corpus case, this might be something like: *../data/original/swda_do.csv*. The function only needs to be run once and does not need to be part of the reproducible workflow.

Running the code in Example 5.12 at the console will create the file. Open it in your preferred text or spreadsheet editor to add the appropriate information.

Example 5.12.

```
# Load the `qtalrkit` package
library(qtalrkit)

# Create a data origin file template
create_data_origin("../data/original/swda_do.csv")
```

Our complete project structure for the SWDA corpus data acquisition is shown in Example 5.13.

Example 5.13. Project structure for the SWDA corpus data acquisition

```
project/
├── code/
```

```
│   ├── 1_acquire_data.qmd
│   └── ...
├── data/
│   ├── analysis/
│   ├── derived/
│   └── original/
│       ├── swda_do.csv
│       └── swda/
│           ├── README
│           ├── doc/
│           ├── sw00utt/
│           ├── sw01utt/
│           ├── sw02utt/
│           ├── sw03utt/
│           ├── sw04utt/
│           ├── sw05utt/
│           ├── sw06utt/
│           ├── sw07utt/
│           ├── sw08utt/
│           ├── sw09utt/
│           ├── sw10utt/
│           ├── sw11utt/
│           ├── sw12utt/
│           └── sw13utt/
├── output/
│   ├── figures/
│   ├── reports/
│   ├── results/
│   └── tables/
├── README.md
└── _main.R
```

Great, we've successfully acquired and documented the SWDA Corpus data. We've leveraged R to automate the download and extraction of the data, depending on the existence of the data in our project directory. But you may be asking yourself, "Can't I just navigate to the corpus page and download the data manually myself?" The simple answer is, "Yes, you can." The more nuanced answer is, "Yes, but consider the trade-offs."

The following scenarios highlight the some advantages to automating the process. If you are acquiring data from multiple files, it can become tedious to document the manual process for each file such that it is reproducible. It's possible, but it's error prone. Now, if you are collaborating with others, you will want to share this data with them. It is very common

to find data that has limited restrictions for use in academic projects, but the most common limitation is redistribution. This means that you can use the data for your own research, but you cannot share it with others. If you plan on publishing your project to a repository, like GitHub, to share the data as part of your reproducible project, you would be violating the terms of use for the data. By including the programmatic download in your project, you can ensure that your collaborators can easily and effectively acquire the data themselves and that you are not violating the terms of use.

5.2 APIs

A convenient alternative method for acquiring data in R is through package interfaces to web services. These interfaces are built using R code to make connections with resources on the web through **Application Programming Interfaces** (APIs). Websites such as Project Gutenberg, Twitter, Facebook, and many others provide APIs to allow access to their data under certain conditions, some more limiting for data collection than others. Programmers (like you!) in the R community take up the task of wrapping calls to an API with R code to make accessing that data from R convenient, and of course reproducible.

> **Dive deeper**
> Many, many web services provide API access. These APIs span all kinds of data, from text to images to video to audio. Visit the Public APIs website[a] to explore the diversity of APIs available.
> ROpenSci maintains a curated list of R packages that provide access to data from web services. Visit the ROpenSci website[b] to explore the packages available.
>
> [a] https://publicapis.io/
> [b] https://ropensci.org/packages/data-access/

In addition to popular public APIs, there are also APIs that provide access to repositories and databases which are of particular interest to linguists. For example, Wordbank[10] provides access to a large collection of child language corpora through the `wordbankr` package (Braginsky 2022), and Glottolog[11], World Atlas of Language Structures[12] (WALS), and PHOIBLE[13] provide access to large collections of language metadata that can be accessed through the `lingtypology` package (Moroz 2023).

Let's work with an R package that provides access to the TalkBank[14] database. The TalkBank project [CITATION] contains a large collection of spoken language corpora from

[10] http://wordbank.stanford.edu/
[11] https://glottolog.org/
[12] https://wals.info/
[13] https://phoible.org/
[14] https://talkbank.org/

various contexts: conversation, child language, multilinguals, *etc.* Resource information, web interfaces, and links to download data in various formats can be found by perusing individual resources linked from the main page. However, the `TBDBr` package (Kowalski and Cavanaugh 2022) provides convenient access to data using R once a data resource is identified.

The CABNC (Albert, de Ruiter, and de Ruiter 2015) contains the demographically sampled portion[15] of the spoken portion of the British National Corpus (BNC) (Leech 1992).

Useful for a study aiming to research spoken British English, either in isoloation or in comparison to American English (SWDA).

First, we need to install and load the `TBDBr` package. The `install.package()` and `library()` functions work just great for this. An alternative is to use the `pacman` package (Rinker and Kurkiewicz 2019) to install and load the package in one step with the function `p_load()`, as shown in Example 5.14.

Example 5.14.

```
# Install and load the TBDBr package
pacman::p_load(TBDBr)
```

The `TBDBr` package provides a set of common `get*()` functions for acquiring data from the TalkBank corpus resources. These include:

- `getParticipants()`
- `getTranscripts()`
- `getTokens()`
- `getTokenTypes()`
- `getUtterances()`

For each of these function the first argument is `corpusName`, which is the name of the corpus resource as it appears in the TalkBank database. The second argument is `corpora`, which takes a character vector describing the path to the data. For the CABNC, these arguments are `"ca"` and `c("ca", "CABNC")` respectively. To determine these values, TBDBr provides the `getLegalValues()` interactive function which allows you to interactively select the repository name, corpus name, and transcript name (if necessary).

Another important aspect of these function is that they return data frame objects. Since we are accessing data that is in a structured database, this makes sense. However, we should always check the documentation for the object type that is returned by function to be aware of how to work with the data.

Let's start by retrieving the utterance data for the CABNC and preview the data frame it returns using `glimpse()`.

[15] http://www.natcorp.ox.ac.uk/docs/URG/BNCdes.html#body.1_div.1_div.5_div.1

Example 5.15.

```
# Set corpus_name and corpus_path
corpus_name <- "ca"
corpus_path <- c("ca", "CABNC")

# Get utterance data
utterances <-
  getUtterances(
    corpusName = corpus_name,
    corpora = corpus_path)
```

```
> [1] "Fetching data, please wait..."
> [1] "Success!"
```

```
# Preview the data
glimpse(utterances)
```

```
> Rows: 235,901
> Columns: 10
> $ filename  <list> "KB0RE000", "KB0RE000", "KB0RE000", "KB0RE000", "KB0RE000",~
> $ path      <list> "ca/CABNC/KB0/KB0RE000", "ca/CABNC/KB0/KB0RE000", "ca/CABNC~
> $ utt_num   <list> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 1~
> $ who       <list> "PS002", "PS006", "PS002", "PS006", "PS002", "PS006", "PS00~
> $ role      <list> "Unidentified", "Unidentified", "Unidentified", "Unidentifi~
> $ postcodes <list> <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NU~
> $ gems      <list> <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NULL>, <NU~
> $ utterance <list> "You enjoyed yourself in America", "Eh", "did you", "Oh I c~
> $ startTime <list> "0.208", "2.656", "2.896", "3.328", "5.088", "6.208", "8.32~
> $ endTime   <list> "2.672", "2.896", "3.328", "5.264", "6.016", "8.496", "9.31~
```

Inspecting the output from Example 5.15, we see that the data frame contains 235901 observations and 10 variables.

The summary provided by `glimpse()` also provides other useful information. First, we see the data type of each variable. Interestingly, the data type for each variable in the data frame is `list`. Being that a list is two-dimensional data type, like a data frame, we have list-type data in each value. This is known as a **nested structure**. We will see, and create, nested structures later, but for now it will suffice to say that we would like to 'unnest' these lists and reveal the list-contained vector types at the data frame level.

To do this we will pass the `utterances` data frame to the `unnest()` function from the `tidyr` package (Wickham, Vaughan, and Girlich 2023). `unnest()` takes a data frame and a vector of variable names to unnest, `cols = c()`. To unnest all variables, we will use the `everything()` function from `dplyr` (Wickham, François, et al. 2023) to select all variables. We will use the result to overwrite the `utterances` object with the unnested data frame.

Example 5.16.

```
# Unnest the data frame
utterances <-
  utterances |>
  unnest(cols = everything())

# Preview the data
glimpse(utterances)
```

```
> Rows: 235,901
> Columns: 10
> $ filename  <chr> "KB0RE000", "KB0RE000", "KB0RE000", "KB0RE000", "KB0RE000", ~
> $ path      <chr> "ca/CABNC/KB0/KB0RE000", "ca/CABNC/KB0/KB0RE000", "ca/CABNC/~
> $ utt_num   <dbl> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17~
> $ who       <chr> "PS002", "PS006", "PS002", "PS006", "PS002", "PS006", "PS002~
> $ role      <chr> "Unidentified", "Unidentified", "Unidentified", "Unidentifie~
> $ postcodes <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
> $ gems      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
> $ utterance <chr> "You enjoyed yourself in America", "Eh", "did you", "Oh I co~
> $ startTime <chr> "0.208", "2.656", "2.896", "3.328", "5.088", "6.208", "8.32"~
> $ endTime   <chr> "2.672", "2.896", "3.328", "5.264", "6.016", "8.496", "9.312~
```

The output from Example 5.16 shows that the variables are now one-dimensional vector types.

Returning to the information about our data frame from `glimpse()`, the second thing to notice is we get a short preview of the values for each variable. There are a couple things we can gleen from this. One is that we can confirm or clarify the meaning of the variable names by looking at the values. The other thing to consider is whether the values show any patterns that may be worthy of more scrutiny. For example, various variables appear to contain the same values for each observation. For a variable like `filename`, this is expected as the first values likely correspond to the same file. However, for the variables `postcodes` and `gems` the values are 'NA'. This suggests that these variables may not contain any useful information and we may want to remove them later.

For now, however, we want to acquire and store the data in its original form (or as closely as possible). So now, we have acquired the utterances data and have it in our R session as a data frame. To store this data in a file, we will first need to consider the file format. Data frames are tabular, so that gives us a few options. Since we are working in R, we could store this data as an R object, in the form of an RDS file. An RDS file is a binary file that can be read back into R as an R object. This is a good option if we want to store the data for use in R, but not if we want to share the data with others or use it in other software. Another option is to store the data as a spreadsheet file, such as XSLX (MS Excel). This may make viewing and editing the contents more convenient, but it depends on the software available to you and others. A third, more viable option, is to store the data as a CSV file. CSV files are plain text files that can be read and written by most software. This makes CSV files one of the most popular for sharing tablular data. For this reason, we will store the data as a CSV file.

The `readr` package (Wickham, Hester, and Bryan 2023) provides the `write_csv()` function for writing data frames to CSV files. The first argument is the data frame to write, and the second argument is the path to the file to write. Note, however, that the directories in the path we specify need to exist. If they do not, we will get an error. In this case, I would like to write the file *utterances.csv* to the *../data/original/cabnc/* directory. The original project structure does not contain a *cabnc/* directory, so I need to create one. To do this, I will use `dir_create()` from the `fs` package (Hester, Wickham, and Csárdi 2023).

Example 5.17.

```
# Create the target directory
dir_create("../data/original/cabnc/")

# Write the data frame to a CSV file
write_csv(utterances, "../data/original/cabnc/utterances.csv")
```

Chaining the steps covered in Examples 5.15, 5.16, and 5.17, we have a succinct and legible code to acquire, adjust, and write utterances from the CABNC in Example 5.18.

Example 5.18.

```
# Set corpus_name and corpus_path
corpus_name <- "ca"
corpus_path <- c("ca", "CABNC")

# Create the target directory
dir_create("../data/original/cabnc/")

# Get utterance data
```

```
getUtterances(
  corpusName = corpus_name,
  corpora = corpus_path
) |>
  unnest(cols = everything()) |>
  write_csv("../data/original/cabnc/utterances.csv")
```

If our goal is just to acquire utterances, then we are done acquiring data and we move on to the next step. However, if we want to acquire other datasets from the CABNC, say participants, tokens, *etc.*, then we can either repeat the steps in Example 5.18 for each data type, or we can write a function to do this for us. A function serves us to make our code more legible and reusable for the CABNC, and since the TalkBank data is structured similarly across corpora, we can also use the function to acquire data from other corpora, if need be.

To write a function, we need to consider the following:

1. What is the name of the function?
2. What arguments does the function take?
3. What functionality does the function provide?
4. Does the function have optional arguments?
5. How does the function return the results?

Taking each in turn, the name of the function should be descriptive of what the function does. In this case, we are acquiring and writing data from Talkbank corpora. A possible name is `get_talkbank_data()`. The required arguments of the the `get*()` functions will definitely figure in our function. In addition, we will need to specify the path to the directory to write the data.

Example 5.19.

```
get_talkbank_data <- function(corpus_name, corpus_path, target_dir) {
  # ...
}
```

The next thing to consider is what functionality the function provides. In this case, we want to acquire and write data from Talkbank corpora. We can start by leveraging the code steps in Example 5.18, making some adjustments to the code replacing the hard-coded values with the function arguments and adding code to create the target file name based on the `target_dir` argument.

Example 5.20.

```
get_talkbank_data <- function(corpus_name, corpus_path, target_dir) {

  # Create the target directory
  dir_create(target_dir)

  # Set up file path name
  utterances_file  <- path(target_dir, "utterances.csv")

  # Acquire data and write to file
  getUtterances(corpusName = corpus_name, corpora = corpus_path) |>
    unnest(cols = everything()) |>
    write_csv(utterances_file)
}
```

Before we address the obvious feature missing, which is the fact that this function in Example 5.20 only acquires and writes data for utterances, let's consider some functionality which would make this function more user-friendly.

What if the data is already acquired? Do we want to overwrite it, or should the function skip the process for files that already exist? By skipping the process, we can save time and computing resources. If the files are periodically updated, then we might want to overwrite existing files. To achieve this functionality we will use an `if()` statement to check if the file exists. If it does, then we will skip the process. If it does not, then we will acquire and write the data.

Example 5.21.

```
get_talkbank_data <- function(corpus_name, corpus_path, target_dir) {

  # Create the target directory
  dir_create(target_dir)

  # Set up file path name
  utterances_file  <- path(target_dir, "utterances.csv")

  # If the file does not exist, then...
  # Acquire data and write to file
  if(!file_exists(utterances_file)) {
    getUtterances(corpusName = corpus_name, corpora = corpus_path) |>
      unnest(cols = everything()) |>
      write_csv(utterances_file)
```

```
  }
}
```

We can also add functionality to Example 5.21 to force overwrite existing files, if need be. To do this, we will add an optional argument to the function, `force`, which will be a logical value. We will set the default to `force = FALSE` to preserve the existing functionality. If `force = TRUE`, then we will overwrite existing files. Then we add another condition to the `if()` statement to check if `force = TRUE`. If it is, then we will overwrite existing files.

Example 5.22.

```
get_talkbank_data <- function(corpus_name, corpus_path, target_dir, force =
↪  FALSE) {

  # Create the target directory
  dir_create(target_dir)

  # Set up file path name
  utterances_file  <- path(target_dir, "utterances.csv")

  # If the file does not exist, then...
  # Acquire data and write to file
  if(!file_exists(utterances_file) | force) {
    getUtterances(corpusName = corpus_name, corpora = corpus_path) |>
      unnest(cols = everything()) |>
      write_csv(utterances_file)
  }
}
```

From this point, we add the functionality to acquire and write the other data available from Talkbank corpora, such as participants, tokens, *etc.* This involves adding additional file path names and `if()` statements to check if the files exist surrounding the processing steps to Example 5.22. It may be helpful to perform other input checks, print messages, *etc.* for functions that we plan to share with others. I will leave these enhancements as an exercise for the reader.

Before we leave the topic of functions, let's consider where to put functions after we write them. Here are a few options:

1. In the same script as the code that uses the function.
2. In a separate script, such as *functions.R*.

3. In a package, which is loaded by the script that uses the function.

The general heuristic for choosing where to put functions is to put them in the same script as the code that uses them if the function is only used in that script. If the function is used in multiple scripts or the function or number of functions clutters the readability of the code, then put it in a separate script. If the function is used in multiple projects, then put it in an R package.

> **Dive deeper**
> If you are interested in learning more about writing functions, check out the Writing Functions chapter[a] in the R for Data Science[b] book.
> If you find yourself writing functions that are useful for multiple projects, you may want to consider creating an R package. R packages are a great way to share your code with others. If you are interested in learning more about creating R packages, check out the R Packages book[c] by Hadley Wickham and Jenny Bryan.
>
> [a] https://r4ds.had.co.nz/functions.html
> [b] https://r4ds.had.co.nz/
> [c] https://r-pkgs.org/

In this case, we will put the function in a separate file, *functions.R*, in the same directory as the other code files as in Example 5.25.

Example 5.23.

```
code/
|   ├── 1_acquire_data.qmd
|   ├── ...
|   └── functions.R
```

> **Tip**
> Note that that the *functions.R* file is an R script, not a Quarto document. Therefore code blocks that are used in *.qmd* files are not used, only the R code and code comments.

To include this, or other functions in in the R session of the code file that uses them, use the `source()` function, as seen in Example 5.24.

Example 5.24.

```
# Source functions
source("functions.R")
```

Given the utility of this function to my projects and potentially others', I've included the `get_talkbank_data()` function in the `qtalrkit` package. You can view the source code by calling the function without parentheses `()`, or on the `qtalrkit` GitHub repository.

After running the `get_talkbank_data()` function, we can see that the data has been acquired and written to the *data/original/cabnc/* directory.

Example 5.25.

```
data/
├── analysis
├── derived
└── original
    └── cabnc
        ├── participants.csv
        ├── token_types.csv
        ├── tokens.csv
        ├── transcripts.csv
        └── utterances.csv
```

Add comments to your code in *1-acquire-data.Rmd* and create and complete the data origin documentation file for this resource, and the acquisition is complete.

5.3 Web scraping

There are many resources available through downloads and APIs. There are, however, cases in which you want to acquire data from the public-facing web. R can be used to access the web programmatically through a process known as web scraping. The complexity of web scrapes can vary but in general it requires more advanced knowledge of R as well as the structure of the language of the web: HTML (Hypertext Markup Language).

HTML: language of the web

HTML is a cousin of XML (eXtensible Markup Language) and as such organizes web documents in a hierarchical format that is read by your browser as you navigate the web. Take for example the toy webpage I created as a demonstration in Figure 5.3.

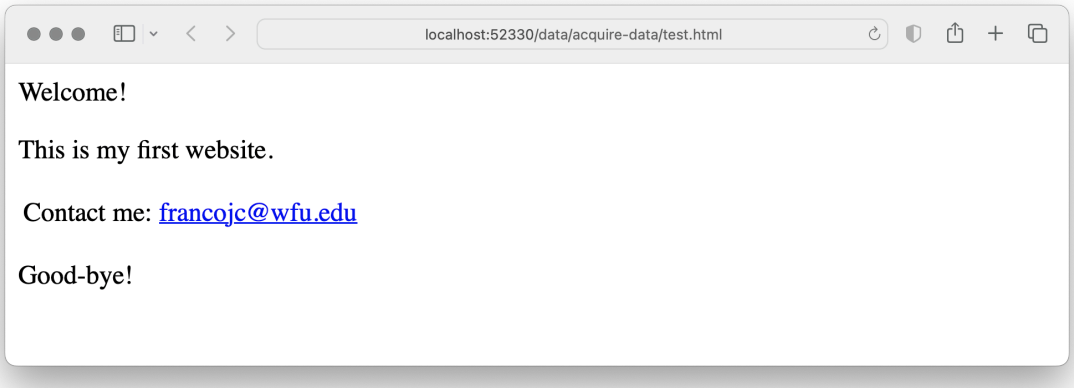


Figure 5.3: Example web page.

The file accessed by my browser to render this webpage is `test.html` and in plain-text format as seen in Example 5.26.

Example 5.26.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"
 ↪  "http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>My website</title>
  </head>
  <body>
    <div class="intro">
      <p>Welcome!</p>
      <p>This is my first website.</p>
    </div>
    <table>
      <tr>
        <td>Contact me:</td>
        <td>
          <a href="mailto:francojc@wfu.edu">francojc@wfu.edu</a>
        </td>
      </tr>
    </table>
```

```
    <div class="conc">
      <p>Good-bye!</p>
    </div>
  </body>
</html>
```

Each element in this file is delineated by an opening and closing tag, `<head></head>`. Tags are nested within other tags to create the structural hierarchy. Tags can take class and id labels to distinguish them from other tags and often contain other attributes that dictate how the tag is to behave when rendered visually by a browser. For example, there are two `<div>` tags in our toy example: one has the label `class = "intro"` and the other `class = "conc"`. `<div>` tags are often used to separate sections of a webpage that may require special visual formatting. The `<a>` tag, on the other hand, creates a web link. As part of this tag's function, it requires the attribute `href=` and a web protocol –in this case it is a link to an email address `mailto:francojc@wfu.edu`. More often than not, however, the `href=` contains a URL (Uniform Resource Locator). A working example might look like this: `<a href="https://francojc.github.io/">My homepage</a>`.

The aim of a web scrape is to download the HTML file(s) that contain the data we are interested in. This will include more information that we may ultimately need, but by downloading the raw source HTML we are effectively creating a local archive, or copy, of the webpage. Thus, if the webpage is updated or removed from the web, we will still have access to the data we accessed.

Later in the curation process we will parse (*i.e.* read and extract) target information that is relevant for the research at hand. However, it often useful to parse the raw HTML in the process of acquiring data if we are interested in harvesting data from multiple pages and we would like to use the HTML structure to guide our data extraction (*i.e.* URLs to other pages)

To provide some preliminary background on working with HTML, we will use the toy example above to demonstrate how to read and parse HTML using R. To do this we will use the rvest[16](Wickham 2022) package. First, install/load the package, then, read and parse the HTML from the character vector named `web_file` assigning the result to `html`.

Example 5.27.

```
# Load package
library(rvest) # read and parse HTML
```

[16]https://CRAN.R-project.org/package=rvest

```
html <- read_html(web_file) # retrieve raw html
html
```

```
> {html_document}
> <html>
> [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8 ...
> [2] <body>\n    <div class="intro">\n      <p>Welcome!</p>\n      <p>This is  ...
```

In Example 5.27 `read_html()` retrieves the raw HTML and it makes it accessible to parsing in R. Being a subtype of XML, `read_html()` converts the raw HTML into an object of class `xml_document`, as we can see by calling `class()` on the `html` object in Example 5.28.

Example 5.28.

```
class(html)
```

```
> [1] "xml_document" "xml_node"
```

An object of class `xml_document` represents each HTML tag as a node. The tag nodes are elements can be accessed by using the `html_elements()` function by specifying the tag/node/element to isolate.

Example 5.29.

```
html |>
  html_elements("div")
```

```
> {xml_nodeset (2)}
> [1] <div class="intro">\n      <p>Welcome!</p>\n      <p>This is my first web ...
> [2] <div class="conc">\n      <p>Good-bye!</p>\n    </div>
```

Notice that the output of Example 5.29 has returned both `div` tags and their respective children, tags contained within. To isolate one of tags by its class, we add the class name to the tag separating it with a `.`.

Example 5.30.

```
  html |>
    html_elements("div.intro")
```

```
> {xml_nodeset (1)}
> [1] <div class="intro">\n      <p>Welcome!</p>\n      <p>This is my first web ...
```

Great. Now say we want to drill down and isolate the subordinate `<p>` nodes. We can add `p` to our node filter, as in Example 5.31.

Example 5.31.

```
  html |>
    html_elements("div.intro p")
```

```
> {xml_nodeset (2)}
> [1] <p>Welcome!</p>
> [2] <p>This is my first website. </p>
```

To extract the text contained within a node we use the `html_text()` function.

Example 5.32.

```
  html |>
    html_elements("div.intro p") |>
    html_text()
```

```
> [1] "Welcome!"                   "This is my first website. "
```

The result of Example 5.32 is a character vector with two elements corresponding to the text contained in each `<p>` tag. If you were paying close attention you might have noticed that the second element in our vector includes extra whitespace after the period. To trim leading and trailing whitespace from text we can add the `trim = TRUE` argument to `html_text()`, as in Example 5.33.

Example 5.33.

```
html |>
  html_elements("div.intro p") |>
  html_text(trim = TRUE)
```

```
> [1] "Welcome!"                    "This is my first website."
```

With this basic understanding of how to read and parse HTML, we can now turn to a more realistic example.

Federalist Papers

Say we investigate the authorship question of the the Federalist Papers following in the footsteps of Mosteller and Wallace (1963). We want to scrape the text of the Federalist Papers from the Library of Congress website. The main page for the Federalist Papers is located at *https://guides.loc.gov/federalist-papers/full-text* and can be seen in Figure 5.4.

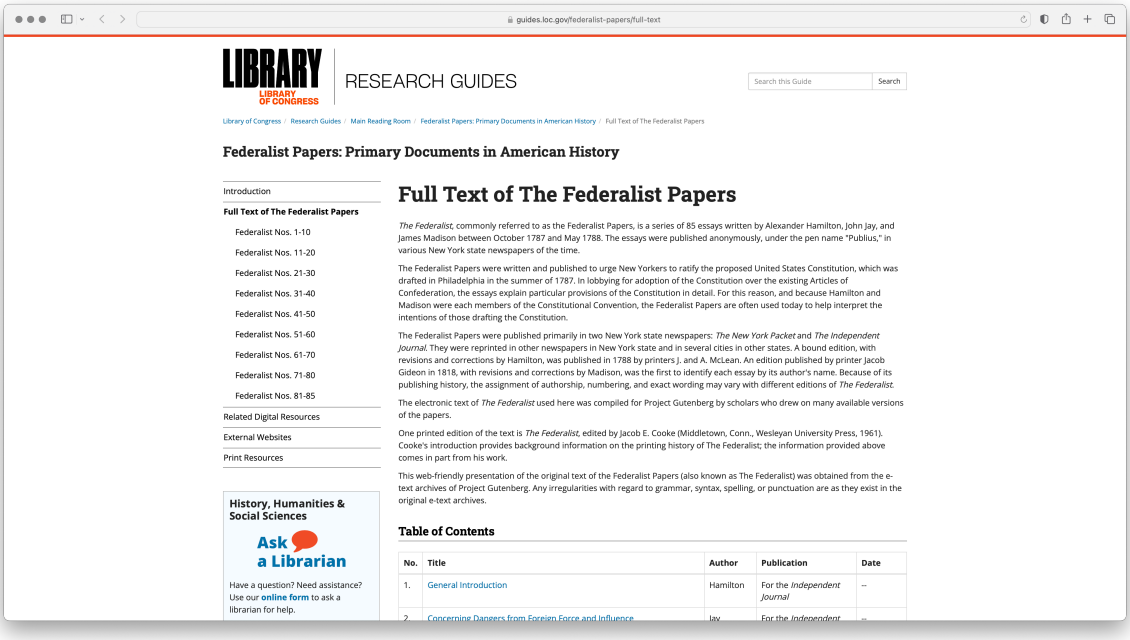


Figure 5.4: Screenshot of the Library of Congress website for the Federalist Papers

The main page contains links to the text of each of the 85 papers. Our goal will be to scrape and archive the raw HTML for this page and then parse the HTML to extract the links to each of the papers. We will then scrape and archive the raw HTML for each of the 85 papers.

The first step in any web scrape is to investigate the site and page(s) we want to scrape to ascertain if there any licensing restrictions. Many, but not all websites, will include a plain text file `robots.txt`[17] at the root of the main URL. This file is declares which webpages a 'robot' (including web scraping scripts) can and cannot access. We can use the `robotstxt` package to find out which URLs are accessible [18].

Example 5.34.

```
# Install and load package
pacman::p_load(robotstxt)

# URL for the Federalist Papers (LOC)
url <- "https://guides.loc.gov/federalist-papers/full-text"

# Check permissions
paths_allowed(url)
```

```
> [1] TRUE
```

The next step is to read and parse the raw HTML. We can do this using the `read_html()` function.

Example 5.35.

```
# Read raw html and parse to xml
html <- read_html(url)

# Preview html
html
```

```
> {html_document}
> <html lang="en">
> [1] <head>\n<meta http-equiv="X-UA-Compatible" content="IE=Edge">\n<meta http ...
> [2] <body class="s-lg-guide-body">\r\n<a id="s-lg-public-skiplink" class="ale ...
```

At this point we have captured the raw HTML assigning it to the object named `html`. Let's archive the raw HTML to a file in our project directory. We can do this using the `write_html()` function from the `xml2` package (Wickham, Hester, and Ooms 2023).

[17] https://www.cloudflare.com/learning/bots/what-is-robots.txt/

[18] It is important to check the paths of sub-domains as some website allow access in some areas and not in others

Example 5.36.

```
# Create directory for HTML files
dir_create("../data/original/federalist_papers/")

# Write raw html to file
write_html(html, "../data/original/federalist_papers/main.html")
```

Our update project directory structure can be seen in Example 5.37.

Example 5.37.

```
data/
|── analysis/
├── derived/
└── original/
    └── federalist_papers/
        └── main.html
```

Now, we also want to scrape the HTML that contains of the pages corresponding to the 85 Federalist Papers. Perusing the main page we can see that papers are organized into nine groups, *e.g.* "Federalist Nos. 1-10". So our aim will be to scrape the HTML for each of these nine pages. We can do this using the `rvest` package, but we need to identify the HTML elements that contain the URLs first in the main webpage we have in `html`.

To do this it is helpful to use a browser to inspect specific elements of the webpage, much as we did in the toy example in Example 5.26. To view the raw and displayed HTML, your browser will be equipped with a command that you can enable by hovering your mouse over the element of the page you want to target and using a right click to select "Inspect" (Chrome) or "Inspect Element" (Safari, Brave). This will split your browser window vertical or horizontally showing you the displayed and raw HTML underlying the webpage.

We can see the HTML elements that contain the URLs in Figure 5.5.

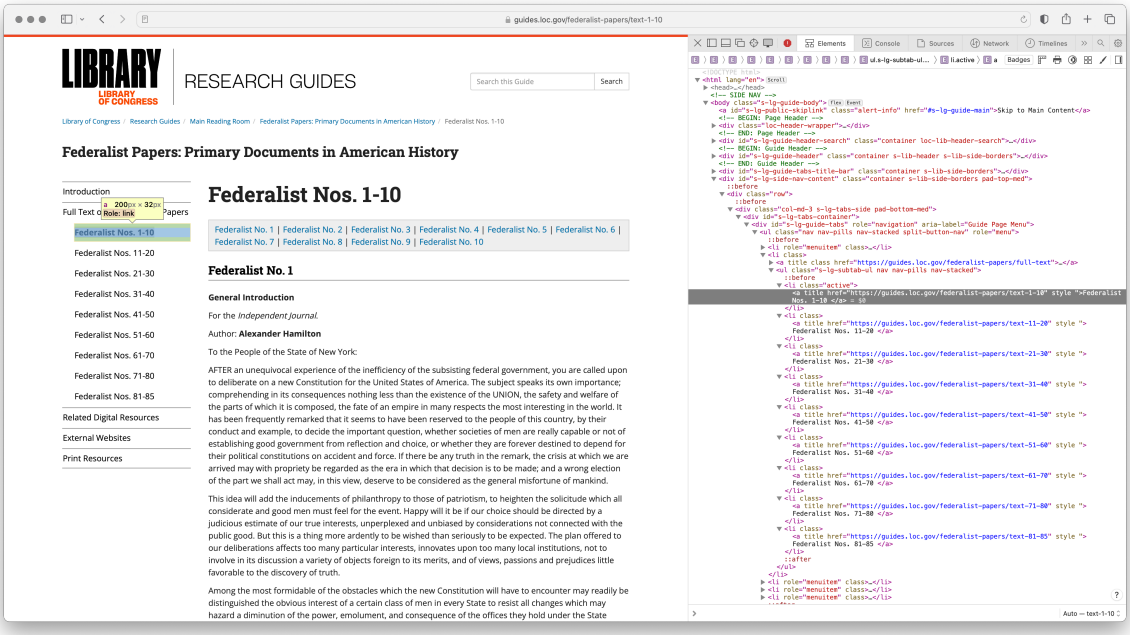


Figure 5.5: Screenshot of the HTML elements containing the URLs for the Federalist Papers

Let's take a closer look at the source HTML in Figure 5.6 so we can inspect the elements that contain the URLs and divise a strategy for isolating them to be extracted.

```
▼ <div class="col-md-3 s-lg-tabs-side pad-bottom-med">
  ▼ <div id="s-lg-tabs-container">
    ▼ <div id="s-lg-guide-tabs" role="navigation" aria-label="Guide Page Menu">
      ▼ <ul class="nav nav-pills nav-stacked split-button-nav" role="menu">
          ::before
        ▶ <li role="menuitem" class>…</li>
        ▼ <li class>
          ▶ <a title class href="https://guides.loc.gov/federalist-papers/full-text">…</a>
          ▼ <ul class="s-lg-subtab-ul nav nav-pills nav-stacked">
              ::before
            ▼ <li class="active">
                <a title href="https://guides.loc.gov/federalist-papers/text-1-10" style ">Federalist
                Nos. 1-10 </a> = $0
              </li>
            ▼ <li class>
                <a title href="https://guides.loc.gov/federalist-papers/text-11-20" style ">
                Federalist Nos. 11-20 </a>
              </li>
            ▼ <li class>
                <a title href="https://guides.loc.gov/federalist-papers/text-21-30" style ">
                Federalist Nos. 21-30 </a>
              </li>
            ▼ <li class>
                <a title href="https://guides.loc.gov/federalist-papers/text-31-40" style ">
                Federalist Nos. 31-40 </a>
```

Figure 5.6: Screenshot of the source HTML for the Federalist Papers

In Figure 5.6 we can see that the HTML elements that contain the URLs are nested within a `<ul>` element. The `<ul>` element has a set of class attributes (`.s-lg-subtab-ul`, `nav`, `nav-pills`, `nav-stacked`). If one of these is unique to this `<ul>` element we can use it to isolate the element. Let's search the HTML for all the `<ul>` elements on the page.

Example 5.38.

```
html |>
  html_nodes("ul")
```

```
> {xml_nodeset (4)}
> [1] <ul class="nav nav-pills nav-stacked split-button-nav" role="menu">\n<li  ...
> [2] <ul class="s-lg-subtab-ul nav nav-pills nav-stacked">\n<li class=""><a ti ...
> [3] <ul id="s-lg-page-prevnext" class="pager s-lib-hide">\n<li class="previou ...
> [4] <ul id="s-lg-guide-header-attributes" class="">\n<li id="s-lg-guide-heade ...
```

The output from Example 5.38 shows that there are 4 `<ul>` elements on the page. It's a little hard to see in the output, but the second `<ul>` element is the one we are targeting, as it contains the class attributes we identified in Figure 5.6. We can use the `html_attr()` function to extract the `class` attribute from the `<ul>` elements to see if one of them is unique to the `<ul>` element we want to isolate.

Example 5.39.

```
html |>
  html_nodes("ul") |>
  html_attr("class")
```

```
> [1] "nav nav-pills nav-stacked split-button-nav"
> [2] "s-lg-subtab-ul nav nav-pills nav-stacked"
> [3] "pager s-lib-hide"
> [4] ""
```

Effectively this is the case, as the second `<ul>` element in the output of Example 5.39 has a unique class attribute, `.s-lg-subtab-ul`. We can use this to isolate the element using the `html_nodes()` function. We then pipe this to another `html_nodes()` function to isolate the `<li>` elements nested within the `<ul>` element. See Example 5.40.

Example 5.40.

```
html |>
  html_nodes("ul.s-lg-subtab-ul") |>
  html_nodes("li")
```

```
> {xml_nodeset (9)}
> [1] <li class=""><a title="" href="https://guides.loc.gov/federalist-papers/t ...
> [2] <li class=""><a title="" href="https://guides.loc.gov/federalist-papers/t ...
> [3] <li class=""><a title="" href="https://guides.loc.gov/federalist-papers/t ...
> [4] <li class=""><a title="" href="https://guides.loc.gov/federalist-papers/t ...
> [5] <li class=""><a title="" href="https://guides.loc.gov/federalist-papers/t ...
> [6] <li class=""><a title="" href="https://guides.loc.gov/federalist-papers/t ...
> [7] <li class=""><a title="" href="https://guides.loc.gov/federalist-papers/t ...
> [8] <li class=""><a title="" href="https://guides.loc.gov/federalist-papers/t ...
> [9] <li class=""><a title="" href="https://guides.loc.gov/federalist-papers/t ...
```

Great. Now, to get the URLs we add another `html_nodes()` function to Example 5.40 to isolate the `<a>` elements nested within the `<li>` elements and then a function `html_attr()` to extract the value of an attribute. In this case, the attribute of the `<a>` elements we want is `href`. See Example 5.41.

Example 5.41.

```
html |>
  html_nodes("ul.s-lg-subtab-ul") |>
  html_nodes("li") |>
  html_nodes("a") |>
  html_attr("href")
```

```
> [1] "https://guides.loc.gov/federalist-papers/text-1-10"
> [2] "https://guides.loc.gov/federalist-papers/text-11-20"
> [3] "https://guides.loc.gov/federalist-papers/text-21-30"
> [4] "https://guides.loc.gov/federalist-papers/text-31-40"
> [5] "https://guides.loc.gov/federalist-papers/text-41-50"
> [6] "https://guides.loc.gov/federalist-papers/text-51-60"
> [7] "https://guides.loc.gov/federalist-papers/text-61-70"
> [8] "https://guides.loc.gov/federalist-papers/text-71-80"
> [9] "https://guides.loc.gov/federalist-papers/text-81-85"
```

We can assign the URLs to a variable, `fed_urls`.

With the URLs in hand, we can now retrieve the HTML for each of the nine pages. We can, of course, do this manually, as in Example 5.42.

Example 5.42.

```
# Read the HTML from the first URL in `fed_urls`
html <- read_html(fed_urls[1])
write_html(html, "../data/original/federalist_papers/fed1.html")

# Read the HTML from the second URL in `fed_urls`
html <- read_html(fed_urls[2])
write_html(html, "../data/original/federalist_papers/fed2.html")

# ... and so on
```

But this is tedious and error prone, furthermore, it doesn't scale well. If we had 1000 URLs to retrieve the HTML from, we would have to write 1000 lines of code. Instead, we can write a function to do this for us. See Example 5.43.

Example 5.43.

```
# Function to retrieve HTML from a URL and write to a file
read_write_html <- function(url) {
  # Create a file name and path from the URL
  file_name <- path_file(url) |> path_ext_set(".html")
  file_path <- path("../data/original/federalist_papers/", file_name)

  # Read the HTML from the URL
  html <- read_html(url)

  # Write the HTML to the file
  write_xml(html, file_path)
}
```

The function in Example 5.43 takes a URL as it's only argument. It then creates a file name and path from the URL. The file name is the last part of the URL, with the extension `.html`. The file path is the path to the `federalist_papers` directory in the `data/original` directory. The function then reads the HTML from the URL and writes it to the file.

Example 5.43 might not seem like a step up from Example 5.42, but it is. We can now use the `map()` function from `purrr` to iterate over the URLs in `fed_urls` and apply the `read_write_html()` function to each URL. See Example 5.44.

Example 5.44.

```
# Retrieve the HTML from each URL in `fed_urls` and write to a file
fed_urls |>
  map(read_write_html)
```

Tip

When processing multiple webpages, it's often important to manage the load on the server. In R, we can use the `Sys.sleep()` to introduce short delays between requests. This helps reduce server load when iterating over a list of webpages.
For example, we can use `Sys.sleep(1)` in our function to introduce a 1 second delay between requests.

```
read_write_html <- function(url) {
  Sys.sleep(1) # 1 second delay
  # ...
}
```

Another tip is to use the `message()` function to print a status message to the console. This can be helpful when processing a large number of webpages.

```
read_write_html <- function(url) {
  Sys.sleep(1) # 1 second delay
  message("Processing ", url) # Prints: "Processing
   ↪  https://www.example.com"
  # ...
}
```

The result of Example 5.44 can be seen in the project directory in Example 5.45.

Example 5.45.

```
data/
|-- analysis/
|-- derived/
└── original/
    └── federalist_papers/
        |-- main.html
        |-- text-1-10.html
```

```
        |-- text-11-20.html
        |-- text-21-30.html
        |-- text-31-40.html
        |-- text-41-50.html
        |-- text-51-60.html
        |-- text-61-70.html
        |-- text-71-80.html
        └── text-81-85.html
```

And of course, to finish the acquisition process, we need to ensure we have documented the code and created a data origin file. Since we have created this resource it much of the information will be up to use to document. Keep in mind that the data origin file should be written in a way that is transparent to the researcher and to would-be collaborators and the general research community.

In this section, we have built on previously introduced R coding concepts and employed various others in the process of acquiring data from the web. We have also considered topics that are more general in nature and concern interacting with data found on the internet. As you likely appreciate, web scraping often requires more knowledge of and familiarity with R as well as other web technologies. Rest assured, however, practice will increase confidence in your abilities. I encourage you to practice on your own with other websites.

Summary

In this chapter we have covered a lot of ground. On the surface we have discussed three methods for acquiring corpus data for use in text analysis. In the process we have delved into various aspects of the R programming language. Some key concepts include writing custom functions, control statements, and applying functions iteratively. We have also considered topics that are more general in nature and concern interacting with data found on the internet.

Each of these methods should be approached in a way that is transparent to the researcher and to would-be collaborators and the general research community. For this reason the documentation of the steps taken to acquire data are key both in the code and in human-facing documentation.

At this point you have both a bird's eye view of the data available on the web and strategies on how to access a great majority of it. It is now time to turn to the next step in our data analysis project: data curation. In the next chapter, I will cover how to wrangle your raw data into a tidy dataset.

Activities

☐ Add description of outcomes

Recipe

☐ update

What: Control statements, custom functions, and iteration[a]
How: Read Recipe 6 and participate in the Hypothes.is online social annotation.
Why: To increase your ability to produce effective, concise, and reproducible code. The three main areas we will cover are working with control statements, writing custom functions, and leveraging iteration. These programming strategies are often useful for acquiring data but, as we will see, they are powerful concepts that can be used throughout a reproducible research project.

[a]https://lin380.github.io/tadr/articles/recipe_6.html

Lab

☐ update

What: Control statements, custom functions, and iteration[a]
How: Clone, fork, and complete the steps in Lab 6.
Why: To gain experience working with coding strategies such as control statements, custom functions, and iteration, practice working with direct downloads and API interfaces to acquire data, and implement organizational strategies for organizing data in reproducible fashion.

[a]https://github.com/lin380/lab_6

Questions

☐ create conceptual and technical questions

Conceptual questions

- ...
- For many resources, information to describe the data origin is found on the resource's website. Visit the XXX resource and complete the data origin information.

Technical exercises

- ...
- ...

File 5.1 `1-acquire-data.qmd`: Acquire data file

````
---
title: "Acquire data"
format: html
---



## Overview
The goal of this script is to acquire and document data for this project from
 ↪  the CEDEL2 corpus. The acquired data will be stored in the
 ↪  `data/original/cedel2/` directory.



## Data origin
To document the origin of the data we created a file named `cedel2_do.csv` in the
 ↪  `data/original/` directory. This file contains the following information:

```{r}

#| label: tbl-cedel2-data-origin
#| tbl-cap: "Data origin file for the CEDEL2 corpus"
#| echo: false

# Display data origin file


readr::read_csv("../data/original/cedel2_do.csv") |> knitr::kable()
```
## Download procedures
The process to acquire data from the CEDEL2 corpus involved the following steps:



L2 Spanish Learners:
1. Navigate to the [CEDEL2 Corpus](http://cedel2.learnercorpora.com/search)
 ↪  search interface
2. Select the subcorpus "Learners of L2 Spanish"
3. Set the results to provide full texts

4. Filter the results to "L1 English - L2 Spanish"
5. Set the medium to "Written"
6. Download the data in CSV format
7. Save the CSV file to the `data/original/cedel2/` directory as
 ↪  `cedel2-l1-english-learners.csv`
````



````
Spanish natives:
````

File 5.2 `1-acquire-data.qmd`: Acquire data file

````
---
title: "Acquire data"
format: html
---

## Overview

The goal of this script is to ...

## Data origin

To document the origin of the data we created a file named ...

## Download procedures

```{r}
#| label: setup

# Load libraries
library(tidyverse)
```
```{r}
# .. additional code here to acquire data ...
```
... and so on
````

6 Curate datasets

Caution

In progress...

> The hardest bit of information to extract is the first piece.
>
> — Robert Ferrigno

Keys

- what are some of the formats that data can take?
- what R programming strategies are used to read these formats into tabular, tidy dataset structures?
- what is the importance of maintaining modularity between data and data processing in a reproducible research project?

In this chapter we will now look at the next step in a text analysis project: data curation. That is, the process of converting the original data we acquire to a tidy dataset. As Acquired data can come in a wide variety of formats that depend largely on the richness of the metadata that is included, but also can reflect individual preferences. In this chapter we will consider three general types of formats: (1) unstructured data, (2) structured data, and (3) semi-structured data. Regardless of the file type and the structure of the data, it will be necessary to consider how to curate a dataset that such that the structure reflects the basic the unit of analysis that we wish to investigate (see Chapter 4, section 4.2. The resulting dataset will be the base from which we will work to further transform the dataset such that it aligns with the analysis method(s) that we will implement. And as in previous implementation steps, we will discuss the important role of documentation.

Swirl

What: Regular Expressions[a]
How: In the R Console pane load `swirl`, run `swirl()`, and follow prompts to select the lesson.
Why: To learn the basics of how to define search patterns to match strings, or characters, using Regular Expressions.

[a]https://github.com/qtalr/lessons

6.1 Unstructured

The bulk of text that is available in the wild is of the unstructured variety. Unstructured data is data that has not been organized to make the information contained within explicit. Explicit information that is included with data is called metadata. Metadata can be linguistic or non-linguistic in nature. So for unstructured data there is little to no metadata directly associated with the data. This information needs to be added or derived for the purposes of the research, either through manual inspection or (semi-)automatic processes. For now, however, our job is just to get the unstructured data into a structured format with a minimal set of metadata that we can derive from the resource.

As an example of an unstructured source of text data, let's take a look at the Europarle Parallel Corpus[1], as introduced in Chapter 2 "Understanding data". This data contains parallel texts (source and translated documents) from the European Parliamentary proceedings for some 21 European languages. Here we will focus in on the translation from Spanish to English (Spanish-English).

6.1.1 Orientation

With the data downloaded into the `data/original/europarle/` directory we see that there are two files. One corresponding to the source language (Spanish) and one for the target language (English).

```
data/original/europarle/
├── europarl-v7.es-en.en
└── europarl-v7.es-en.es
```

Looking at the first 10 lines of the first file, we can see that this is running text.

The only meta information that we can surmise from these files is the fact that we know one is the source language and one is the target language and that each sentence is aligned (parallel) with the lines in the other file.

So with what we have we'd like to create a data frame that has the seen in **?@tbl-cd-unstructured-europarle-structure-example**.

[1]https://www.statmt.org/europarl/

6.1.2 Tidy the data

To create this dataset structure lets's read the files with the `readtext()` function from readtext package and assign them to a meaningful variable.

```
# Read the Europarle files
europarle_en <-  # English target text
  readtext::readtext("../data/original/europarle/europarl-v7.es-en.en", # path
 ↪  to the data
                     verbosity = 0) # don't show warnings

europarle_es <- # Spanish source text
  readtext::readtext("../data/original/europarle/europarl-v7.es-en.es", # path
 ↪  to the data
                     verbosity = 0) # don't show warnings
```

> **Tip**
>
> The `readtext()` function can read many different types of file formats, from structured to unstructured. However, it depends in large part on the extension of the file to recognize what algorithm to use when reading a file. In this particular case the Europarle files do not have a typical extension (they have `.en` and `.es`). The `readtext()` function will treat them as plain text (`.txt`), but it will throw a warning message. To suppress the warning message you can add the `verbosity = 0` argument.

Now there are a couple things to note about thbe `europarle_en` and `europarle_es` objects. If we inspect their structure, we will find that the dimensions of the data frame that is created is one row by two columns.

```
str(europarle_en) # inspect the structure of the object
```

> **Tip**
>
> Note that the `str()` function from base R is similar to `glimpse()`. However, `glimpse()` will attempt to show you as much data as possible. In this case since our column `text` is a very long character vector it will take a long time to render. I've chosen the `str()` function as it will automatically truncate the data.

The columns are `doc_id` and `text`. `doc_id` is created by readtext to index each file that is read in. The `text` column is where the text appears. The fact that we only have one row means that all the text in the entire file is contained in one cell! We will want to break this cell up

into rows for each sentence, but for now let's work with getting the columns to line up with our idealized dataset structure.

First let's change the type of data frame that we are working with to a tibble. This will make sure we don't accidentally print hundreds of lines to our R Markdown output and/ or the R Console. Then we will rename the `doc_id` column to `type` and change the value of that column to "Target" (for English) and "Source" (for Spanish).

```
europarle_target <-
  europarle_en |> # readtext data frame
  as_tibble() |> # convert to tibble
  rename(type = doc_id) |> # rename doc_id to type
  mutate(type = "Target") # change type value to 'Target'

europarle_source <-
  europarle_es |> # readtext data frame
  as_tibble() |> # convert to tibble
  rename(type = doc_id) |> # rename doc_id to type
  mutate(type = "Source") # change type value to 'Source'
```

We have two objects now, one corresponding to the 'Source' and the other the 'Target' parallel texts. Let's now join these two datasets, one on top of the other –that is, by rows. We wil use the `bind_rows()` function for this.

```
europarle <-
  bind_rows(europarle_target, europarle_source)

str(europarle) # inspect the structure of the object
```

The `europarle` dataset now has 2 columns, as before, and 2 rows –each corresponding to the distinct language types (Source/ Target).

Remember our goal is to create a dataset structure with three columns `type`, `sentence_id`, and `sentence`. At the moment we have `type` and `text` –where `text` has all of the sentences in for each type in a cell. So we are going to want to break up the `text` column into sentences, group the sentences that are created by `type`, and then number these sentences so that they are aligned between the distinct types.

To break up the text into sentences we are going to turn to the tidytext package. This package has a extremely useful function `unnest_tokens()` which provides an effective way to break text into various units (see `?tidytext::unnest_tokens` for a full list of token types). Since I know from looking at the raw text that each sentence is on its own line, the best strategy to break the text into sentence units is to find a way to break each line into a new row in our dataset.

To do this we need to use the `token = "regex"` (for Regular Expression) and use the `pattern = "\\n"` which tells R to look for carriage returns to use as the breaking criterion.

```
europarle_sentences <-
  europarle |>
  tidytext::unnest_tokens(output = sentence, # new column
                          input = text, # column to find text
                         token = "regex", # use a regular expression to break up
                          ↪  the text
                          pattern = "\\n", # break text by carriage returns
                          ↪  (returns after lines)
                          to_lower = FALSE) # do not lowercase the text

glimpse(europarle_sentences) # preview the structure
```

> **Tip**
>
> Regular Expressions are a powerful pattern matching syntax. They are used extensively in text manipulation and we will see them again and again. A good website to practice Regular Expressions is RegEx101[a]. You can also install the regexplain package in R to get access to a useful RStudio Addin[b].
>
> [a] https://regex101.com/
> [b] https://rstudio.github.io/rstudioaddins/

Our new `europarle_sentences` object is a data frame with almost 4 million rows! The final step to get to our envisioned dataset structure is to add the `sentence_id` column which will be calculated by grouping the data by `type` and then assigning a row number to each of the sentences in each group.

```
europarle_sentences_id <-
  europarle_sentences |> # dataset
  group_by(type) |> # group by type
  mutate(sentence_id = row_number()) |> # add a row number for each sentence for
   ↪  each level of type
  select(type, sentence_id, sentence) |> # select the relevant columns to keep
  ungroup() |>  # ungroup by type
  arrange(sentence_id, type) # arrange the dataset

europarle_sentences_id |>
  slice_head(n = 10) |>
  knitr::kable(booktabs = TRUE)
```

Table 6.1: **?(caption)**

6.1.3 Write dataset

At this point we have the curated dataset (`europarle_sentences_id`) in a tidy format. This dataset, however, is only in the current R session. We will want to write this dataset to disk so that in the next step of the text analysis workflow (transform data) we will be able to start work on this dataset and make changes as needed to fit our analysis needs.

We will leverage the project directory structure which has distinct directories for `original/` and `derived/` data(sets).

```
data/
├── derived
└── original
```

Since this dataset is derived by our work, we will added it to the `derived/` directory. I'll create a `europarle/` directory just to keep things organized.

```
# Write the curated dataset to disk
fs::dir_create(path = "../data/derived/europarle/") # create the europarle
 ↪  directory
write_csv(x = europarle_sentences_id, # object to write
          file = "../data/derived/europarle/europarle_curated.csv") # target
           ↪  file location/ name
```

This is how the directory structure under the `derived/` directory looks now.

```
data/
├── derived
│   └── europarle
│       └── europarle_curated.csv
└── original
    └──europarle
        ├── europarl-v7.es-en.en
        └── europarl-v7.es-en.es
```

6.1.4 Summary

In this section we worked with unstructured data and looked at how to read the data into an R session and manipulate the data to form a tidy dataset with a few columns that we could derive based on the information we have about the corpus.

In our discussion we worked step by step to curate the Europarle Corpus, adding in intermediate steps for illustration purposes. However, in a more realistic case the code would most likely make more extensive use of piping (`|>`) to reduce the number of intermediate objects and make the code more legible. Below I've included a sample of what that code might look like.

```
# Read data and set up `type` column
europarle_en <-
  readtext::readtext("../data/original/europarle/europarl-v7.es-en.en", # path
 ↪  to the data
                     verbosity = 0) |> # don't show warnings
  as_tibble() |> # covert to tibble
  rename(type = doc_id) |> # rename doc_id to type
  mutate(type = "Target") # change type value to 'Target'

europarle_es <-
  readtext::readtext("../data/original/europarle/europarl-v7.es-en.en", # path
 ↪  to the data
                     verbosity = 0) |>  # don't show warnings
  as_tibble() |> # covert to tibble
  rename(type = doc_id) |> # rename doc_id to type
  mutate(type = "Source") # change type value to 'Source'

# Join the datasets by rows
europarle <-
  bind_rows(europarle_en, europarle_es)

# Segment the `text` column into `sentence` units
europarle <-
  europarle |> # dataset
  tidytext::unnest_tokens(output = sentence, # new column
                          input = text, # column to find text
                         token = "regex", # use a regular expression to break up
                          ↪  the text
                          pattern = "\\n", # break text by carriage returns
                          ↪  (returns after lines)
```

```
                          to_lower = FALSE) # do not lowercase the text

# Add `sentence_id` to each `type`
europarle <-
  europarle |> # dataset
  group_by(type) |> # group by type
  mutate(sentence_id = row_number()) |> # add a row number for each sentence for
   ↪  each level of type
  select(type, sentence_id, sentence) |> # select the relevant columns to keep
  ungroup() |>  # ungroup by type
  arrange(sentence_id, type) # arrange the dataset

# Write the curated dataset to disk
fs::dir_create(path = "../data/derived/europarle/") # create the europarle
 ↪  directory
write_csv(x = europarle_sentences_id, # object to write
          file = "../data/derived/europarle/europarle_curated.csv") # target
           ↪  file location/ name
```

6.2 Structured

On the opposite side of the spectrum from unstructured data, structured data includes more metadata information –often much more. The association of metadata with the language to be analyzed means that the data has already be curated to some degree, therefore it is more apt to discuss structured data as a dataset. There are two questions, however, that need to be taken into account. One, logistical question, is what file format the dataset is in and how to we read it into R. And the second, more research-based, is whether the data is curated in a fashion that makes sense for the current research. Let's look at each of these questions briefly and then get to a practical example.

There are file formats which are purposely designed for storing structured datasets. Some very common file types are .csv, .xml, .json, etc. The data within these files is explicitly organized. For example, in a .csv file, the dataset structure is represented by delimiting the columns and rows by commas.

```
column_1,column_2,column_3
row 1 value 1,row 1 value 2,row 1 value 3
row 2 value 1,row 2 value 2,row 2 value 3
```

When read into R, the .csv file format is converted to a data frame with the appropriate structure.

With an understanding of how the information is encoding into a file, we can now turn to considerations about how the original dataset is structure and how that structure is to be used for a given research project. The curation process that is reflected in a structured dataset may or may not initially align with the goals of our research either in terms of the type(s) of information or the unit of analysis of the structured dataset. The aim, then, is to take advantage of the information and curate it such that it does align.

As an example case of curating structured datasets, we will look at the song lyric datasets acquired from Last.fm in the previous chapter.

6.2.1 Orientation

The individual datasets from the Last.fm webscrape are found inside the `data/original/lastfm/` directory, and includes the `README.md` documentation file.

```
data/
├── derived/
└── original/
    └── lastfm/
        ├── README.md
        ├── country.csv
        ├── hip_hop.csv
        ├── lyrics.csv
        ├── metal.csv
        ├── pop.csv
        └── rock.csv
```

Let's take a look at the structure of one of genres from these set of lyrics to familiarize ourselves with the structure.

```
lf_country <- read_csv(file = "../data/original/lastfm/country.csv") # read the
↪  csv file
lf_country |> # dataset
  slice_head(n = 10) |> # first 10 observations
knitr::kable(booktabs = TRUE) # print pretty table
```

Table 6.2: **?(caption)**

We can see a couple important characteristics from this preview of the dataset. First, we see the columns include `artist`, `song`, `lyrics`, `lyrics_url`, and `genre`. Second, we see that for each son the lyrics are segmented across multiple rows.

> ⚠ Tip
>
> You may notice that in addition to the lyrics being separated by line, there appears to be an artifact from the original webscrape of this data which has individual lyric lines run in to the next. An example is the the lyrics "… hurt myself toda**yT**osee if I still fee**lI** focus…". We will address this issue when it comes time to normalize the dataset in the transform process.

Given the fact that each of these files will include a `genre` label, that means that we will be able to read in each of these files in one operation and the distinction between genres will be recoverable. The next thing to think about is how we want to curate the dataset for our purposes. That is, what should the base structure of our curated dataset look like?

Let's make the assumption that we want to have the columns `artist`, `song`, `lyrics`, and `genre`. The `lyrics_url` could be useful for documentation purposes, but for our text analysis it does not appear to be very relevant –so we will drop it. The second aspect concerns the observations. As it stands, the dataset the observations reflect the formatting of the website from which the lyrics were drawn. A potentially better organization would to have each observation correspond to all the lyrics for a single song. In this case we will want to collapse the current `lyrics` column's values into lyrics for the entire song –maintaining the other measure for each of the other columns.

With this structure in mind, we are shooting for an idealized structure such as the one below.

6.2.2 Tidy the datasets

So our objectives are set, let's first read in all the files. To do this we will again use the `readtext()` function. But instead of reading one file at a time we will read all the files of interest (those with the .csv extension) in one go. The `readtext()` function allows for the use of 'wildcard' notation (`*`) in the file(s) path to enable pattern matching.

So the files in the `data/original/lastfm/` directory look like this.

```
../data/original/lastfm/README.md
../data/original/lastfm/country.csv
../data/original/lastfm/hip_hop.csv
../data/original/lastfm/lyrics.csv
../data/original/lastfm/metal.csv
../data/original/lastfm/pop.csv
../data/original/lastfm/rock.csv
```

We want all the files, except the `REAME.md` file. To do this we want our path to look like this:

```
../data/original/lastfm/*.csv
```

The wildcard `*` replaces the genre names and this effectively only matches files ending in `.csv`.

Great, that will capture the files we are looking for but when working with `readtext()` we will need to set the `text_field` argument to the column that corresponds to the text in our dataset. That is the `lyrics` column. Let's go ahead and do this and convert the result to a tibble.

```
lastfm <-
  readtext(file = "../data/original/lastfm/*.csv", # files to match using *.csv
           text_field = "lyrics") |> # text column from the datasets
  as_tibble() # convert to a tibble

glimpse(lastfm) # preview
```

Looking at the preview of the data frame we now have in `lastfm` there are a couple things to note. First, we see that a column `doc_id` has been added. This column is used by `readtext()` to index the file from which the data was read. In our case since we already have sufficient information to index our dataset, we can drop this column. Next we see that the `lyrics` column has been renamed to `text`. This is because we set this as the `text_field` when we read in the files. We can easily rename this column, but we'll leave that for later.

Let's go ahead and drop the columns that we have decided will not figure in our curated dataset. We can use the `select()` function to either select those columns we want to keep or by using the - operator, identify the columns we want to drop. The decision of 'selecting' or 'deselecting' is usually one of personal choice and code succinctness. In this case, we are dropping two columns and keeping four, so let's deselect. I will assign the result to the same name as our current dataset, effectively overwriting that dataset.

Table 6.3: **?(caption)**

```
lastfm <- # new dataset
  select(lastfm, # original dataset
         -doc_id, -lyrics_url) # drop these columns

glimpse(lastfm) # preview
```

Now let's work to collapse the lyrics in the `text` column by each distinct `artist`, `song`, and `genre` combination. We will use the `group_by()` function to create `artist song genre` groupings and then use `summarize()` to create a new column in which the `text` field is collapsed into all the song lyrics for this grouping. Inside the `summarize()` function we use `str_flatten()` with the argument `collapse = " "` to collapse each observation in `text` leaving a single whitespace between the observations (otherwise each line would then be joined contiguously to the next).

```
lastfm <-
  lastfm |> # dataset
  group_by(artist, song, genre) |> # grouping
  summarise(lyrics = str_flatten(text, collapse = " ")) |>  # collapse text into
   ↪  the new column `lyrics` (dropping `text`)
  ungroup() # unset the groupings

glimpse(lastfm) # preview
```

Let's take a look at the first 5 observations from this collapsed dataset.

```
lastfm |> # dataset
  slice_head(n = 5) |> # first 5 observations
  knitr::kable(booktabs = TRUE) # print pretty table
```

At this point, the only thing left to do to get this dataset to align with our idealized dataset structure is to organize the column ordering (using `select()`). I will also arrange the dataset alphabetically by `genre` and `artist` (using `arrange()`).

```
lastfm <-
  lastfm |> # original dataset
  select(artist, song, lyrics, genre) |> # order columns (and rename `text` to
   ↪  `lyrics`)
  arrange(genre, artist) # arrange rows by `genre` and `artist`
```

Table 6.4: **?(caption)**

```
lastfm |> # curated dataset
  slice_head(n = 5) |> # first 5 observations
  knitr::kable(booktabs = TRUE) # print pretty table
```

6.2.3 Write dataset

We now have a curated dataset that we can write to disk. Again, as with the Europarle Corpus dataset we curated before, we will write this dataset to the `data/derived/` directory –effectively ensuring that it is clear that this dataset was created by our project work.

```
fs::dir_create(path = "../data/derived/lastfm/") # create lastfm subdirectory
write_csv(lastfm,
          file = "../data/derived/lastfm/lastfm_curated.csv") # write lastfm to
           ↪  disk and label as the curated dataset
```

And here's an overview of our new directory structure.

```
data/
├── derived/
│   └── lastfm/
│       └── lastfm_curated.csv
└── original/
    └── lastfm/
        ├── README.md
        ├── country.csv
        ├── hip_hop.csv
        ├── lyrics.csv
        ├── metal.csv
        ├── pop.csv
        └── rock.csv
```

6.2.4 Summary

Again, to summarize, here is the code that will accomplish the steps we covered in this section on curating structured datasets.

```
# Read Last.fm lyrics and subset relevant columns
lastfm <-
  readtext(file = "../data/original/lastfm/*.csv", # files to match using *.csv
           text_field = "lyrics") |> # text column from the datasets
  as_tibble() |> # convert to a tibble
  select(-doc_id, -lyrics_url) # drop these columns

# Collapse text by artist, song, and genre grouping
lastfm <-
  lastfm |> # dataset
  group_by(artist, song, genre) |> # grouping
  summarise(lyrics = str_flatten(text, collapse = " ")) |>  # collapse text into
   ↪  the new column `lyrics` (dropping `text`)
  ungroup() # unset the groupings

# Order columns and arrange rows
lastfm <-
  lastfm |> # original dataset
  select(artist, song, lyrics, genre) |> # order columns (and rename `text` to
   ↪  `lyrics`)
  arrange(genre, artist) # arrange rows by `genre` and `artist`

# Write curated dataset to disk
fs::dir_create(path = "../data/derived/lastfm/") # create lastfm subdirectory
write_csv(lastfm,
          file = "../data/derived/lastfm/lastfm_curated.csv") # write lastfm to
           ↪  disk and label as the curated dataset
```

6.3 Semi-structured

At this point we have discussed curating unstructured data and structured datasets. Between these two extremes falls semi-structured data. And as the name suggests, it is a hybrid between unstructured and structured data. This means that there will be important structured metadata included with unstructured elements. The file formats and approaches to encoding the structured aspects of the data vary widely from resource to resource and therefore often requires more detailed attention to the structure of the data and often includes more sophisticated programming strategies to curate the data to produce a tidy dataset.

As an example we will work with the The Switchboard Dialog Act Corpus (SDAC) which extends the Switchboard Corpus[2] with speech act annotation. **(ADD CITATION)**

> **Tip**
>
> The SDAC dialogues (`swb1_dialogact_annot.tar.gz`) are available as a free download from the LDC[a]. To download, decompress, and organize this resource, follow the strategies discussed in "Acquire data" for Direct Downloads. The tadr package provides the `tadr::get_compressed_data()` function to accomplish this step.
>
> [a] https://catalog.ldc.upenn.edu/docs/LDC97S62/

6.3.1 Orientation

The main directory structure of the SDAC data looks like this:

```
data/
├── derived/
└── original/
    └── sdac/
        ├── README
        ├── doc/
        ├── sw00utt/
        ├── sw01utt/
        ├── sw02utt/
        ├── sw03utt/
        ├── sw04utt/
        ├── sw05utt/
        ├── sw06utt/
        ├── sw07utt/
        ├── sw08utt/
        ├── sw09utt/
        ├── sw10utt/
        ├── sw11utt/
        ├── sw12utt/
        └── sw13utt/
```

The `README` file contains basic information about the resource, the `doc/` directory contains more detailed information about the dialog annotations, and each of the following directories

[2] https://catalog.ldc.upenn.edu/LDC97S62

prefixed with `sw...` contain individual conversation files. Here's a peek at internal structure of the first couple directories.

```
├── README
├── doc
│   └── manual.august1.html
├── sw00utt
│   ├── sw_0001_4325.utt
│   ├── sw_0002_4330.utt
│   ├── sw_0003_4103.utt
│   ├── sw_0004_4327.utt
│   ├── sw_0005_4646.utt
```

Let's take a look at the first conversation file (`sw_0001_4325.utt`) to see how it is structured.

There are few things to take note of here. First we see that the conversation files have a meta-data header offset from the conversation text by a line of `=` characters. Second the header contains meta-information of various types. Third, the text is interleaved with an annotation scheme.

Some of the information may be readily understandable, such as the various pieces of meta-data in the header, but to get a better understanding of what information is encoded here let's take a look at the `README` file. In this file we get a birds eye view of what is going on. In short, the data includes 1155 telephone conversations between two people annotated with 42 'DAMSL' dialog act labels. The `README` file refers us to the `doc/manual.august1.html` file for more information on this scheme.

At this point we open the the `doc/manual.august1.html` file in a browser and do some investigation. We find out that 'DAMSL' stands for 'Discourse Annotation and Markup System of Labeling' and that the first characters of each line of the conversation text correspond to one or a combination of labels for each utterance. So for our first utterances we have:

```
o = "Other"
qw = "Wh-Question"
qy^d = "Declarative Yes-No-Question"
+ = "Segment (multi-utterance)"
```

Each utterance is also labeled for speaker ('A' or 'B'), speaker turn ('1', '2', '3', etc.), and each utterance within that turn ('utt1', 'utt2', etc.). There is other annotation provided withing each utterance, but this should be enough to get us started on the conversations.

Now let's turn to the meta-data in the header. We see here that there is information about the creation of the file: 'FILENAME', 'TOPIC', 'DATE', etc. The `doc/manual.august1.html` file

doesn't have much to say about this information so I returned to the LDC Documentation[3] and found more information in the Online Documentation[4] section. After some poking around in this documentation I discovered that that meta-data for each speaker in the corpus is found in the `caller_tab.csv` file. This tabular file does not contain column names, but the `caller_doc.txt` does. After inspecting these files manually and comparing them with the information in the conversation file I noticed that the 'FILENAME' information contained three pieces of useful information delimited by underscores `_`.

```
*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*x*


FILENAME:   4325_1632_1519
TOPIC#:     323
DATE:       920323
TRANSCRIBER:    glp
```

The first information is the document id (`4325`), the second and third correspond to the speaker number: the first being speaker A (`1632`) and the second speaker B (`1519`).

In sum, we have 1155 conversation files. Each file has two parts, a header and text section, separated by a line of `=` characters. The header section contains a 'FILENAME' line which has the document id, and ids for speaker A and speaker B. The text section is annotated with DAMSL tags beginning each line, followed by speaker, turn number, utterance number, and the utterance text. With this knowledge in hand, let's set out to create a tidy dataset with the following column structure:

6.3.2 Tidy the data

Let's begin by reading one of the conversation files into R as a character vector using the `read_lines()` function from the readr package.

```
doc <-
  read_lines(file = "../data/original/sdac/sw00utt/sw_0001_4325.utt") # read a
   ↪  single file as character vector
```

To isolate the vector element that contains the document and speaker ids, we use `str_detect()` from the stringr package. This function takes two arguments, a string and a pattern, and returns a logical value, `TRUE` if the pattern is matched or `FALSE` if not. We can use the output

[3] https://catalog.ldc.upenn.edu/docs/LDC97S62/
[4] https://catalog.ldc.upenn.edu/docs/LDC97S62/

of this function, then, to subset the `doc` character vector and only return the vector element (line) that contains `digits_digits_digits` with a regular expression. The expression combines the digit matching operator `\\d` with the `+` operator to match 1 or more contiguous digits. We then separate three groups of `\\d+` with underscores `_`. The result is `\\d+_\\d+_\\d+`.

```
doc[str_detect(doc, pattern = "\\d+_\\d+_\\d+")] # isolate pattern
```

> **Tip**
>
> The stringr package has a handy function `str_view()` and `str_view_all()` which allow for interactive pattern matching. There is also an RStudio Addin with the regexplain package which also can be very helpful for developing regular expression syntax.

The next step is to extract the three digit sequences that correspond to the `doc_id`, `speaker_a_id`, and `speaker_b_id`. First we extract the pattern that we have identified with `str_extract()` and then we can break up the single character vector into multiple parts based on the underscore `_`. The `str_split()` function takes a string and then a pattern to use to split a character vector. It will return a list of character vectors.

```
doc[str_detect(doc, "\\d+_\\d+_\\d+")] |> # isolate pattern
  str_extract(pattern = "\\d+_\\d+_\\d+") |> # extract the pattern
  str_split(pattern = "_") # split the character vector
```

A **list** is a special object type in R. It is an unordered collection of objects whose lengths can differ (contrast this with a data frame which is a collection of objects whose lengths are the same –hence the tabular format). In this case we have a list of length 1, whose sole element is a character vector of length 3 –one element per segment returned from our split. This is a desired result in most cases as if we were to pass multiple character vectors to our `str_split()` function we don't want the results to be conflated as a single character vector blurring the distinction between the individual character vectors. If we *would* like to conflate, or *flatten* a list, we can use the `unlist()` function.

```
doc[str_detect(doc, "\\d+_\\d+_\\d+")] |> # isolate pattern
  str_extract(pattern = "\\d+_\\d+_\\d+") |> # extract the pattern
  str_split(pattern = "_") |> # split the character vector
  unlist() # flatten the list to a character vector
```

Let's flatten the list in this case, as we have a single character vector, and assign this result to `doc_speaker_info`.

```
doc_speaker_info <-
  doc[str_detect(doc, "\\d+_\\d+_\\d+")] |> # isolate pattern
  str_extract(pattern = "\\d+_\\d+_\\d+") |> # extract the pattern
  str_split(pattern = "_") |>  # split the character vector
  unlist() # flatten the list to a character vector
```

`doc_speaker_info` is now a character vector of length three. Let's subset each of the elements and assign them to meaningful variable names so we can conveniently use them later on in the tidying process.

```
doc_id <- doc_speaker_info[1] # extract by index
speaker_a_id <- doc_speaker_info[2] # extract by index
speaker_b_id <- doc_speaker_info[3] # extract by index
```

The next step is to isolate the text section extracting it from rest of the document. As noted previously, a sequence of `=` separates the header section from the text section. What we need to do is to index the point in our character vector `doc` where that line occurs and then subset the `doc` from that point until the end of the character vector. Let's first find the point where the `=` sequence occurs. We will again use the `str_detect()` function to find the pattern we are looking for (a contiguous sequence of `=`), but then we will pass the logical result to the `which()` function which will return the element index number of this match.

```
doc |>
  str_detect(pattern = "=+") |> # match 1 or more `=`
  which() # find vector index
```

So for this file `31` is the index in `doc` where the `=` sequence occurs. Now it is important to keep in mind that we are working with a single file from the `sdac/` data. We need to be cautious to not create a pattern that may be matched multiple times in another document in the corpus. As the `=+` pattern will match `=`, or `==`, or `===`, etc. it is not implausible to believe that there might be a `=` character on some other line in one of the other files. Let's update our regular expression to avoid this potential scenario by only matching sequences of three or more `=`. In this case we will make use of the curly bracket operators `{}`.

```
doc |>
  str_detect(pattern = "={3,}") |> # match 3 or more `=`
  which() # find vector index
```

We will get the same result for this file, but will safeguard ourselves a bit as it is unlikely we will find multiple matches for `===`, `====`, etc.

`31` is the index for the `=` sequence, but we want the next line to be where we start reading the text section. To do this we increment the index by 1.

```
text_start_index <-
  doc |>
  str_detect(pattern = "={3,}") |> # match 3 or more `=`
  which() # find vector index
text_start_index <- text_start_index + 1 # increment index by 1
```

The index for the end of the text is simply the length of the `doc` vector. We can use the `length()` function to get this index.

```
text_end_index <- length(doc)
```

We now have the bookends, so to speak, for our text section. To extract the text we subset the `doc` vector by these indices.

```
text <- doc[text_start_index:text_end_index] # extract text
head(text) # preview first lines of `text`
```

The text has some extra whitespace on some lines and there are blank lines as well. We should do some cleaning up before moving forward to organize the data. To get rid of the whitespace we use the `str_trim()` function which by default will remove leading and trailing whitespace from each line.

```
text <- str_trim(text) # remove leading and trailing whitespace
head(text) # preview first lines of `text`
```

To remove blank lines we will use the a logical expression to subset the `text` vector. `text != ""` means return TRUE where lines are not blank, and FALSE where they are.

```
text <- text[text != ""] # remove blank lines
head(text) # preview first lines of `text`
```

Our first step towards a tidy dataset is to now combine the `doc_id` and each element of `text` in a data frame.

```
data <- data.frame(doc_id, text) # tidy format `doc_id` and `text`
slice_head(data, n = 5) |> # preview first lines of `text`
  knitr::kable(booktabs = TRUE)
```

Table 6.5: **?(caption)**

With our data now in a data frame, its time to parse the `text` column and extract the damsl tags, speaker, speaker turn, utterance number, and the utterance text itself into separate columns. To do this we will make extensive use of regular expressions. Our aim is to find a consistent pattern that distinguishes each piece of information from other other text in a given row of `data$text` and extract it.

The best way to learn regular expressions is to use them. To this end I've included a link to the interactive regular expression practice website regex101[5].

Open this site and copy the text below into the 'TEST STRING' field.

```
o          A.1 utt1: Okay.  /
qw          A.1 utt2: {D So, }
qy^d          B.2 utt1: [ [ I guess, +
+          A.3 utt1: What kind of experience [ do you, + do you ] have, then with
 ↪  child care? /
+          B.4 utt1: I think, ] + {F uh, } I wonder ] if that worked. /
qy          A.5 utt1: Does it say something? /
sd          B.6 utt1: I think it usually does.  /
ad          B.6 utt2: You might try, {F uh, }  /
h          B.6 utt3: I don't know,  /
ad          B.6 utt4: hold it down a little longer,  /
```

Now manually type the following regular expressions into the 'REGULAR EXPRESSION' field one-by-one (each is on a separate line). Notice what is matched as you type and when you've finished typing. You can find out exactly what the component parts of each expression are doing by toggling the top right icon in the window or hovering your mouse over the relevant parts of the expression.

```
^.+?\s
[AB]\.\d+
utt\d+
:.+$
```

As you can now see, we have regular expressions that will match the damsl tags, speaker and speaker turn, utterance number, and the utterance text. To apply these expressions to our data and extract this information into separate columns we will make use of the `mutate()` and `str_extract()` functions. `mutate()` will take our data frame and create new columns with

[5]https://regex101.com

values we match and extract from each row in the data frame with `str_extract()`. Notice that `str_extract()` is different than `str_extract_all()`. When we work with `mutate()` each row will be evaluated in turn, therefore we only need to make one match per row in `data$text`.

I've chained each of these steps in the code below, dropping the original `text` column with `select(-text)`, and overwriting `data` with the results.

```
# Extract column information from `text`
data <-
  data |> # current dataset
  mutate(damsl_tag = str_extract(string = text, pattern = "^.+?\\s")) |>  #
   ↪  extract damsl tags
  mutate(speaker_turn = str_extract(string = text, pattern = "[AB]\\.\\d+")) |>
   ↪  # extract speaker_turn pairs
  mutate(utterance_num = str_extract(string = text, pattern = "utt\\d+")) |> #
   ↪  extract utterance number
  mutate(utterance_text = str_extract(string = text, pattern = ":.+$")) |>  #
   ↪  extract utterance text
  select(-text) # drop the `text` column

glimpse(data) # preview the data set
```

> **Tip**
>
> One twist you will notice is that regular expressions in R require double backslashes (\\\\) where other programming environments use a single backslash (\\).

There are a couple things left to do to the columns we extracted from the text before we move on to finishing up our tidy dataset. First, we need to separate the `speaker_turn` column into `speaker` and `turn_num` columns and second we need to remove unwanted characters from the `damsl_tag`, `utterance_num`, and `utterance_text` columns.

To separate the values of a column into two columns we use the `separate()` function. It takes a column to separate and character vector of the names of the new columns to create. By default the values of the input column will be separated by non-alphanumeric characters. In our case this means the `.` will be our separator.

```
data <-
  data |> # current dataset
  separate(col = speaker_turn, # source column
           into = c("speaker", "turn_num")) # separate speaker_turn into
            ↪  distinct columns: speaker and turn_num
```

```
glimpse(data) # preview the data set
```

To remove unwanted leading or trailing whitespace we apply the `str_trim()` function. For removing other characters we matching the character(s) and replace them with an empty string (`""`) with the `str_replace()` function. Again, I've chained these functions together and overwritten `data` with the results.

```
# Clean up column information
data <-
  data |> # current dataset
  mutate(damsl_tag = str_trim(damsl_tag)) |> # remove leading/ trailing
   ↪  whitespace
  mutate(utterance_num = str_replace(string = utterance_num, pattern = "utt",
   ↪  replacement = "")) |> # remove 'utt'
  mutate(utterance_text = str_replace(string = utterance_text, pattern = ":\\s",
   ↪  replacement = "")) |> # remove ': '
  mutate(utterance_text = str_trim(utterance_text)) # trim leading/ trailing
   ↪  whitespace

glimpse(data) # preview the data set
```

To round out our tidy dataset for this single conversation file we will connect the `speaker_a_id` and `speaker_b_id` with speaker A and B in our current dataset adding a new column `speaker_id`. The `case_when()` function does exactly this: allows us to map rows of `speaker` with the value "A" to `speaker_a_id` and rows with value "B" to `speaker_b_id`.

```
# Link speaker with speaker_id
data <-
  data |> # current dataset
  mutate(speaker_id = case_when( # create speaker_id
    speaker == "A" ~ speaker_a_id, # speaker_a_id value when A
    speaker == "B" ~ speaker_b_id # speaker_b_id value when B
  ))

glimpse(data) # preview the data set
```

We now have the tidy dataset we set out to create. But this dataset only includes one conversation file! We want to apply this code to all 1155 conversation files in the `sdac/` corpus. The approach will be to create a custom function which groups the code we've done for this single

file and then iterative send each file from the corpus through this function and combine the results into one data frame.

Here's the custom function with some extra code to print a progress message for each file when it runs.

```
extract_sdac_metadata <- function(file) {
  # Function: to read a Switchboard Corpus Dialogue file and extract meta-data
  cat("Reading", basename(file), "...")

  # Read `file` by lines
  doc <- read_lines(file)

  # Extract `doc_id`, `speaker_a_id`, and `speaker_b_id`
  doc_speaker_info <-
    doc[str_detect(doc, "\\d+_\\d+_\\d+")] |> # isolate pattern
    str_extract("\\d+_\\d+_\\d+") |> # extract the pattern
    str_split(pattern = "_") |> # split the character vector
    unlist() # flatten the list to a character vector
  doc_id <- doc_speaker_info[1] # extract `doc_id`
  speaker_a_id <- doc_speaker_info[2] # extract `speaker_a_id`
  speaker_b_id <- doc_speaker_info[3] # extract `speaker_b_id`

  # Extract `text`
  text_start_index <- # find where header info stops
    doc |>
    str_detect(pattern = "={3,}") |> # match 3 or more `=`
    which() # find vector index

  text_start_index <- text_start_index + 1 # increment index by 1
  text_end_index <- length(doc) # get the end of the text section

  text <- doc[text_start_index:text_end_index] # extract text
  text <- str_trim(text) # remove leading and trailing whitespace
  text <- text[text != ""] # remove blank lines

  data <- data.frame(doc_id, text) # tidy format `doc_id` and `text`

  # Extract column information from `text`
  data <-
    data |>
```

```
    mutate(damsl_tag = str_extract(string = text, pattern = "^.+?\\s")) |>  #
     ↪  extract damsl tags
    mutate(speaker_turn = str_extract(string = text, pattern = "[AB]\\.\\d+"))
     ↪  |> # extract speaker_turn pairs
    mutate(utterance_num = str_extract(string = text, pattern = "utt\\d+")) |> #
     ↪  extract utterance number
    mutate(utterance_text = str_extract(string = text, pattern = ":.+$")) |>  #
     ↪  extract utterance text
    select(-text)

  # Separate speaker_turn into distinct columns
  data <-
    data |> # current dataset
    separate(col = speaker_turn, # source column
             into = c("speaker", "turn_num")) # separate speaker_turn into
              ↪  distinct columns: speaker and turn_num

  # Clean up column information
  data <-
    data |>
    mutate(damsl_tag = str_trim(damsl_tag)) |> # remove leading/ trailing
     ↪  whitespace
    mutate(utterance_num = str_replace(string = utterance_num, pattern = "utt",
     ↪  replacement = "")) |> # remove 'utt'
    mutate(utterance_text = str_replace(string = utterance_text, pattern =
     ↪  ":\\s", replacement = "")) |> # remove ': '
    mutate(utterance_text = str_trim(utterance_text)) # trim leading/ trailing
     ↪  whitespace

  # Link speaker with speaker_id
  data <-
    data |> # current dataset
    mutate(speaker_id = case_when( # create speaker_id
      speaker == "A" ~ speaker_a_id, # speaker_a_id value when A
      speaker == "B" ~ speaker_b_id # speaker_b_id value when B
    ))
  cat(" done.\n")
  return(data) # return the data frame object
}
```

As a sanity check we will run the `extract_sdac_metadata()` function on a the conversation file we were just working on to make sure it works as expected.

```
extract_sdac_metadata(file = "../data/original/sdac/sw00utt/sw_0001_4325.utt")
 ↪  |>
  glimpse()
```

Looks good!

So now it's time to create a vector with the paths to all of the conversation files. `fs::dir_ls()` interfaces with our OS file system and will return the paths to the files in the specified directory. We also add a pattern to match conversation files (`regexp = \\.utt$`) so we don't accidentally include other files in the corpus. `recurse` set to `TRUE` means we will get the full path to each file.

```
sdac_files <-
  fs::dir_ls(path = "../data/original/sdac/", # source directory
             recurse = TRUE, # traverse all sub-directories
             type = "file", # only return files
             regexp = "\\.utt$") # only return files ending in .utt
head(sdac_files) # preview file paths
```

```
../data/original/sdac/sw00utt/sw_0001_4325.utt
../data/original/sdac/sw00utt/sw_0002_4330.utt
../data/original/sdac/sw00utt/sw_0003_4103.utt
../data/original/sdac/sw00utt/sw_0004_4327.utt
../data/original/sdac/sw00utt/sw_0005_4646.utt
../data/original/sdac/sw00utt/sw_0006_4108.utt
```

o pass each conversation file in the vector of paths to our conversation files iteratively to the `extract_sdac_metadata()` function we use `map()`. This will apply the function to each conversation file and return a data frame for each. `bind_rows()` will then join the resulting data frames by rows to give us a single tidy dataset for all 1155 conversations. Note there is a lot of processing going on here we have to be patient.

```
# Read files and return a tidy dataset
sdac <-
  sdac_files |> # pass file names
  map(extract_sdac_metadata) |> # read and tidy iteratively
  bind_rows() # bind the results into a single data frame
```

We now see that we have `nrow(sdac)` observations (individual utterances in this dataset).

```
glimpse(sdac) # preview complete curated dataset
```

6.3.3 Write datasets

Again as in the previous cases, we will write this dataset to disk to prepare for the next step in our text analysis project.

```
fs::dir_create(path = "../data/derived/sdac/") # create sdac subdirectory
write_csv(sdac,
          file = "../data/derived/sdac/sdac_curated.csv") # write sdac to disk
           ↪  and label as the curated dataset
```

The directory structure now looks like this:

```
data/
├── derived/
│   └── sdac/
│       └── sdac_curated.csv
└── original/
    └── sdac/
        ├── README
        ├── doc/
        ├── sw00utt/
        ├── sw01utt/
        ├── sw02utt/
        ├── sw03utt/
        ├── sw04utt/
        ├── sw05utt/
        ├── sw06utt/
        ├── sw07utt/
        ├── sw08utt/
        ├── sw09utt/
        ├── sw10utt/
        ├── sw11utt/
        ├── sw12utt/
        └── sw13utt/
```

6.3.4 Summary

In this section we looked at semi-structured data. This type of data often requires the most work to organize into a tidy dataset. We continued to work with many of the R programming strategies introduced to this point in the coursebook. We also made more extensive use of regular expressions to pick out information from a semi-structured document format.

To round out this section I've provided a code summary of the steps involved to conduct the curation of the Switchboard Dialogue Act Corpus files. Note that I've added the `extract_sdac_metadata()` custom function to a file called `curate_functions.R` and sourced this file. This will make the code more succinct and legible here, as well in your own research projects.

```
# Source the `extract_sdac_metadata()` function
source("../functions/curate_functions.R")

# Get list of the corpus files (.utt)
sdac_files <-
  fs::dir_ls(path = "../data/original/sdac/", # source directory
             recurse = TRUE, # traverse all sub-directories
             type = "file", # only return files
             regexp = "\\.utt$") # only return files ending in .utt

# Read files and return a tidy dataset
sdac <-
  sdac_files |> # pass file names
  map(extract_sdac_metadata) |> # read and tidy iteratively
  bind_rows() # bind the results into a single data frame

# Write curated dataset to disk
fs::dir_create(path = "../data/derived/sdac/") # create sdac subdirectory
write_csv(sdac,
          file = "../data/derived/sdac/sdac_curated.csv") # write sdac to disk
           ↪  and label as the curated dataset
```

6.4 Documentation

At this stage we again want to ensure that the data that we have derived is well-documented. Where in the data acquisition process the documentation was focused on the sampling frame, curated datasets require documentation that describes the structure of the now rectangular dataset and its attributes. This documentation is known as a **data dictionary**. At the

curation stage this documentation often contains the following information ("How to Make a Data Dictionary" 2021):

- names of the variables (as they appear in the dataset)
- human-readable names for the variables
- short prose descriptions of the variables, including units of measurement (where applicable)

A data dictionary will take the format of a table and can be stored in a tabular-oriented file format (such as .csv). It is often easier to work with a spreadsheet to create this documentation. I suggest creating a .csv file with the basic structure of the documentation. You can do this however you choose, but I suggest using something along these lines as seen in the following custom function, `data_dic_starter()`.

```
data_dic_starter <- function(data, file_path) {
  # Function:
  # Creates a .csv file with the basic information
  # to document a curated dataset

  tibble(variable_name = names(data), # column with existing variable names
       name = "", # column for human-readable names
       description = "") |> # column for prose description
  write_csv(file = file_path) # write to disk
}
```

Running this function in the R Console on the curated dataset (in this case the `sdac` dataset), will provide this structure.

The resulting .csv file can then be opened with spreadsheet software (such as MS Excel, Google Sheets, etc.) and edited.[6]

Save this file as a .csv file and replace the original starter file. Note that it is important to use a plain-text file format for the official documentation file and avoid proprietary formats to ensure open accessibility and future compatibility.[7]

Our `data/derived/` directory now looks like this.

[6]Note on RStudio Cloud you will need to download the .csv file and, after editing, upload the complete data dictionary file. Make sure to save the edited file as a .csv file.

[7]Although based on spreadsheets, Broman and Woo (2018) outlines many of the best for good data organization regardless of the technology.

```
data/
└── derived/
    └── sdac/
        ├── sdac_curated.csv
        └── data_dictionary_sdac.csv
```

Summary

In this chapter we looked at the process of structuring data into a dataset. This included a discussion on three main types of data –unstructured, structured, and semi-structured. The level of structure of the original data(set) will vary from resource to resource and by the same token so will the file format used to support the level of meta-information included. The results from our data curation resulted in a curated dataset that is saved separate from the original data to maintain modularity between what the data(set) looked like before we intervene and afterwards. In addition to the code we use to derived the curated dataset's structure, we also include a data dictionary which documents the names of the variables and provides sufficient description of these variables so that it is clear what our dataset contains.

It is important to recognized that this curated dataset will form the base for the next step in our text analysis project and the last step in data preparation for analysis: dataset transformation. This last step in preparing data for analysis is to convert this curated dataset into a dataset that is directly aligned with the research aims (i.e. analysis method(s)) of the project. Since there can be multiple analysis approaches applied the original data in a research project, this curated dataset serves as the point of departure for each of the subsequent datasets derived from the transformational steps.

Activities

Recipe

What: Regular Expressions and reshaping datasets[a]
How: Read Recipe 7 and participate in the Hypothes.is online social annotation.
Why: To how regular expressions are helpful in developing strategies for matching, extracting, and/ or replacing patterns in character sequences and how to change the dimensions of a dataset to either expand or collapse columns or rows.

[a] https://lin380.github.io/tadr/articles/recipe_7.html

Lab

What: Regular Expressions and reshaping datasets[a]
How: Clone, fork, and complete the steps in Lab 7.
Why: To gain experience working with coding strategies reshaping data using tidyverse functions and regular expressions, to practice reading/ writing data from/ to disk, and to implement organizational strategies for organizing and documenting a dataset in reproducible fashion.

[a]https://github.com/lin380/lab_7

Questions

Conceptual questions

1. ...
2. ...

Technical exercises

1. ...
2. ...

7 Transform datasets

> **Caution**
>
> Under development.

Part IV

Analysis

In this section we turn to the analysis of datasets, the evaluation of results, and the interpretation of the findings. We will outline the three main types of statistical analyses: Exploratory Data Analysis (EDA), Predictive Data Analysis (PDA), and Inferential Data Analysis (IDA). Each of these analysis types have distinct, non-overlapping aims and therefore should be determined from the outset of the research project and included as part of the research blueprint. The aim of this section is to establish a clearer picture of the goals, methods, and value of each of these approaches.

8 Exploration

> **Caution**
>
> Under development.

9 Prediction

> **Caution**
>
> Under development.

10 Inference

> **Caution**
>
> Under development.

Part V

Communication

In this section I cover the steps in presenting the findings of the research both as a research document and as a reproducible research project. Both research documents and reproducible projects are fundamental components of modern scientific inquiry. On the one hand a research document provides readers a detailed summary of the main import of the research study. On the other hand making the research project available to interested readers ensures that the scientific community can gain insight into the process implemented in the research and thus enables researchers to vet and extend this research to build a more robust and verifiable research base.

11 Reports

> **Caution**
>
> Under development.

12 Collaboration

> **Caution**
>
> Under development.

References

Ackoff, Russell L. 1989. "From Data to Wisdom." *Journal of Applied Systems Analysis* 16 (1): 3–9.

Ädel, Annelie. 2020. "Corpus Compilation." In *A Practical Handbook of Corpus Linguistics*, edited by Magali Paquot and Stefan Th. Gries, 3–24. Switzerland: Springer.

Albert, Saul, Laura E. de Ruiter, and J. P. de Ruiter. 2015. "CABNC: The Jeffersonian Transcription of the Spoken British National Corpus." TalkBank.

Allaire, JJ. 2022. *Quarto: R Interface to Quarto Markdown Publishing System*. https://github.com/quarto-dev/quarto-r.

Allaire, JJ, Yihui Xie, Christophe Dervieux, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, et al. 2023. *Rmarkdown: Dynamic Documents for r*. https://CRAN.R-project.org/package=rmarkdown.

Baayen, R. Harald, L. B. Feldman, and R. Schreuder. 2006. "Morphological Influences on the Recognition of Monosyllabic Monomorphemic Words." *Journal of Memory and Language* 55: 290–313. https://doi.org/10.1016/j.jml.2006.03.008.

Bao, Wang, Ning Lianju, and Kong Yue. 2019. "Integration of Unsupervised and Supervised Machine Learning Algorithms for Credit Risk Assessment." *Expert Systems with Applications* 128 (August): 301–15. https://doi.org/10.1016/j.eswa.2019.02.033.

Benoit, Kenneth. 2020. *Quanteda.corpora: A Collection of Corpora for Quanteda*. http://github.com/quanteda/quanteda.corpora.

Benoit, Kenneth, David Muhr, and Kohei Watanabe. 2021. *Stopwords: Multilingual Stopword Lists*. https://github.com/quanteda/stopwords.

Blischak, John, Peter Carbonetto, and Matthew Stephens. 2021. *Workflowr: A Framework for Reproducible and Collaborative Data Science*. https://github.com/workflowr/workflowr.

Braginsky, Mika. 2022. *Wordbankr: Accessing the Wordbank Database*. https://CRAN.R-project.org/package=wordbankr.

Bresnan, Joan. 2007. "A Few Lessons from Typology." *Linguistic Typology* 11 (1): 297–306.

Broman, Karl W., and Kara H. Woo. 2018. "Data Organization in Spreadsheets." *The American Statistician* 72 (1): 2–10. https://doi.org/10.1080/00031305.2017.1375989.

Brown, Keith. 2005. *Encyclopedia of Language and Linguistics*. Vol. 1. Elsevier.

Buckheit, Jonathan B., and David L. Donoho. 1995. "Wavelab and Reproducible Research." In *Wavelets and Statistics*, 55–81. Springer.

Bychkovska, Tetyana, and Joseph J. Lee. 2017. "At the Same Time: Lexical Bundles in L1 and L2 University Student Argumentative Writing." *Journal of English for Academic Purposes* 30 (November): 38–52. https://doi.org/10.1016/j.jeap.2017.10.008.

Campbell, Lyle. 2001. "The History of Linguistics." In *The Handbook of Linguistics*, edited by Mark Aronoff and Janie Rees-Miller, 81–104. Blackwell Handbooks in Linguistics. Blackwell Publishers.

Carmi, Elinor, Simeon J. Yates, Eleanor Lockley, and Alicja Pawluczuk. 2020. "Data Citizenship: Rethinking Data Literacy in the Age of Disinformation, Misinformation, and Malinformation." *Internet Policy Review* 9 (2).

Chambers, John M. 2020. "S, r, and Data Science." *Proceedings of the ACM on Programming Languages* 4 (HOPL): 1–17. https://doi.org/10.1145/3386334.

Chan, Sin-wai. 2014. *Routledge Encyclopedia of Translation Technology*. Routledge.

Conway, Lucian Gideon, Laura Janelle Gornick, Chelsea Burfeind, Paul Mandella, Andrea Kuenzli, Shannon C. Houck, and Deven Theresa Fullerton. 2012. "Does Complex or Simple Rhetoric Win Elections? An Integrative Complexity Analysis of u.s. Presidential Campaigns." *Political Psychology* 33 (5): 599–618. https://doi.org/10.1111/j.1467-9221.2012.00910.x.

Cross, Nigel. 2006. "Design as a Discipline." *Designerly Ways of Knowing*, 95–103.

"Data Never Sleeps 7.0 Infographic." 2019. https://www.domo.com/learn/infographic/data-never-sleeps-7.

Desjardins, Jeff. 2019. "How Much Data Is Generated Each Day?" *Visual Capitalist*.

Donoho, David. 2017. "50 Years of Data Science." *Journal of Computational and Graphical Statistics* 26 (4): 745–66. https://doi.org/10.1080/10618600.2017.1384734.

Dubnjakovic, Ana, and Patrick Tomlin. 2010. *A Practical Guide to Electronic Resources in the Humanities*. Elsevier.

Eisenstein, Jacob, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2012. "Mapping the Geographical Diffusion of New Words." *Computation and Language*, 1–13. https://doi.org/10.1371/journal.pone.0113114.

Francom, Jerid. 2022. "Corpus Studies of Syntax." In *The Cambridge Handbook of Experimental Syntax*, edited by Grant Goodall, 687–713. Cambridge Handbooks in Language and Linguistics. Cambridge University Press.

———. 2023. *Qtalrkit: Quantitative Text Analysis for Linguists Resource Kit*.

Gandrud, Christopher. 2015. *Reproducible Research with r and r Studio*[1]. Second edition. CRC Press.

Gentleman, Robert, and Duncan Temple Lang. 2007. "Statistical Analyses and Reproducible Research." *Journal of Computational and Graphical Statistics* 16 (1): 1–23.

Gilquin, Gaëtanelle, and Stefan Th Gries. 2009. "Corpora and Experimental Methods: A State-of-the-Art Review." *Corpus Linguistics and Linguistic Theory* 5 (1): 1–26. https://doi.org/10.1515/CLLT.2009.001.

Gomez-Uribe, Carlos A., and Neil Hunt. 2015. "The Netflix Recommender System: Algorithms, Business Value, and Innovation." *ACM Transactions on Management Information Systems (TMIS)* 6 (4): 1–19.

Gries, Stefan Th. 2021. *Statistics for Linguistics with r*. De Gruyter Mouton.

[1]https://www.ncbi.nlm.nih.gov/pubmed/17811671

Gries, Stefan Th. 2013. *Statistics for Linguistics with r. A Practical Introduction*. 2nd revise.

Grieve, Jack, Andrea Nini, and Diansheng Guo. 2018. "Mapping Lexical Innovation on American Social Media." *Journal of English Linguistics* 46 (4): 293–319.

Hay, Jennifer. 2002. "From Speech Perception to Morphology: Affix Ordering Revisited." *Language* 78 (3): 527–55.

Hester, Jim, Hadley Wickham, and Gábor Csárdi. 2023. *Fs: Cross-Platform File System Operations Based on Libuv*. https://CRAN.R-project.org/package=fs.

Hicks, Stephanie C., and Roger D. Peng. 2019. "Elements and Principles for Characterizing Variation Between Data Analyses." arXiv. https://doi.org/10.48550/arXiv.1903.07639.

"How to Make a Data Dictionary." 2021. *OSF Guides*. https://help.osf.io/hc/en-us/articles/360019739054-How-to-Make-a-Data-Dictionary.

Ide, Nancy, Collin Baker, Christiane Fellbaum, Charles Fillmore, and Rebecca Passonneau. 2008. "MASC: The Manually Annotated Sub-Corpus of American English." In *6th International Conference on Language Resources and Evaluation, LREC 2008*, 2455–60. European Language Resources Association (ELRA).

Ignatow, Gabe, and Rada Mihalcea. 2017. *An Introduction to Text Mining: Research Design, Data Collection, and Analysis*. Sage Publications.

Jaeger, T Florian, and Neal Snider. 2007. "Implicit Learning and Syntactic Persistence: Surprisal and Cumulativity." *University of Rochester Working Papers in the Language Sciences* 3 (1).

Kaur, Jashanjot, and P. Kaur Buttar. 2018. "A Systematic Review on Stopword Removal Algorithms." *International Journal on Future Revolution in Computer Science & Communication Engineering* 4 (4): 207–10.

Kloumann, IM, CM Danforth, KD Harris, and CA Bliss. 2012. "Positivity of the English Language." *PloS One*.

Kostić, Aleksandar, Tanja Marković, and Aleksandar Baucal. 2003. "Inflectional Morphology and Word Meaning: Orthogonal or Co-Implicative Cognitive Domains?" In *Morphological Structure in Language Processing*, edited by R. Harald Baayen and Robert Schreuder, 1–44. De Gruyter Mouton. https://doi.org/10.1515/9783110910186.1.

Kowalski, John, and Rob Cavanaugh. 2022. *TBDBr: Easy Access to TalkBankDB via r API*.

Krathwohl, David R. 2002. "A Revision of Bloom's Taxonomy: An Overview." *Theory into Practice* 41 (4): 212–18.

Kross, Sean, Nick Carchedi, Bill Bauer, and Gina Grdina. 2020. *Swirl: Learn r, in r*. http://swirlstats.com.

Kucera, H, and W N Francis. 1967. *Computational Analysis of Present Day American English*. Brown University Press Providence.

Landau, William Michael. 2023. *Targets: Dynamic Function-Oriented Make-Like Declarative Pipelines*. https://CRAN.R-project.org/package=targets.

Lantz, Brett. 2013. *Machine Learning with r*. Birmingham: Packt Publishing.

Leech, Geoffrey. 1992. "100 Million Words of English: The British National Corpus (BNC)," no. 1991: 1–13.

Lewis, Michael. 2004. *Moneyball: The Art of Winning an Unfair Game*. WW Norton & Company.

Lozano, Cristóbal. 2009. "CEDEL2: Corpus Escrito Del Español L2." *Applied Linguistics Now: Understanding Language and Mind/La Lingüística Aplicada Hoy: Comprendiendo El Lenguaje y La Mente. Almería: Universidad de Almería*, 197–212.

Magueresse, Alexandre, Vincent Carles, and Evan Heetderks. 2020. "Low-Resource Languages: A Review of Past Work and Future Challenges." arXiv. https://arxiv.org/abs/2006.07264.

Manning, Christopher. 2003. "Probabilistic Syntax." In *Probabilistic Linguistics*, edited by Bod, Jennifer Hay, and Jannedy, 289–341. Cambridge, MA: MIT Press.

Marcus, Mitchell P, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. "Building a Large Annotated Corpus of English: The Penn Treebank." *Computational Linguistics* 19 (2): 313–30.

Marwick, Ben, Carl Boettiger, and Lincoln Mullen. 2018. "Packaging Data Analytical Work Reproducibly Using r (and Friends)." *The American Statistician* 72 (1): 80–88.

Moroz, George. 2023. *Lingtypology: Linguistic Typology and Mapping*. https://CRAN.R-project.org/package=lingtypology.

Mosteller, Frederick, and David L Wallace. 1963. "Inference in an Authorship Problem." *Journal of the American Statistical Association* 58 (302): 275–309. https://www.jstor.org/stable/2283270.

Muñoz, Carmen, ed. 2006. *Age and the Rate of Foreign Language Learning*. 1st ed. Vol. 19. Second Language Acquisition Series. Clevedon: Multilingual Matters.

Nisioi, Sergiu, Ella Rabinovich, Liviu P. Dinu, and Shuly Wintner. 23-28, 2016-05. "A Corpus of Native, Non-Native and Translated Texts." In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. Portorož, Slovenia: European Language Resources Association (ELRA).

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D Manning, Ryan McDonald, et al. 2016. "Universal Dependencies V1: A Multilingual Treebank Collection." *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 1659–66.

Olohan, Maeve. 2008. "Leave It Out! Using a Comparable Corpus to Investigate Aspects of Explicitation in Translation." *Cadernos de Tradução*, 153–69.

Paquot, Magali, and Stefan Th. Gries, eds. 2020. *A Practical Handbook of Corpus Linguistics*. Switzerland: Springer.

Riehemann, Susanne Z. 2001. "A Constructional Approach to Idioms and Word Formation." PhD thesis, Stanford.

Rinker, Tyler, and Dason Kurkiewicz. 2019. *Pacman: Package Management Tool*. https://github.com/trinker/pacman.

Roediger, H. L. L, and K. B. B McDermott. 2000. "Distortions of Memory." *The Oxford Handbook of Memory*, 149–62.

Rowley, Jennifer. 2007. "The Wisdom Hierarchy: Representations of the DIKW Hierarchy." *Journal of Information Science* 33 (2): 163–80. https://doi.org/10.1177/0165551506070706.

Saxena, Shweta, and Manasi Gyanchandani. 2020. "Machine Learning Methods for Computer-Aided Breast Cancer Diagnosis Using Histopathology: A Narrative Review." *Journal of Medical Imaging and Radiation Sciences* 51 (1): 182–93.

Talarico, Jennifer M., and David C. Rubin. 2003. "Confidence, Not Consistency, Characterizes Flashbulb Memories." *Psychological Science* 14 (5): 455–61. https://doi.org/10.1111/1467-9280.02453.

Tottie, Gunnel. 2011. "Uh and Um as Sociolinguistic Markers in British English." *International Journal of Corpus Linguistics* 16 (2): 173–97.

University of Colorado Boulder. 2008. "Switchboard Dialog Act Corpus. Web Download." Linguistic Data Consortium.

Voigt, Rob, Nicholas P. Camp, Vinodkumar Prabhakaran, William L. Hamilton, Rebecca C. Hetey, Camilla M. Griffiths, David Jurgens, Dan Jurafsky, and Jennifer L. Eberhardt. 2017. "Language from Police Body Camera Footage Shows Racial Disparities in Officer Respect." *Proceedings of the National Academy of Sciences* 114 (25): 6521–26.

White, John Myles. 2023. *ProjectTemplate: Automates the Creation of New Statistical Analysis Projects*. http://projecttemplate.net.

Wickham, Hadley. 2014. "Tidy Data." *Journal of Statistical Software* 59 (10). https://doi.org/10.18637/jss.v059.i10.

———. 2022. *Rvest: Easily Harvest (Scrape) Web Pages*. https://CRAN.R-project.org/package=rvest.

———. 2023. *Tidyverse: Easily Install and Load the Tidyverse*. https://CRAN.R-project.org/package=tidyverse.

Wickham, Hadley, Jennifer Bryan, Malcolm Barrett, and Andy Teucher. 2023. *Usethis: Automate Package and Project Setup*. https://CRAN.R-project.org/package=usethis.

Wickham, Hadley, Romain François, Lionel Henry, Kirill Müller, and Davis Vaughan. 2023. *Dplyr: A Grammar of Data Manipulation*. https://CRAN.R-project.org/package=dplyr.

Wickham, Hadley, Jim Hester, and Jennifer Bryan. 2023. *Readr: Read Rectangular Text Data*. https://CRAN.R-project.org/package=readr.

Wickham, Hadley, Jim Hester, Winston Chang, and Jennifer Bryan. 2022. *Devtools: Tools to Make Developing r Packages Easier*. https://CRAN.R-project.org/package=devtools.

Wickham, Hadley, Jim Hester, and Jeroen Ooms. 2023. *Xml2: Parse XML*. https://CRAN.R-project.org/package=xml2.

Wickham, Hadley, Davis Vaughan, and Maximilian Girlich. 2023. *Tidyr: Tidy Messy Data*. https://CRAN.R-project.org/package=tidyr.

Wulff, S, A Stefanowitsch, and Stefan Th. Gries. 2007. "Brutal Brits and Persuasive Americans." *Aspects of Meaning*.

Xie, Yihui. 2023a. *Bookdown: Authoring Books and Technical Documents with r Markdown*. https://CRAN.R-project.org/package=bookdown.

———. 2023b. *Tinytex: Helper Functions to Install and Maintain TeX Live, and Compile LaTeX Documents*. https://github.com/rstudio/tinytex.

A Data

...

B Feedback 💬

Thank you for taking the time to read through this book . I really value your opinion and I would love to hear your thoughts as I continue to make progress.

Where to review

Chapters that are ready for feedback will appear with the following callout.

> **Draft**
>
> Ready for review.

Those that are not ready will appear with the following callout.

> **Caution**
>
> Under development.

In a few cases a chapter will be ready for review, but I'll still be working on the exercises, callouts, *etc.* In those cases, the items that are still under development will be marked with the 🔧 icon.

What to look for

As you read over the draft, I'd appreciate your feedback in the following areas:

1. Clarity and Comprehensibility
   I'd love to know if you think the content is clear and easy to understand. Do you think the concepts are broken down enough? Are the examples helpful? If anything seems too jargon-y or confusing, definitely let me know.
2. Consistency
   It's pretty important to keep things smooth. So, keep an eye out for any inconsistent writing styles, terminology, or layout. If something seems off, I'd appreciate it if you point it out.

3. Relevance
   Does the material match the current standards and knowledge? Will it the topics and questions be of interest to linguists? If something feels outdated or irrelevant, don't hesitate to mention it.

4. Engagement
   I'm not looking to drop a boring read on people. So, as you're going through it, think about whether it holds your interest. Maybe the prose needs more life, the examples need to be more diverse, or the exercises could be more or less challenging. If you have any ideas, I'm all ears.

How to submit feedback

Depending on your preference, you can submit feedback in one of three ways:

- hypothes.is[1]
  This is the easiest way to submit feedback. Join the "qtal_feedback" annotation group and just highlight the text you want to comment on and click the "Annotate" button. You can also add comments to the right sidebar.
- GitHub issues[2]
  This book is hosted on GitHub, so you can submit feedback directly through the issues page for the repository. Just click the "New issue" button and fill out the form. You'll need a GitHub account to do this.
- Email me at francojc@wfu.edu[3]
  If you'd rather not use the other options, you can always email me directly. Just make sure to try to include references to the specific parts of the book you're referring to. A link or section number will do.

Thank yous!

I want to thank you beforehand for your willingness to help me out. I really appreciate it. I also want to thank you in print. Please give me the name you would like to appear in the Acknowledgements section. If you'd rather not be acknowledged in the final version of the book, please let me know.

[1] https://hypothes.is/groups/wppaYKxy/qtal-feedback
[2] https://github.com/qtalr/book/issues
[3] mailto:francojc@wfu.edu

Index