Computer Specifications and Setup

Before beginning any analysis, it is important to make sure the computing environment is ready for the tasks ahead. This chapter introduces the basic setup required to follow the examples in the book, including installing essential R packages and managing the computational resources needed to work with large-scale textual datasets.

This setup reflects one of the central challenges of the book itself. Our aim is to teach how to perform text mining at scale—an approach that, at the time of writing, remains uncommon and experimental in the practice of digital history—while also preparing readers to handle the practical constraints involved in working with large historical corpora.

Because historical datasets vary widely in size—and because readers will be working on many different types of computers—this chapter also offers guidance for managing limited memory or processing power. When necessary, readers can load smaller samples of the data to practice the methods. These sampled outputs may differ from the full visualizations and results presented in the book, but they will still allow readers to follow the workflow and understand the techniques.

Installing Packages

We use several packages throughout. A description of these packages is given in the book as well as in Appendix A, "Other Essential R Packages for Computational Historical Analysis." We install them all here so we can seemlessly use the libraries in the book.

To use a function from another package, you must first install the package with `install.packages()`. Here is how you would install the packages used in this chapter with `install.packages()` before loading them with `library()`.

```
# For core data science and visualization packages, central to this book
install.packages("tidyverse")

# For tokenizing, cleaning, and computing tf-idf
install.packages("tidytext")

# For high-performance tools to work with large data tables
install.packages("data.table")

# For a comprehensive text analysis toolkit
# (e.g., keywords-in-context, tokenization, document-feature matrices)
install.packages("quanteda")

# For handling and manipulating dates and times
# (useful for organizing historical data chronologically)
install.packages("lubridate")

# For stemming and lemmatization
# (reducing words to their dictionary or root forms for cleaner analysis)
install.packages("textstem")
```

Visualization-only packages:

```
# For creating word clouds from text data
install.packages("wordcloud")

# For formatting numbers, colors, and axes in visualizations
install.packages("scales")

# For ridge plots (density plots stacked by category)
install.packages("ggridges")

# For non-overlapping text labels in ggplot2
install.packages("ggrepel")

# For accessible and colorblind-friendly palettes for visualizations
install.packages("viridis")
```

Installing packages for document formatting and reporting.

```
# The core engine for rendering R Markdown documents
install.packages("knitr")

# For creating elegantly formatted tables in HTML and PDF
install.packages("kableExtra")

# For arranging multiple tables, plots, or graphics on a page
install.packages("gridExtra")
```

Install packages for webscraping and data extraction:

```
# Tools for scraping data from websites
install.packages("rvest")

# Tools for working with and parsing XML or HTML documents
install.packages("xml2")

# Install tools for accessing and working with U.S. Department of Justice datasets
install.packages("usdoj")
```

Install TMHA data packages:

Many packages can be installed directly from the Comprehensive R Archive Network (CRAN) using `install.packages()`. However, many specialty, research-oriented packages—such as those used in this book—are hosted on GitHub instead. Packages hosted exclusively on GitHub may be there because they change rapidly, support specialized workflows, or, as in our case, include larger datasets that CRAN is not designed to support.

When a package is hosted on GitHub, it can be installed in several ways. Fpr this book, we try to make the installation of our packages as easy as possible while working on as many computers as possible. To do this we provide special installation scripts that automate downloading and installing the required files. These scripts are accessed using `source()`, which tells R to fetch and run the installation code directly from the provided URL. For example:

```
source("https://raw.githubusercontent.com/Democracy-Lab/hansardr/master/tools/install_hansardr.R")
library(hansardr)

source("https://raw.githubusercontent.com/Democracy-Lab/tmha.data/main/tools/install_tmha.data.R")
library(tmha.data)
```

In RStudio, you can also verify whether a package is already installed by checking the Packages tab in the lower-right panel. This tab displays all packages currently available in your R library. If a package appears in that list, it is already installed; if not, you can install it using `install.packages()`. You can also load an installed package by checking the box next to its name in this tab, which is equivalent to calling `library()` in your script.

When installing code or data from a repository such as GitHub, we can use a special installation script that we created for this book.

If RStudio prompts you to restart R while installing a package, click Yes to allow the restart, and then re-run the installation command so the package loads into the fresh session.

As we explained in the introduction, packages only have to be installed once, but they need to be reloaded for each new R session using `library()`. Only once we have run `library(hansardr)` will the data be available for our use.

```
install.packages("text2vec")  # Install text2vec
```

```
library("httr")
library("fs")
library("pdftools")
```

We can test the installation of any of these packages using the `library()` function like so:

```
library(hansardr)
```

if the code returns something like `Error in library(hansardr) : there is no package called 'hansardr'`, the package is not installed.

Sometimes version issues ..

```
packageVersion("hansardr")
```

```
## [1] '1.0.0'
```

Update a pacakge:

```
install.packages("tidytext")
```

```
update.packages()
```

Handling Resources

The minimum resources

If at any point it is too much bog down, can take a sample. the sample will not necessarily be historically correct or match with the visualizations in the book.