

UCSF

Advancing Digital Health Humanities Institute

Title

DIGITAL HEALTH HUMANITIES TOOLKIT

Permalink

<https://escholarship.org/uc/item/8ps7d6hc>

Author

Agostini, Caterina

Publication Date

2026-06-29

DIGITAL HEALTH HUMANITIES TOOLKIT

1. Computational Text Analysis

Caterina Agostini, Ph.D.

These lesson plans are part of the Digital Health Humanities toolkit curated by UC San Francisco, after a National endowment for the Humanities Institute held at UC San Francisco (2024—2026).

The original training materials used for in-person and online sessions are available online here: <https://bit.ly/SFTextAnalysis>

Topics

- Defining text analysis
- Examining words and contexts
- Experimenting with text mining

Learning Objectives

- Exploring text analysis methods
- Browsing a textual database
- Running text analysis on a textual database
- Applying computational methods to research and teaching
- Learning about ongoing and completed digital scholarship projects.

Text Analysis: Principles and Goals

At its most basic level, text analysis involves counting words and analyzing their frequency. Text analysis typically works best with large datasets, which is why texts are often compiled around one million words. A collection of machine-readable texts is known as a **corpus**. Multiple collections are referred to as **corpora**.

For more accurate insights, especially when looking for patterns across genres or languages, having hundreds of thousands to a million words and more is often recommended (Large Language Models or LLMs). When choosing texts for a corpus, it is important to consider what to include or exclude, and why.

Here are some examples of corpora:

- All works by a specific **author**; all works by a medical humanist, for example, Rita Charon's books or articles
- All works on a topic by **more than one author**, in a collaborative capacity or not; for example, the Mary Lasker papers at Columbia University, preserving correspondence, reports, bulletins, clippings, photographs, awards, and printed material from 1940 to 1993
- All works within a particular **genre** or subgenre; for example, medical records of a clinic
- All mentions of a specific **concept** (e.g., a corpus of #MedHum tweets).

While counting is apparently straightforward, deciding what to count and why is much more challenging. In text analysis, there are two main approaches:

1. Identifying **specific words or phrases** to study and looking for patterns
2. Automating **pattern** detection, allowing the computer to identify relationships and present findings.

Each method has its advantages and disadvantages, and the most interesting research often combines both approaches. Using machine learning and unsupervised methods provides a broad overview of language use across large corpora, offering fresh perspectives for **close reading**. Text analysis is an **iterative process**. Ultimately, text analysis is most effective when the detail you seek exceeds what you can manually analyze as a traditional reader, allowing you to explain the patterns you uncover.

Key Terms

Keyword-in-Context (KWIC) Analysis: Provides a list of a specific word or phrase in context (up to five words in each direction is common). This approach is best for pattern identification and close reading.

Lexical co-occurrence or collocation: Observes clusters of terms which are likely to appear together in each dataset, based on statistical relationships. This is good for getting a sense of proximity for a specific term or finding specific word associations. Topic modeling is based on this principle.

Most Frequent Word Analysis: Finds the most frequent terms of a given dataset and highlights the small function words which make up the bulk of language. Good for identifying unique stylistic fingerprints and keeping track of presence and absence.

N-grams: Observes clusters of terms which appear together in each dataset. This is very good for identifying common phrases in a particular genre, but also stylistic features which are unique to a specific author.

Sentiment Analysis: A computational method used to identify the emotional tone or attitude in a piece of text, often categorizing it as positive, negative, or neutral.

Topic modeling: A statistical method used to uncover themes or topics within a collection of documents. It works by automatically grouping words that frequently appear together across multiple texts, aiming to identify clusters of words that represent distinct topics.

Word Vector Modeling: Like lexical co-occurrence or collocation, this approach looks for terms which are likely to appear together in each population, and projects terms into multi-dimensional space to model semantic relationships between words at scale.

An Early Example: The “Index Thomisticus”

A pivotal project that marked the inclusion of digital scholarship began with Father Roberto Busa, S.J., and his Index Thomisticus (in Latin, “Index of St Thomas Aquinas”). Busa made a concordance of the corpus, or collected works, of St Thomas Aquinas (<https://www.britannica.com/biography/Saint-Thomas-Aquinas>), who is a Doctor of the Church. Aquinas, an Italian Dominican friar and priest, is regarded as the foremost Scholastic thinker due to the profound impact of his theological treatises, commentaries, and philosophical writings. The Corpus Thomisticum, the collected works of St Thomas Aquinas, encompasses approximately 11 million words. Father Busa’s work represents an initiative in **computational linguistics**—a field now recognized as part of **Digital Humanities (DH)**.

Previous concordances, such as the one compiled by James Strong, the Concordance of the Bible (https://www.google.com/books/edition/The_Exhaustive_Concordance_of_the_Bible/3TBGAQAAMAAJ?hl=en&gbpv=0) (1894), were created manually and listed thousands of words, including the 8,674 Hebrew words of the Old Testament and the 5,624 Greek words of the New Testament. This project took thirty-five years to complete. Knowing about the work of philologists, Father Busa believed there had to be a more efficient way to generate textual information, particularly with the aid of technology.

The Index Thomisticus project is now accessible online (<https://www.corpusthomisticum.org/it/index.age>).

From Corpus Linguistics to IBM

Father Busa recognized the **scale** of the task ahead, especially if he attempted to produce a **concordance** by hand—an extensive process that would involve systematically ordering all words in a text alongside relevant metadata. In 1946, he became interested in creating a concordance of the corpus he studied—the works of [St. Thomas Aquinas] (<https://www.britannica.com/biography/Saint-Thomas-Aquinas>).

Building on his doctoral research (1949), Father Busa planned to track **word occurrences** and the decision to work computationally gained traction after he began collaborating with **IBM** in 1949 (Busa 1951). Father Busa’s project, thus, combined theological studies with computer-based technologies. The Index Thomisticus aimed to make a comprehensive concordance of all St.

Thomas Aquinas's works and related texts, culminating in a series of fifty-six printed volumes completed in 1980.

From Punch Cards to Machine-Readable Texts

Father Busa made significant efforts to collaborate with a leading technology company of his time. Through his Jesuit connections at Fordham University, he arranged a meeting with Thomas J. Watson, the President of IBM. This meeting led to his collaboration with Paul Tasman, an IBM engineer who was knowledgeable about contemporary data processing equipment. **Punch cards**, previously used for data storage, served as a medium for storing, sorting, and reporting data processed by punched card equipment and computers.

Rockwell and Sinclair argued that the technology utilized by Busa and Tasman represented key innovations facilitating the application of data processing to unstructured text data. Consequently, Busa and Tasman's **humanistic perspective on technology** and their goals influenced the algorithm they developed for the final product (Rockwell and Sinclair 2020).

Their process with the punch cards generally followed these steps:

- **Keypunch:** Encoding words and phrases as patterns of punched holes on a card. This was done in two stages:
 - First, a scholar forms phrases
 - Second, a punch card operator encodes phrases using keypunch technology.
- **Reproducing Punch:** Copying the encoded information onto multiple cards, with each card representing a single contextual entry.
- **Sorter:** Alphabetizing the decks of cards and grouping them by related classifications.
- **Tabulator:** Reading the data from the cards and printing it onto continuous-feed paper.
- **Collator:** Checking the accuracy of the transcriptions using multiple sets of cards, akin to the collation of variant texts in traditional textual studies.

Father Busa sought to go beyond the contextual interpretation of the texts so that he could renew the study of philology. By rigorously examining the semantic and lexical properties of St. Thomas Aquinas's writings, he devised his own algorithmic process for recording textual information using handwritten cards, reflecting a deeply traditional approach to text. Steven Jones discussed Busa's technological innovation as follows:

For Busa, mechanization was to serve hermeneutics. He aimed to interpret, to reveal meaningful patterns, different dimensions of the language. It's just that some dimensions were too extensive (while their evidence was too minute) to be grasped by the unassisted eye and mind of the reader across an oeuvre of over ten million words. Philosophical and theological questions drove his linguistic research, and this led him to develop the complex process of literary data analysis. Looked at another way, in the Jesuit tradition, Father Busa sought the divine presence in all

things; IBM's punched-card machinery was at first only a means to this ultimate end (Jones 2016).

Father Busa recognized the significance of **deep reading** and connected it to the **Jesuit tradition of scholarship**. He dedicated many years to his Index Thomisticus project, ultimately producing over fifty volumes containing a concordance of Aquinas's works. During this process, he established a **linguistics center in Gallarate, Italy**, which trained many technical workers in the complex IBM technology of the time. This center effectively served as a quasi-institute for Jesuit education in the region and connected with other institutions worldwide, including Georgetown University in Washington, D.C.

The main technology used was based on punched cards, which were modified versions of the IBM 80-column punch card, adapted to accommodate the replication and semantic information required for the concordance (Rockwell and Sinclair 2020). These developments would play an essential role in literary and textual analysis, highlighting the potential of computational power for purposes beyond numerical analysis.

Each step in the punch card process required knowledgeable and trained punch card operators, often women. Then, Busa and Tasman iterated through this procedure many times before finalizing the process that produced the Index Thomisticus.

Corpora

In addition to corpora shared by the ADHHI toolkit, here is a selection of pre-assembled corpora, available to use.

BBC Script Library, <http://www.bbc.co.uk/writersroom/scripts>.

Xiao, Richard (2008) "Well-Known and Influential Corpora", in Anke Lüdeling and Merja Kytö, Eds. **Corpus Linguistics: An International Handbook**. Berlin: Mouton de Gruyter. Volume 1: 383–457. See also the [University of Lancaster's Corpus Survey] (<https://www.lancaster.ac.uk/fass/projects/corpus/cbls/corpora.asp>).

1a. Text Analysis with AntConc

AntConc is a corpus analysis toolkit to examine words, texts, and corpora. It is a free corpus analysis tool used for studying text collections. For linguistic research and text analysis, it helps users gain insights into language use and structure. AntConc analyzes and visualizes text data through various functions, including word frequency counts, keyword extraction, concordance, and collocation analysis.

Download

[Antconc](<https://www.laurenceanthony.net/software/antconc/>) from the [AntLab Website] (<https://www.laurenceanthony.net/>).

Unzip the download, if necessary. Next, launch the application.

[Updated releases](<https://www.laurenceanthony.net/software/antconc/releases/AntConc324/>) are published on a regular basis by AntConc's creator, [Laurence Anthony] (<http://www.laurenceanthony.net/>).

[Laurence Anthony's guide to help users understand how to use AntConc.](https://prod-files-secure.s3.us-west-2.amazonaws.com/7b6b9cc5-d502-45c7-bc5d-561183134672/931a28b4-9fa4-4fda-9236-cd21771a0289/Anthony_Tutorial.pdf)

Laurence Anthony's guide to help users understand how to use AntConc.

Datasets

We can add files in the format of plain text (.txt), Word (.docx), and PDF (.pdf) files.

ADHHI datasets, <https://ucsf.box.com/s/obaj39x7v0be56aj9shclvw60arr06fx>

No More Silence, <https://ucsf.app.box.com/s/ifdndlmnje0xme8jdg3i697s51zznect>

Industry Documents Library, <https://ucsf.app.box.com/v/IDL-DataSets/folder/78644252849>

Corpus Analysis

Corpus analysis is a form of text analysis which allows you to make comparisons between textual objects at a large scale. **Distant reading** enables us to notice things that we might overlook when reading. If you have a collection of documents, you might want to find frequently recurring phrases in your corpus; statistically likely/unlikely phrases for a particular author or genre of text; grammatical structures; also, examples of a particular term or idea across many documents, and much more. Corpus analysis is valuable for testing hypotheses about texts and for cross-checking results from other digital methods.

This tutorial provides an overview of methods rather than a rigid step-by-step formula, and following the exact order is not always necessary. Corpus linguistic methods are customizable, so remember we need to think about how each step helps answer specific research questions.

A webtool we discussed earlier this week, [Voyant Tools](<http://voyant-tools.org/>), offers similar options for exploring text analysis methods.

The AntConc User Interface

When AntConc opens, it shows an interface with the following tabs:

- KWIC: Keyword in Context, which is a concordance view.
- Plot: A visualization of your KWIC search along a bar, showing the position and frequency of words within the text.
- File View: Showing the broader context of a finding.
- Cluster: Words frequently found in the same context(s).
- N-gram: Given that n is a number you decide, this feature shows common terms.
- Collocate: Given a certain distance (number of words away from a specific word), words statistically likely to be found in the same context(s).
- Word: A list of all the words in the corpus.
- Keyword: Comparing most frequent, significant words across two corpora.
- Wordcloud: A visual representation of textual data in which the size of each word reflects its frequency or significance.
- ChatAI: This feature seems to be still in progress. There is a disclaimer, “Remember that OpenAI API calls will be charged to your account. Use the ChatAI tool with caution.”

Start with File, then Open File as Quick Corpus or Open Corpus Manager.

Add more than one file or a folder. Click Create so you can have a corpus loaded. Click Create. You will have a confirmation message, “Your corpus has been created.”

This is your target corpus. You can always check what has been uploaded (and is being analysed) on the left, in the window to see all corpus files—that would be your target corpus.

Important: For all search tabs, except the Word tab (marked with an asterisk in this list), you need to type a word or more, then click Start.

Search operators for Keyword-in-Context (KWIC)

The * operator (for zero or more characters) can help find variations of the same word. For example, the singular and plural forms of nouns, or words with similar spellings but different meanings, but also irregular verbs in various tenses.

polic* would track police, policy, policies, etc.

s*t retrieves sit, sat

sit* finds sit, sits, sitting, etc.

The ? operator for d??r finds dear, deer, door, etc.

The | Operator (“or”)

The | Operator (“or”) can search alternative words, for example yes|no.

Saving Outputs

While running AntConc, you can generate output files for reference. It is good practice to use descriptive filenames (for example, “med_hum_20241106.text”).

1b. Text Analysis in R

Text analysis examines large amounts of textual information. R is a programming language designed primarily for statistical computing and data analysis.

Text analysis, also known as text mining, refers to the computational examination of large volumes of documents, focusing on language, context, and patterns. It is a methodological approach that is applicable to various fields, including the humanities, sociology, communications, and computer science. Common text analysis case studies involve books, articles, news, and tweets. Text mining consists of these steps:

- Creating a dataset
- Exploring text analysis strategies
- Interpreting findings.

Download

R is a programming language designed primarily for statistical computing and data analysis. It offers a wide array of statistical techniques, including linear and nonlinear modeling, time-series analysis, classification, and clustering. Additionally, R supports graphical representations of data, making it a popular choice for making plots and charts.

Highly extensible, R allows users to create packages that add new functionalities. While it is widely utilized by statisticians and data analysts, it has been adapted to answer research questions in the humanities and digital humanities.

[<https://posit.co/products/open-source/rstudio/>](<https://posit.co/products/open-source/rstudio/#Desktop>)

RStudio is an integrated development environment (IDE) for R, designed to make it easier to write and run R code. It provides a user-friendly interface that includes features such as syntax highlighting, code completion, and a built-in console. RStudio also offers tools for data visualization, package management, and project organization, making it easier to manage R scripts and files. Additionally, it supports RMarkdown for creating dynamic reports that combine code, output, and text. Overall, RStudio enhances the R programming experience by providing a comprehensive environment for coding, analysis, and reporting.

Another option is to access R and RStudio through university-licensed software from any device.

We can run R without RStudio, but we cannot run RStudio without R.

If you want a fast, easy way to try R, you can use the [RStudio Cloud](<https://rstudio.cloud/>). It is a cloud-based version of RStudio that allows users to write and run R code in a web browser without needing to install software on their local machines.

This web browser version of RStudio looks and functions just like the desktop version, and it will save data between sessions as well.

All it takes is a Google or GitHub account to log on.

Setup

First, [download R](<https://cran.rstudio.com/>) for your operating system. R is the programming language.

Second, it is recommended to [download RStudio](<https://www.rstudio.com/products/RStudio/>). RStudio is an integrated development environment (IDE) that makes scripting in R much easier.

Both R and RStudio are free and open-source software.

Caption: This is what the RStudio interface looks like when you open it.

When you launch RStudio, you'll notice the default panes:

- Console (left, with the > prompt)
- Environment/History/Connections/Tutorials (tabbed, upper right)
- Files/Plots/Packages/Help/Viewer/Presentation (tabbed, lower right).

Programming is helpful in the humanities because you understand more deeply what computers do to your data. By understanding the workings of an algorithm, you can better justify your claims as well as understand the ways algorithms may distort the data.

The work of data analysis becomes transparent. Since we're writing code, we can trace the steps we made to make something.

We start to correct for the problem of tools shaping research questions. In other words, we build the tool to fit our questions.

Install R packages

To install R packages, open RStudio, then type and run these lines in the console:

```
install.packages("tidyverse")
```

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

```
install.packages("tokenizers")
```

```
install.packages("quanteda")
```

```
install.packages("quanteda.textmodels")
```

```
install.packages("quanteda.textstats")
```

```
install.packages("quanteda.textplots")
```

```
install.packages("readtext")
```

```
install.packages("devtools")
```

```
install.packages("quanteda.corpora")
```

```
install.packages("newsmap")
```

```
install.packages("seededlda")
```

tidyverse provides convenient tools for reading in and working with data sets.

tokenizers contains the functions that allow us to split text data into words and sentences.

Depending on your system setup, these may open a dialog box asking you to choose a “mirror” to download from. Select one near your current location. Then, follow and confirm download and installation as you get the automatic prompt to do so.

Now, ask R to load these packages through the ``library`` command. The ``library`` commands must be run every time you restart R.

```
library(tidyverse)
```

```
library(tokenizers)
```

Running R

```
library(tidyverse)
```

```
library(tokenizers)
```

5

Caption: Entering a numerical value in R.

```
a <- 10
```

```
b <- 20
```

```
a <- b
```

```
print(a)
```

Caption: Mathematical values in R.

More examples with mathematical operations.

```
sum(c(10,20,30))
```

Caption: Summing numbers.

One Text

One Text

To display (or print) a text, use the R-command `cat()` or `print()`. The text is considered by R as a script, so we need to have the text in quotes.

```
cat("Hello world")
```

```
print("Hello world")
```

```
msg <- "Have a good day!"
```

```
print(msg)
```

```
print("msg")
```

Caption: After writing a message, you can render it by printing it.

What differences do you notice?

Word Tokenization

Let's start text analysis with a single paragraph of text. The example here is the opening paragraph from [Henry James, *The Wings of the Dove*](<https://www.gutenberg.org/files/29452/29452-h/29452-h.htm>) (1902).

To read this into R, copy and paste the following into the R console.

```
text <- paste("She waited, Kate Croy, for her father to come in, but he kept her unconscionably, and there were moments at which she showed herself, in the glass over the mantel, a face positively pale with the irritation that had brought her to the point of going away without sight of him. It was at this point, however, that she remained; changing her place, moving from the shabby sofa to the armchair upholstered in a glazed cloth that gave at once—she had tried it—the sense of the slippery and of the sticky. She had looked at the sallow prints on the walls and at the lonely
```

magazine, a year old, that combined, with a small lamp in coloured glass and a knitted white centre-piece wanting in freshness, to enhance the effect of the purplish cloth on the principal table; she had above all, from time to time, taken a brief stand on the small balcony to which the pair of long windows gave access. The vulgar little street, in this view, offered scant relief from the vulgar little room; its main office was to suggest to her that the narrow black house-fronts, adjusted to a standard that would have been low even for backs, constituted quite the publicity implied by such privacies. One felt them in the room exactly as one felt the room—the hundred like it or worse—in the street. Each time she turned in again, each time, in her impatience, she gave him up, it was to sound to a deeper depth, while she tasted the faint, flat emanation of things, the failure of fortune and of honour. If she continued to wait it was really, in a manner, that she might not add the shame of fear, of individual, personal collapse, to all the other shames. To feel the street, to feel the room, to feel the table-cloth and the centre-piece and the lamp, gave her a small, salutary sense, at least, of neither shirking nor lying.")

text

Caption: Entering a paragraph of text into R.

Hit enter. You will see the paragraph of text “printed.”

Next, we get the ``tokenize_words`` function from the `tokenizers` package.

```
words <- tokenize_words(text)
```

Caption: Tokenizing words.

Words

Caption: Asking to make a list of words.

And this shows an output of a word list, removing all punctuation, finding individual words, and making everything lowercase.

To count words, we need to use the `length` function and associate it with the number ``[[1]]`` because a list object is one entry per document.

```
length(words[[1]])
```

Caption: Requesting to count words.

How many times is each word used in the text?

```
tab <- table(words[[1]])
```

```
tab <- data_frame(word = names(tab), count = as.numeric(tab))
```

```
tab
```

Caption: Counting word occurrences.

Can we see most common words in order?

```
arrange(tab, desc(count))
```

Caption: Ordering frequent words.

The most common words tend to be pronouns, articles, and conjunctions.

A popular technique is to maintain a list of highly used words and removing these prior to any formal analysis. The words on such a list are called “stopwords”, and usually include words such as pronouns, conjugations of the most common verbs, and conjunctions.

Stopwords (<https://github.com/koheiw/stopwords>) in several languages are available on GitHub.

Visualizing Sentences

We have seen individual instances of words. We can also see individual sentences.

```
sentences <- tokenize_sentences(text)
```

```
sentences
```

Caption: Viewing each sentence.

We can also view each sentence and count its words.

```
sentence_words <- tokenize_words(sentences[[1]])
```

```
sentence_words
```

Caption: Counting words in sentences.

Checking the size of the output directly, we can see that there are “documents” in the object `sentence_words`:

```
length(sentence_words)
```

Caption: Viewing how many words are in each sentence.

Analyzing a Corpus

Read all your files into R together. Point to where your files are.

```
input_loc <- "/Users/yourname/Desktop/corpus_text"
```

Caption: Retrieving a corpus in R.

Finding Your Dataset

Now let’s make sure you’re working in an appropriate directory on your computer.

Enter ``getwd()`` in the Console to see current working directory or, in RStudio, this is displayed in the bar at the top of Console.

In RStudio, go to Session > Set Working Directory > Choose Directory > [folder you saved to your Desktop].

The current working directory is displayed by the RStudio IDE within the title region of the Console pane. You can also check your current working directory by running the command `getwd()` in the console.

There are a few ways to change the current working directory:

Use the `setwd` R function

Use the Tools | Change Working Dir... menu (Session | Set Working Directory on a mac). This will also change directory location of the Files pane.

From within the Files pane, use the More | Set As Working Directory menu. (Navigation within the Files pane alone will not change the working directory.)

Be careful to consider the side effects of changing your working directory:

Relative file references in your code (for datasets, source files, etc) will become invalid when you change working directories.

The location where `.RData` is saved at exit will be changed to the new directory.

If you want to type the specific location for your files, you can follow these directions.

```
file.copy(from="C:/Users/[Your Name]/Documents/[Folder Name]/[File Name].txt",
          to="C:/Users/[Your Name]/Documents/[Another Folder Name]/[Same File Name].txt")
```

Stylometry

Stylometry, the study of linguistic style, makes extensive use of computational methods to describe the style of an author's text. The corpus can be split into sentences using the `tokenize_sentences` function.

```
sentences <- tokenize_sentences(text)
```

Caption: Find sentence length.

Find words in sentences.

```
sentence_words <- sapply(sentences, tokenize_words)
```

Caption: Words in sentences.

```
sentence_length <- list()
```

```
for (i in 1:nrow(metadata)) {
```

```
  sentence_length[[i]] <- sapply(sentence_words[[i]], length)
```

```
}
```

Caption: Sentences and length.

```
sentence_length_median <- sapply(sentence_length, median)
```

Caption: Sentences and style by year.

```
qplot(metadata$year, sentence_length_median)
```

More Resources

Posit (formerly RStudio) has a comprehensive guide, <https://support.posit.co/hc/en-us>

R Markdown

A script written in R is customarily stored in a plain text file with the `.R` extension. But this is an R Markdown document, and instead has the file extension `.Rmd`. It connects human language and code. Markdown is a simple formatting syntax for authoring documents.

Furthermore, the same markdown document can produce a variety of presentation outputs (e.g. HTML, PDF, MS Word). For more details on using R Markdown, see [\[http://rmarkdown.rstudio.com\]](http://rmarkdown.rstudio.com)(<http://rmarkdown.rstudio.com/>).

1c. Cleaning Data in OpenRefine

OpenRefine is a tool for cleaning and organizing messy data.

- Checking data formats and accuracy
- Managing project workflows
- Exploring linked open data authorities
- Transforming datasets
- Exploring data workflows

Working with Data

We often work with research materials, whether they are digital or physical. But turning these materials into usable data is a process where researchers choose what (and how) to focus on, shaping the way the information is understood. In this discussion, we'll look at the key factors to consider when working on a digital project, explore real-world examples, and talk about the advantages and challenges of converting materials into formats that computers can read.

Types of Sources

For this workshop, I have picked some sample primary materials. To start, we have different types of sources to turn into data, such as texts (from libraries and digital collections) and images (from libraries and museums).

Organizing Information into Spreadsheets

Organizing information into a spreadsheet is a recommended step to build a digital project around humanities so that you can have research data. Why do we need a spreadsheet from research materials? One reason could be to visualize what you have found, through a map, network analysis, a graph for trends or value distributions, and to see the implications of your claims around the data.

Exploratory questions can also be addressed in <http://hdlab.stanford.edu/breve/>, a tool currently developed for the analysis of tabular data.

- A spreadsheet is a file in which we organize information into categories.
- Rows are items of interest (people, objects, court cases, places...).
- Columns are attributes of the items (birth town, cost, date a law was established, status, language spoken, gender...).
- Cells are the boxes.

Possible file formats for spreadsheets are .xls and .csv. A comma-separated value file is particularly useful to build digital projects, for example through Palladio, a web tool for mapping: <http://hdlab.stanford.edu/palladio/>.

To see one example, let's find a list of passengers from the Ellis Island records

<https://www.statueofliberty.org/discover/passenger-ship-search/>

Let's now compare this set of information with other historical sources available online.

<https://www.archives.gov/nyc/finding-aids/passenger-lists.html#online>

From a catalog of disciplines and faculty in a university in the Renaissance and early modern period, we will see how information is organized and presented. What questions arise?

[https://www.google.com/books/edition/Gymnasium_Patavinum_Jacobi_Philippi_Toma/nFEVAAAQAAJ?hl=en](<https://www.notion.so/432b4ac8c1734f0c9932cfa3081a470e?pvs=21>)

Now we have information about passengers or scholars. What categories do we notice? How can we transcribe information?

Tips

Always keep a separate, original copy of the original, unstructured data for publication, reference, and future uses.

Always make another spreadsheet with your cleaned, structured data for visualization with columns and formats that satisfy the tool(s) you want to use.

Data should be uniform, consistent, and simple. Document all changes you make to the data to achieve this.

Build a Spreadsheet

Let's go from unstructured to structured data. Every column represents an attribute that can inform your analysis.

Diacritics, Punctuation, Special Characters, and Non-Latin Script

Avoid diacritics, punctuation, special characters, and non-Latin script if you can.

If you can't, be sure to save your files in UTF-8 encoding (Google Sheets does this automatically) to ensure your documents/data can be understood by a large variety of tools.

Excel for Mac does not natively support the import or export of UTF-8 encoded files

Import the file into Google Sheets, then save as .csv

Create a separate column with the diacritics, punctuation, special characters, and non-Latin script items if necessary.

If you need to guess or estimate, create a column to write out your rationale, and consider another column with a percentage, or score indicating how certain you are of your guess or estimate: for example, rough guess, source derived, contextually likely, source likely, mean of range.

Consistent Dates

Dates should be entered in the following format: Year-Month-Day (2020-11-18)

Years must always be entered as four digit integers. Thus, the years between 0 and 1000 should be rendered as 0001 and 0999.

BCE dates should follow the same format, but preceded by a minus sign (0200-01-01)

Be sure to specify that the column reads as “Text” rather than “dates” (Format Cell > Text), so Excel/Google Sheets does not change the dates into its own format

=TEXT(B2, “yyyy-mm-dd”)

LatLong and decimal degrees

Location information must be rendered as latitude and longitude

Use three geocoded columns to work around the preferences of various mapping tools: LatLong, Latitude, and Longitude.

If you have a list of place names, but no associated coordinates, you can use this geocoding tool:

<https://gis.ucla.edu/geocoder/>, or Google Maps for individual location information

In Google Sheets, you can also try Geocode by Awesome Table (Add ons > Geocode by Awesome Table > Start Geocoding)

If no modern address is associated with your location, you can manually pin a location on a Google Map to get this information, which is useful for sites with no address.

Images

You should create a column in your spreadsheet with the URL to an image you want to appear in a pop-up. The URL must be publicly accessible and reference the image, not the page with the image.

Transcription Tools

Transkribus.eu is being developed to recognize handwritten records and transcribe them into digital form. It requires 100 pages of transcription to create an engine unique to your scribe.

Another tool under development is Data Scribe, an Omeka S plugin:

<https://rrchnm.org/news/from-historical-sources-to-datasets-a-preview-of-datascibe/>

Recogito (<https://recogito.pelagios.org/>) can be used to find location information in digital texts. It runs its Name Entity Recognition Engine on your text, using English, but other languages and engines can be integrated, and it correlates words in your text with geographic information from four gazetteers.

To Map and Annotate Your Materials

You can use Recogito for individual or team projects, when you need to collect and categorize people, places, events, or other elements that you can add as annotations. Mapping is an option, too, given that geographic information is available through built-in gazetteers that the Recogito team connected to the web tool.

Preserving and Documenting Digital Work

The Digital Documentation Process (DDP) guides scholars through a catalogue record, a persistent identifier, and an archiving dossier narrative. The DDP guidelines and project were developed by Laura Morreale, so that following the DDP allows to make DH scholarship findable and citable, durable, and it enhances the value of DH work. Go to <https://digitalhumanitiesddp.com/> for more information.

You can also share your publications and presentations on an open-access repository such as <https://zenodo.org/>

More Resources

A quick how-to guide introducing Recogito is online at <https://recogito.pelagios.org/help/tutorial>

For more digital project concepts, see the Cambridge Digital Humanities website:

[<https://www.cdh.cam.ac.uk/lab/concept-1>](<https://www.notion.so/bd80a7a3cb594090b8de47bc9a77bdf1?pvs=21>)

For teaching purposes, see a blog post I wrote: [[https://dh.rutgers.edu/digital-](https://dh.rutgers.edu/digital-humanities-tools-in-online-humanities-classes/)

[humanities-tools-in-online-humanities-classes/](https://dh.rutgers.edu/digital-humanities-tools-in-online-humanities-classes/)](<https://www.notion.so/b98ef7688cff4e90b2351a461bcbb91f?pvs=21>)

Download

Download [OpenRefine](<http://openrefine.org/>)

Install it.

Open a dataset:

- Basic data on all accessioned works in the Metropolitan Museum of Art collection, New York City. Available for unrestricted use under [Creative Commons Zero (CC0)] (<https://creativecommons.org/publicdomain/zero/1.0/>).
- <https://www.metmuseum.org/about-the-met/policies-and-documents/open-access>
- Download the Met data, updated on a regular basis on Github, <https://github.com/metmuseum/openaccess>
- A numismatic dataset from MANTIS that includes the Roman, Byzantine, and Islamic coins from between 400 and 500 CE. This is an interesting collection because it uses regions and the earliest, or latest possible date for items, [<https://raw.githubusercontent.com/leadr-msu/hst251-NumismaticViz/master/MANTISRomanByzIslamic400-500-cleaned.csv>](<https://www.notion.so/Cleaning-Data-in-OpenRefine-12d805d17d8d80b599f1f6629718f37e?pvs=21>)
- A dataset on the French “encyclopédistes” shared by Melanie Conroy
- [<https://ageofrevolutions.com/2017/09/25/visualizing-social-networks-palladio-and-theencyclopdistes-pt-i/>](<https://ageofrevolutions.com/2017/09/25/visualizing-social-networks-palladio-and->)
- A dataset from the Getty Provenance Index Databases, <https://github.com/chnm/doingdh>

Home page for OpenRefine: <http://openrefine.org/>

OpenRefine is a powerful tool for cleaning many kinds of data – numeric and textual, and can import from and also export to several useful formats. In this workshop we will focus on textual and numeric data that originates in tabular form, and will go through some basic views and transformations. After this introduction to OpenRefine, you can learn more from the many tutorials and examples on the web.

“Cleaning data” means we are **modeling** and **remodeling** information.

Install from <http://openrefine.org/download.html>

Documentation

- OpenRefine home page: <http://openrefine.org/>

Tasks

1. Import file and types of source file

Create a new project

Get source file

- • Example of use: <http://dataist.wordpress.com/2012/04/10/tutorial-using-google-refine-to-clean-mortgage-data/>

• in depth tutorial: [<http://www.propublica.org/nerds/item/using-google-refine-for-data-cleaning>] (<https://www.propublica.org/nerds/using-google-refine-for-data-cleaning>) - •

<http://programminghistorian.org/lessons/cleaning-data-with-openrefine> Basically the first few chapters of the OpenRefine book.

- • Miriam Posner's OpenRefine tutorial: <http://miriamposner.com/classes/dh101f17/tutorials-guides/data-manipulation/get-started-with-openrefine/>

- • Grel reference: <https://openrefine.org/docs/manual/grelfunctions>

1. Import file and types of source file

Create a new project

2. Explore the dataset using Filtering and make some simple changes – OpenRefine works primarily on columns. Use dropdown arrow to available actions.

- In the Notes column, what can you find out about which languages common? (*Hint:* use some filters)
- Make the Notes column more readable by converting the cells to mixed case. (*Hint:* From column dropdown, see Edit Cells>Common Transforms)
- Look at all the other cell options

3. History and Undo (Tab on the left)

4. Facet / Clustering OpenRefine is an open-source tool designed for working with messy data. It helps users clean, transform, and explore large datasets through an intuitive interface. Key features include data cleaning (such as removing duplicates and correcting inconsistencies), data transformation (including reformatting and restructuring data), and data exploration (allowing users to filter and visualize data). OpenRefine also supports reconciliation with external datasets, enabling users to match their data against other sources for enrichment. Overall, OpenRefine is a powerful tool for anyone looking to improve the quality and usability of their data.

2. Introduction to the International Image Interoperability Framework (IIIF)

In this lesson plan, we will explore the International Image Interoperability Framework (<https://iiif.io>) and the work of the IIIF community to create universal standards for describing and sharing images online. With common viewing platforms, we can obtain interoperable digital image content to display, edit, annotate, and share images on the web.

Tool-Specific Components

Research libraries and image repositories have been collaborating to produce an interoperable technology and community framework for image delivery called IIIF, an acronym standing for International Image Interoperability Framework. IIIF images can be artworks, photographs, maps, music scores, books, newspapers, manuscripts, scrolls, single sheet collections, and archival materials. IIIF also handles audiovisual materials, with annotations for sections of the time dimension.

IIIF develops an open Application Programming Interface (API), an intermediary software that allows two applications to communicate to each other, while implementing them in software that exposes interoperable content. Through image servers and web clients, anyone can view, compare, manipulate, and annotate images. Therefore, there are hundreds of different implementations that are called APIs. Those APIs are open source and anyone can implement.

When we explore IIIF, we need the following tool-specific components:

- Image API (Application Programming Interface), to deliver the pixels of an image in a simple and intuitive way that easily specifies the region, size, rotation, quality characteristics, and format of the image
- Presentation API, to allow for viewing
- Content Search API, with the option to search within the transcription (OCR, Optical Character Recognition) associated with a document
- Authentication API, to allow lightweight handling of authentication workflows
- IIIF Discovery API to allow users to easily find and engage with content available via existing IIIF API

What is IIIF?

There are two meanings associated to IIIF:

- A set of open standards that expands uses of images from digitized collections, and implements viewing, comparing, manipulating, deep zooming, editing, and annotation capabilities
- The community of users and developers who collaborate on IIIF, such as libraries, museums, archives, software companies, and other organizations that create the IIIF specifications and advocate for the use of IIIF.

Goals

The goals of IIIF are the following:

- To have high-quality access to images hosted around the world
- To define a set of common application programming interfaces that support interoperability between image repositories
- To develop and document shared technologies for an unprecedented user experience, including the possibility to share, cite, and embed digital images that are IIIF compliant.

Finding IIIF Resources

Now that we have a definition for IIIF and an understanding of its potential uses, let's find images

that are IIIF compliant. Here are some websites to browse:

- A guide to find IIIF resources has been recently published by the IIIF community: <https://guides.iiif.io>
- Many, many more collections: https://bit.ly/iiif_collections

Some institutions providing IIIF compliant images are the following:

- Bayerische Staatsbibliothek: <https://app.digitale-sammlungen.de/bookshelf>
- Stanford Special Collection OCLC IIIF Explorer, which is currently experimental: <https://researchworks.oclc.org/iiif-explorer>
- Biblissima, a source that aggregates many European libraries: <https://iiif.biblissima.fr/collections>
- Europeana, Europe's platform for digital cultural heritage: <https://www.europeana.eu/en>
- The Vatican Library: <https://digi.vatlib.it>
- Harvard: <https://library.harvard.edu/digital-collections>
- Yale Center for British Art: <https://collections.britishart.yale.edu>.

IIIF Viewers: Mirador

Open an instance of Mirador at this link: <https://projectmirador.org> or try a live demo <https://iiif.gdmrdigital.com/mirador/>.

Zooming and Comparing

One common use for IIIF images is to zoom in for specific details, as well as a side-by-side comparison. Using the Mirador site listed above (<https://projectmirador.org/demo> or <https://projectmirador.org>), let's build a workspace view that compares two or more images.

Click one image showing in the Mirador Viewer here (for example, artworks by Van Gogh), or click

on the 'plus' button to add or change images.

You can also add images from other repositories. Click 'Add Resource' and copy and paste the IIIF

manifest we have practiced retrieving earlier, while we were exploring collections.

You can change the layout for your images in Mirador Viewer. Your images layout depends likely on how many images you would like to display at once. Click Windows view and thumbnail display (upper right corner) and select the layout you prefer.

Find an item, or more items from the collections listed above that you want to explore.

Find the IIIF badge or the IIIF manifest URL and drag it into Mirador Viewer. There are ways to find a manifest URL, which is easier to find in some websites. For example, here we can practice with the Yale Center for British Art (<https://collections.britishart.yale.edu>). An intuitive approach for all IIIF repositories is to look for the IIIF logo.

Annotations

* This feature is supported by some version of Mirador. Select a Manifest by clicking on one of the rows or add your own manifest (Add new object from URL). Click on Toggle Annotations, which looks like a speech bubble. Choose the way you want your annotations to look. Go to the image, select an area, and draw your annotation space there. Type a description and, if you want, tags too. Click Save.

Click the speech bubble on the top left:

- Speech bubble, or hide annotation buttons
- Pointer, to select existing annotations and change the size or edit the content
- Rectangle, to draw a rectangle annotation
- Oval, to draw an oval annotation
- Freeform, to have more creativity with the shape of the annotation
- Polygon, to create a shape with straight sides
- Pin, to drop a pin as the target of the annotation
- Thickness, to change the thickness of the shape's line
- Line color, to change the color of the shape's line
- Fill, to fill the shape with a color.

After selecting and drawing a shape on the image, you can see a box where you can add content for the annotation. Now you can see more buttons to format the content of the annotation: bold, italics, inserting a link, inserting an image, inserting a video or audio file, and also the option to clear formatting and change the orientation of the text (left to right or right to left). Save and add tags if you want.

Annotating through Storiies by CogApp

Another way to combine annotations and images for research or teaching purposes is with tools developed to guide viewers through a curated experience. We can do this through Storiies (with three i's for IIIF), developed by CogApp.

We can see one example of a narrative presentation at this link:
<https://storiies.cogapp.com/viewer/aeqi/The-Stoning-of-Saint-Stephen>

Storiies can use IIIF to add annotations to images. For now, let's upload a JPEG file and annotate that image next. You could also start using Storiies Editor by pasting in a manifest, however, there is currently a technical issue that CogApp has been trying to fix.

Storiies is free to use. All services including image uploads and access to created stories are provided 'as is'. They make no warranty as to future availability of images and stories and also reserve the right to restrict access and change the conditions of service at any time.

Here is an example from a collection by Danielle Reay and her colleagues at Drew University who showcased manuscripts through Storiies for pedagogical purposes:
<https://storiies.cogapp.com/viewer/391d3/Drew-University-Shahnama-Paper-Conservation>

IIIF Image API through the Internet Archive

You can upload your pictures and have one IIIF Image API service through the Internet Archive.

* You can delete your images after 30 days if you just want to practice now.

1. Register for a user account at <https://archive.org>
2. Upload your image
3. Upload one image per item
4. Select Community Image collection
5. On the details page, get the identifier for the image. That might look like this: img-1111_222222
6. Some of the IIIF images might appear very small even if you uploaded a large image. In that case, add the filename to the URL to get the original image available as IIIF compliant.

Some images are appearing small, so you will need to add the filename to the URL.
<https://archive.org/details/img- ...> followed by your image specific information You would normally create the following URL:

[https://iiif.archivelab.org/iiif/img-\[your image specific information\]/info.json](https://iiif.archivelab.org/iiif/img-[your image specific information]/info.json) but to make the image larger, you need to add the filename (\$IMG_ jpg):

[https://iiif.archivelab.org/iiif/img-...\\$IMG_ jpg/info.json](https://iiif.archivelab.org/iiif/img-...$IMG_ jpg/info.json)

To find the image, navigate to the details page: <https://archive.org/details/img-...>

Click on the main image. This will show the image full screen and if you look in the address bar, you will be able to see the filename.

7. Now, use the Internet Archive image service with this identifier:

<https://iiif.archivelab.org/iiif/img-.../info.json>

8. You should then be able to use this image and create an image link:

<https://iiif.archivelab.org/iiif/img-.../full/512,/0/default.jpg>

9. Check that it works with a zooming viewer. For an image uploaded via the Internet Archive, we use the OpenSeaDragon Viewer:

<https://iiif.gdmrdigital.com/openseadragon/index.html? plus your image-specific string, followed by /info.json, or http://openseadragon.github.io> and a browser extension.

Create one IIIF Manifest with the Bodleian Manifest Editor

The prime element in IIIF is called a ‘manifest.’ The IIIF Manifest itself is accessible via a URL that points to a document online (JSON, or JavaScript Object Notation). The IIIF Manifest is shown in viewers such as Mirador or OpenSeaDragon, or the Universal Viewer and more tools.

The Bodleian has a manifest generator tool where you can plugin that info.json file from the image API, and the Bodleian Manifest Editor will generate the manifest for you. A further step is to host those on GitHub, for those who wish to do so.

IIIF viewers need the Image and Presentation APIs be implemented. These client viewers can then be embedded on other sites. Mirador additionally allows for viewing and creating annotations on top of images in a manifest. Here are steps to create one IIIF manifest with the Bodleian Manifest Editor:

1. Find images.
2. Have a list of IIIF info.json URLs for your content.
3. Use the Bodleian Manifest Editor to create a manifest
4. Add at least two Canvases to your Manifest. The option to add a Canvas is below the main image preview, to the left.
5. Click the Canvas Metadata expander in the right-hand column.
6. Click Add an Image. Choose the option to supply the info.json URL and enter the URL for your first image.
7. Edit the Canvas Label and provide an appropriate label for your image

8. Click the Manifest Metadata header in the right-hand column. Edit the Label. Supply a label for your manifest.
9. Click Save Manifest at the top of the right-hand column.
10. Click Download Manifest in the dialog box.

You can now open the Manifest in the Universal Viewer.

More Information about IIIF

Visit the website: <https://iiif.io>

See an updated list of implementations:

<https://github.com/IIIF/awesome-iiif/blob/master/implementations.md>