# Edge Cases: The Making of Network Navigator and Critical Approaches to DH Tools

### John R. Ladd Washington and Jefferson College

### Zoe LeBlanc

University of Illinois Urbana-Champaign

[ © John R. Ladd and Zoe LeBlanc. Published by De Gruyter in cooperation with the University of Luxembourg Centre for Contemporary and Digital History. This is an Open Access article distributed under the terms of the [Creative Commons Attribution License CC-BY-NC-ND](https://creativecommons.org/licenses/by-nc-nd/4.0/)

from IPython.display import Image   
  
display(Image("media/cover\_image.png"))



Cover image for article generated using Canva's AI Tool <https://www.canva.com/ai-image-generator/> with the terms network analysis, digital humanities, and tools.

NetworkAnalysis, Networks, Visualization, Tools, DH

This article will explore the role of network analysis research tools as/in scholarly infrastructure, and also detail our attempt to intervene in these practices by building *Network Navigator*, a browser tool for network analysis. In making *Network Navigator*, we intentionally contributed to an existing ecosystem of network analysis tools, namely *Gephi* and *Palladio*, but we were also motivated to combine the principles of minimal computing with new developments in web architecture and DH. We believe our perspective helps complicate our understanding of “tool”-building, making visible the obscured interpretive work of DH infrastructure. Given the centrality of GUI tools in introductory DH pedagogy, we believe that the making and maintaining of DH tools remains one of the profoundly influential, and yet consistently under-theorized, areas of DH scholarship and teaching. Even as tools seem to be so foundational to DH, at the same time they remain rarely debated or even defined as a category. Rather than treating ‘tools’ as simply technical objects, we want to critically examine how their design and materiality directly shapes this conceptual amorphousness, but also more profoundly how this definitional problem makes it difficult to discern the ways DH tools are determining larger DH systems and infrastructure.

## Nodes: Introducing the Network Navigator Tool and Intervening in Network Analysis Practices in DH

Since ‘Digital Humanities’ was first coined, debating the question of “what is Digital Humanities (DH)” has become so established that even asking it functions as a meta-discourse. And yet, even though the answers vary widely, many of them often include some reference to “DH tools.”

START HERMENEUTICS

To start to explore how tools have become so central in DH, this article utilizes a number of datasets, the first of which is Jason Heppler's dataset from whatisdigitalhumanities.com, accessed via <https://github.com/hepplerj/whatisdigitalhumanities>. This dataset contains quotations from the Day of DH between 2009-2014 and the database contains 817 rows. We used this dataset to explore how frequently the term tool is used in these definitions of DH.

END HERMENEUTICS

START HERMENEUTICS

To analyze this data, we first import the relevant libraries and check if we need to download additional packages for text analysis, as well as setting the path to our datasets and charts.

END HERMENEUTICS

#START HERMENEUTICS  
  
import pandas as pd  
import os  
import json  
import statsmodels.api as sm  
import warnings  
warnings.filterwarnings('ignore')  
  
import altair as alt  
alt.data\_transformers.disable\_max\_rows()  
alt.data\_transformers.enable('json')  
# Check if VS Code is running the script  
if os.environ.get('VSCODE\_PID') is not None:  
 alt.renderers.enable('mimetype')  
else:  
 alt.renderers.enable('default')  
  
  
import nltk  
from nltk.tokenize import word\_tokenize  
## If you haven't downloaded the NLTK data sets yet, do so:  
def download\_nltk\_data\_if\_needed(packages):  
 for package in packages:  
 try:  
 nltk.data.find(package)  
 except LookupError:  
 nltk.download(package)  
  
download\_nltk\_data\_if\_needed(['tokenizers/punkt', 'corpora/stopwords'])  
  
article\_data\_directory\_path = "./script/data/csvs/" # Where all the article datasets are stored  
article\_chart\_directory\_path = "./script/data/charts/" # Where all the article charts are stored  
  
#END HERMENEUTICS

Then we read in the dataset and count the frequency of the word tool in the definitions. As Ted Underwood has argued, word counts, though seemingly very straightforward and even basic, can be a powerful tool for exploring change in language, and deciding how to count requires many interpretative choices that can shape our results.(*Underwood 2013*) For a more detailed explanation of the methods we used, see our CountingTools.ipynb notebook, but primarily we are using a combination of *lowercasing* the textual data for consistency and then *tokenizing* this data, and finally searching for matches in these tokens. This choice is deliberate since our primary focus lies in identifying the frequency of tool names and related terms, such as "network analysis." Our goal is to capture as many instances as possible without being overly permissive and including unrelated words that merely contain these terms.

Furthermore, our choice to utilize word counting and not more complex methods is also intentional, since we are primarily interested in tracing the popularity of tools and network analysis in DH, and less interested in exploring how the language around these terms have changed (something we hope to explore in future work).

#START HERMENEUTICS  
  
# Define the path to the output CSV file  
output\_csv\_path = f"{article\_data\_directory\_path}heppler\_data/whatisdh\_df.csv"  
  
# Check if the output CSV file exists  
if os.path.exists(output\_csv\_path):  
 # If it exists, load the data from the output CSV file  
 whatisdh\_quotes\_df = pd.read\_csv(output\_csv\_path)  
else:  
 # If it doesn't exist, load the data from the original CSV file  
 whatisdh\_quotes\_df = pd.read\_csv(f"{article\_data\_directory\_path}heppler\_data/dayofquotes\_full.csv")  
  
 # Add a 'quote\_id' column based on the DataFrame index  
 whatisdh\_quotes\_df['quote\_id'] = whatisdh\_quotes\_df.index  
  
 # Add a 'lower\_quote' column with the lowercase version of the 'quote' column  
 whatisdh\_quotes\_df['lower\_quote'] = whatisdh\_quotes\_df['quote'].str.lower()  
  
 # Add a 'lower\_tokenized\_quote' column with the tokenized version of the 'lower\_quote' column  
 whatisdh\_quotes\_df['lower\_tokenized\_quote'] = whatisdh\_quotes\_df['lower\_quote'].apply(word\_tokenize)  
  
 # Convert the 'date' column to datetime format  
 whatisdh\_quotes\_df['date'] = pd.to\_datetime(whatisdh\_quotes\_df['date'])  
  
 # Add a 'year' column based on the 'date' column  
 whatisdh\_quotes\_df['year'] = whatisdh\_quotes\_df['date'].dt.year  
  
 # Add a 'quote\_length' column with the length of each tokenized quote  
 whatisdh\_quotes\_df['quote\_length'] = whatisdh\_quotes\_df['lower\_tokenized\_quote'].apply(len)  
  
 # Add a 'tool\_frequency' column with the count of 'tool' in each tokenized quote  
 whatisdh\_quotes\_df['tool\_frequency'] = whatisdh\_quotes\_df['lower\_tokenized\_quote'].apply(lambda x: sum([1 for token in x if 'tool' in token]))  
  
 # Add a 'scaled\_tool\_frequency' column with the 'tool\_frequency' scaled by the 'quote\_length'  
 whatisdh\_quotes\_df['scaled\_tool\_frequency'] = whatisdh\_quotes\_df['tool\_frequency'] / whatisdh\_quotes\_df['quote\_length']  
  
 # Group the DataFrame by 'year' and calculate the sum of 'quote\_length', 'tool\_frequency', and 'scaled\_tool\_frequency' for each year  
 yearly\_tool\_counts\_df = whatisdh\_quotes\_df.groupby(['year']).agg({'quote\_length': 'sum', 'tool\_frequency': 'sum', 'scaled\_tool\_frequency': 'sum'}).reset\_index()  
  
 # Group the DataFrame by 'year' and calculate the total count for each year  
 yearly\_counts\_df = whatisdh\_quotes\_df.groupby('year').size().reset\_index(name='total\_counts')  
  
 # Group the DataFrame by 'year' and calculate the count of quotes with 'tool' for each year  
 yearly\_tool\_whatisdh\_quotes\_df = whatisdh\_quotes\_df[whatisdh\_quotes\_df.tool\_frequency > 0].groupby(['year']).size().reset\_index(name='tool\_counts')  
  
 # Merge the total counts and tool counts DataFrames on the 'year' column  
 merged\_counts\_df = pd.merge(yearly\_counts\_df, yearly\_tool\_whatisdh\_quotes\_df, on='year', how='outer')  
  
 # Merge the merged counts DataFrame and the tool counts DataFrame on the 'year' column  
 final\_counts\_whatisdh\_df = pd.merge(merged\_counts\_df, yearly\_tool\_counts\_df, on='year', how='outer')  
  
 # Add a 'ratio' column with the ratio of 'tool\_counts' to 'total\_counts'  
 final\_counts\_whatisdh\_df['ratio'] = final\_counts\_whatisdh\_df['tool\_counts'] / final\_counts\_whatisdh\_df['total\_counts']  
  
 # Reshape the DataFrame from wide format to long format, with 'year' and 'ratio' as the identifier variables, and 'total\_counts' and 'tool\_counts' as the variable names  
 reshaped\_whatisdh\_df = pd.melt(final\_counts\_whatisdh\_df, id\_vars=['year', 'ratio'], value\_vars=['total\_counts', 'tool\_counts'])  
  
 # Replace 'total\_counts' with 'Total Number of Quotes By Year' in the 'variable' column  
 reshaped\_whatisdh\_df.loc[reshaped\_whatisdh\_df['variable'] == 'total\_counts', 'variable'] = 'Total Number of Quotes By Year'  
  
 # Replace 'tool\_counts' with 'Total Number of Quotes Containing "Tool" By Year' in the 'variable' column  
 reshaped\_whatisdh\_df.loc[reshaped\_whatisdh\_df['variable'] == 'tool\_counts', 'variable'] = 'Total Number of Quotes Containing "Tool" By Year'  
  
 # Write the DataFrame to the output CSV file  
 whatisdh\_quotes\_df.to\_csv(output\_csv\_path, index=False)  
  
#END HERMENEUTICS

START HERMENEUTICS

Finally, we visualize the results both over time and by correlation to the total number of words in each definition.

END HERMENEUTICS

#START HERMENEUTICS  
  
whatisdh\_chart\_output\_path = f"{article\_chart\_directory\_path}whatisdh\_chart.json"  
  
# Check if the chart file exists  
if os.path.exists(whatisdh\_chart\_output\_path):  
 # Load the chart from the JSON file  
 with open(whatisdh\_chart\_output\_path) as json\_file:  
 whatisdh\_chart\_dict = json.load(json\_file)  
 whatisdh\_chart = alt.Chart.from\_dict(whatisdh\_chart\_dict)  
else:  
 # Create a bar chart  
 time\_chart = alt.Chart(reshaped\_whatisdh\_df).mark\_bar().encode(  
 x=alt.X('year:O', title='Year', axis=alt.Axis(labelAngle=0)),  
 y=alt.Y('value:Q', title='Total number of quotes defining DH'),  
 color=alt.Color('variable:N', legend=alt.Legend(title='', orient='bottom', symbolLimit=0, labelLimit=400)),  
 tooltip=[  
 alt.Tooltip('year', title='Year'),  
 alt.Tooltip('value', title='Total Number of Quotes'),  
 alt.Tooltip('variable', title='Quote Category'),  
 alt.Tooltip('ratio', title='Ratio of Quotes', format='.2f')  
 ]  
 ).properties(  
 title={'text': 'Change in Number of Quotes Containing "Tool" Over Time', 'fontSize': 12},  
 width=400,  
 height=300  
 )  
  
 # Define the dependent and independent variables  
 X = final\_counts\_whatisdh\_df['tool\_frequency']  
 y = final\_counts\_whatisdh\_df['quote\_length']  
  
 # Add a constant to the independent variable  
 X = sm.add\_constant(X)  
  
 # Perform the OLS regression  
 model = sm.OLS(y, X)  
 results = model.fit()  
  
 # Get the R-squared value  
 r\_squared = results.rsquared  
  
 # Create the chart  
 chart = alt.Chart(final\_counts\_whatisdh\_df).mark\_circle().encode(  
 x=alt.X('tool\_frequency', axis=alt.Axis(title='Frequency of "tool" in any quote defining DH')),  
 y=alt.Y('quote\_length', axis=alt.Axis(title='Length of any quote defining DH')),  
 color=alt.Color('year:N', legend=alt.Legend(title='Year of Quote'), scale=alt.Scale(scheme='plasma')),  
 size='ratio',  
 tooltip=[  
 alt.Tooltip('year', title='Year'),  
 alt.Tooltip('tool\_frequency', title='Frequency of Tool in Quote'),  
 alt.Tooltip('quote\_length', title='Quote Length'),  
 alt.Tooltip('scaled\_tool\_frequency', title='Scaled Frequency of Tool in Quote', format='.2f'),  
 ]  
 ).properties(  
 title={'text': 'Correlation Between Frequency of "Tool" in Quotes', 'fontSize': 12},  
 width=300,  
 )  
  
 # Add the regression line  
 combined\_chart = chart + chart.transform\_regression('tool\_frequency', 'quote\_length', method="linear").mark\_line(color='red').encode(color=alt.Color(legend=None))  
  
 # Add the R-squared value as a text mark  
 text = alt.Chart({'values':[{'x': 0, 'y': 0}]}).mark\_text(  
 text=f'R-squared: {r\_squared:.2f}',  
 align='left',  
 baseline='top',  
 color='black',  
 dx=5 # Nudges text to right so it doesn't appear on top of the bar  
 ).encode(  
 x=alt.value(0), # pixels from left  
 y=alt.value(0), # pixels from top  
 )  
  
 # Combine the charts  
 final\_chart = combined\_chart + text  
  
 whatisdh\_chart = (alt.hconcat(time\_chart, final\_chart).resolve\_scale(color='independent', y="independent").properties(title="Frequency of 'Tool' in Quotes From whatisdigitalhumanities.com and Day of DH").configure(  
 title=alt.TitleConfig(fontSize=20, font='Garamond', anchor='middle'),  
 axis=alt.AxisConfig(labelFont='Garamond', titleFont='Garamond'),  
 legend=alt.LegendConfig(labelFont='Garamond', titleFont='Garamond')  
 ))  
 whatisdh\_chart.save(whatisdh\_chart\_output\_path)  
  
#END HERMENEUTICS

#START HERMENEUTICS  
  
whatisdh\_chart  
  
#END HERMENEUTICS

alt.HConcatChart(...)

START HERMENEUTICS

The first graph shows that the term tool is used more frequently in the definitions of DH over time, and the second graph shows that this frequency has a positive correlation as quotes get longer over time. This suggests that the term tool is becoming more central to the definition of DH, even as definitions of DH become more verbose.

In addition to frequency, we can also look at the words themselves though generating a word cloud. Though word clouds are not statistically rigorous, seeing the term "tool" so prominently displayed in the word cloud suggests that the term is indeed central to the definitions of DH.

END HERMENEUTICS

wordcloud\_path = f"{article\_chart\_directory\_path}whatisdh\_wordcloud.png"  
rerun\_chart\_code = False

#START HERMENEUTICS  
  
if os.path.exists(wordcloud\_path) == False or rerun\_chart\_code:  
 # Import the wordcloud library and matplotlib  
 from wordcloud import WordCloud  
 import matplotlib.pyplot as plt  
  
 # Define a function to color the word cloud  
 def color\_func(word, \*\*kwargs):  
 if 'tool' in word:  
 return 'red'  
 else:  
 return 'black'  
  
 # Join the 'lower\_quote' column into a single string  
 text = ' '.join(whatisdh\_quotes\_df.lower\_quote.astype(str))  
 # Create the word cloud  
 wordcloud = WordCloud(width=800, height=400, background\_color ='white', color\_func=color\_func).generate(text)  
  
 # Display the word cloud using matplotlib  
 plt.figure(figsize=(15, 8))  
 plt.imshow(wordcloud, interpolation='bilinear')  
 plt.axis('off') # Turn off the axis  
 plt.show()  
 wordcloud.to\_file(wordcloud\_path)  
else:  
  
 display(Image(wordcloud\_path))  
  
#END HERMENEUTICS



START HERMENEUTICS

In this graph, we can see that tool and digital tools are both prominently sized in the word cloud, suggesting that the term tool is used relatively frequently in the definitions of DH. While such word counts and word clouds are not definitive, these methods help suggest that this trend exists and is worth further investigation.

END HERMENEUTICS

While the exact tools referenced have changed over time, increasingly many of those most consistently used and cited have been tools intended for network analysis.

START HERMENEUTICS

In addition to looking at the popularity of tool, we can also explore the popularity of tools specifically for network analysis in DH. To do this, we identified a list of tools for network analysis in DH, based partially on our own experiences and through the list available on the Text Analysis Portal for Research (TAPoR) at <http://tapor.ca/tools/category/Network%20Analysis>. We did not include tools that are primarily for data visualization and might have the ability to visualize networks, but are not specifically for network analysis (such as Tableau, RAWGraphs, Voyant Tools, D3.js, etc.) in our list of network analysis-specific tools.

END HERMENEUTICS

#START HERMENEUTICS  
  
# Create a list of all primariy network analysis tools  
network\_tools = ['Gephi', 'Palladio', 'nodegoat', 'igraph', 'Textexture', 'Netlytic', 'sigma.js', 'Neo4j', 'NetworkX', 'NodeXL', 'Graphviz', 'Cytoscape']  
  
#END HERMENEUTICS

START HERMENEUTICS

Then we import any additional required libraries and create a set of shared functions to identify these terms across our datasets.

END HERMENEUTICS

#START HERMENEUTICS  
  
import numpy as np  
from typing import List, Optional, Tuple  
from sklearn.feature\_extraction.text import CountVectorizer  
  
def count\_term\_frequencies(df: pd.DataFrame, text\_column: str, term\_list: List[str]) -> pd.DataFrame:  
 """  
 Vectorize the text and count the frequencies of specific terms.  
  
 Parameters:  
 df (pd.DataFrame): The input DataFrame.  
 text\_column (str): The name of the column in df that contains the text to process.  
 term\_list (List[str]): A list of terms for which to calculate frequencies.  
  
 Returns:  
 term\_freq\_df (pd.DataFrame): A DataFrame where each column corresponds to a term in term\_list and the values are the frequencies of the term in each document.  
 """  
 # Initialize CountVectorizer with the provided terms as the vocabulary  
 count\_vec = CountVectorizer(ngram\_range=(1,2), vocabulary=term\_list)  
  
 # Fit and transform the text data  
 X = count\_vec.fit\_transform(df[text\_column]).toarray()  
  
 # Create a DataFrame with the term frequencies  
 term\_freq\_df = pd.DataFrame(X, columns=count\_vec.get\_feature\_names\_out())  
  
 # For terms that contain a period, use straight string matching  
 for term in term\_list:  
 if '.' in term:  
 term\_freq\_df[term] = df[text\_column].str.count(term)  
  
 return term\_freq\_df  
  
  
def generate\_word\_counts(subset\_df: pd.DataFrame, text\_column: str, terms\_list: List[str]) -> Tuple[pd.DataFrame, pd.DataFrame, pd.DataFrame]:  
 """  
 Process a DataFrame to calculate term frequencies and tokenize text.  
  
 Parameters:  
 subset\_df (pd.DataFrame): The input DataFrame.  
 text\_column (str): The name of the column in subset\_df that contains the text to process.  
 terms\_list (List[str]): A list of terms for which to calculate frequencies.  
  
 Returns:  
 combined\_df (pd.DataFrame): The original DataFrame, with additional columns for the lowercased text, tokenized text, token frequencies, and total tokens.  
 """  
 if 'lower\_text' not in subset\_df.columns:  
 subset\_df['lower\_text'] = subset\_df[text\_column].str.lower()  
  
 if 'total\_tokens' not in subset\_df.columns:  
 subset\_df['total\_tokens'] = subset\_df['lower\_text'].str.split(' ').str.len()  
   
 count\_df = count\_term\_frequencies(subset\_df, 'lower\_text', terms\_list)  
 combined\_df = pd.concat([subset\_df, count\_df], axis=1)  
   
 return combined\_df  
  
def process\_data(file\_path: str, text\_column: str, date\_column: str, terms\_list: List, data\_origin: str, title: str, term\_type: str, term\_mapping: bool, joined\_term: Optional[str] = None):  
 """  
 Process the data and calculate term frequencies.  
  
 Args:  
 file\_path (str): The path to the CSV file.  
 text\_column (str): The name of the column containing the text data.  
 date\_column (str): The name of the column containing the date data.  
 terms\_list (List): A list of terms to calculate frequencies for.  
 data\_origin (str): The origin of the data.  
 title (str): The title of the data.  
 term\_type (str): The type of term data.  
 term\_mapping (bool): Whether to map term names for formatting variables.  
 joined\_term (str, optional): A term that we've searched for in multiple formats that we want to normalize into one term. Defaults to None.  
  
 Returns:  
 pd.DataFrame: The processed and calculated DataFrame.  
 """  
 # Load the CSV file into a DataFrame  
 df = pd.read\_csv(file\_path)  
  
 # Create a subset of the DataFrame that only includes rows where 'full\_text' is not null  
 subset\_df = df[df[text\_column].notna()]  
 subset\_df = subset\_df.reset\_index(drop=True)  
 # Create a 'cleaned\_conference\_year' column by converting the 'conference\_year' column to string and appending "-01-01"  
 subset\_df[f"cleaned\_{date\_column}"] = subset\_df[date\_column].astype(str) + "-01-01"  
  
 # Convert the 'cleaned\_conference\_year' column to datetime format  
 subset\_df[f"cleaned\_{date\_column}"] = pd.to\_datetime(subset\_df[f"cleaned\_{date\_column}"])  
  
 original\_terms\_list = terms\_list.copy()  
 lower\_terms\_list = [term.lower() for term in terms\_list]  
  
 # Process the DataFrame to calculate term frequencies and tokenize text  
 combined\_df = generate\_word\_counts(subset\_df, text\_column, lower\_terms\_list)  
 # If a tool column is provided, create the 'finalized\_tool' column  
 if joined\_term:  
 plural\_joined\_term = joined\_term + "s"  
 combined\_df[f'finalized\_{joined\_term}'] = np.where(combined\_df[joined\_term].notnull(), combined\_df[joined\_term], combined\_df[plural\_joined\_term])  
 combined\_df = combined\_df.drop(columns=[joined\_term, plural\_joined\_term])  
 combined\_df = combined\_df.rename(columns={f"finalized\_{joined\_term}": joined\_term})  
  
 id\_vars\_columns = subset\_df.columns.tolist()  
 melted\_combined\_df = pd.melt(combined\_df, id\_vars=id\_vars\_columns, var\_name=term\_type, value\_name='counts')  
 melted\_combined\_df['scaled\_counts'] = melted\_combined\_df['counts'] / melted\_combined\_df['total\_tokens']  
   
 # If term mapping is enabled, map the terms  
 if term\_mapping:  
 # Create a mapping from lowercase tool name to the correct name  
 term\_mapping = {term.lower(): term for term in original\_terms\_list}  
 melted\_combined\_df[term\_type] = melted\_combined\_df[term\_type].replace(term\_mapping)  
  
 # Group the DataFrame by 'cleaned\_conference\_year' and 'tool' and calculate the sum of 'counts' for each group, then reset the index  
 summed\_df = melted\_combined\_df.groupby([f"cleaned\_{date\_column}", term\_type]).counts.sum().reset\_index()  
  
 # Group the DataFrame by 'cleaned\_conference\_year' and 'tool' and calculate the sum of 'scaled\_counts' for each group, then reset the index  
 scaled\_df = melted\_combined\_df.groupby([f"cleaned\_{date\_column}", term\_type]).scaled\_counts.sum().reset\_index()  
  
 # Group the DataFrame by 'cleaned\_conference\_year' and calculate the sum of 'total\_tokens' for each group, then reset the index  
 total\_tokens\_df = melted\_combined\_df.groupby([f"cleaned\_{date\_column}", term\_type]).total\_tokens.sum().reset\_index()  
  
 # Merge the grouped DataFrames  
 grouped\_df = pd.merge(summed\_df, scaled\_df, on=[f"cleaned\_{date\_column}", term\_type])  
  
 # Merge the grouped DataFrame with the total tokens DataFrame  
 grouped\_df = pd.merge(grouped\_df, total\_tokens\_df, on=[f"cleaned\_{date\_column}", term\_type])  
  
 # Multiply the 'scaled\_counts' column by 100  
 grouped\_df.scaled\_counts = grouped\_df.scaled\_counts \* 100  
  
 # Rename the 'cleaned\_conference\_year' column to 'date'  
 grouped\_df = grouped\_df.rename(columns={f"cleaned\_{date\_column}": "date"})  
  
 # Add a 'data\_origin' column   
 grouped\_df["data\_origin"] = data\_origin  
  
 # Add a 'title' column  
 grouped\_df["title"] = title  
  
 return grouped\_df  
  
#END HERMENEUTICS

START HERMENEUTICS

Now we can start loading in our datasets and processing them with these functions. For the sake of speed, we have presaved the output of our analysis to be loaded in the charts as json files. However, you can rerun this analysis by setting the rerun variable to True.

END HERMENEUTICS

#START HERMENEUTICS  
  
# Define paths to charts json  
network\_tools\_over\_time\_chart\_output\_path = f"{article\_chart\_directory\_path}network\_tools\_over\_time\_chart.json"  
relative\_popularity\_chart\_output\_path = f"{article\_chart\_directory\_path}relative\_popularity\_chart.json"  
rerun\_chart\_code = False # Set to True to rerun the code to generate the charts  
  
#END HERMENEUTICS

START HERMENEUTICS

Our first dataset was downloaded from Barbot, Laure, Frank Fischer, Yoann Moranville, and Ivan Pozdniakov. “Which DH Tools Are Actually Used in Research?,” December 6, 2019. <https://weltliteratur.net/dh-tools-used-in-research/>, which scraped a list of tools from TaPoR and compared them to the proceedings of DH2015 to DH2019.

END HERMENEUTICS

#START HERMENEUTICS  
  
# Check if the files exist and if the code should be rerun  
if (os.path.exists(network\_tools\_over\_time\_chart\_output\_path) == False) or (os.path.exists(relative\_popularity\_chart\_output\_path) == False) or rerun\_chart\_code:  
 # Load the CSV file into a DataFrame  
 tapor\_adho\_df = pd.read\_csv(f"{article\_data\_directory\_path}barbot\_et\_al\_data/tools-dh-proceedings.csv")  
  
 # Reshape the DataFrame from wide format to long format, with 'Tool' as the identifier variable, 'year' as the variable name, and 'counts' as the value name  
 melted\_tapor\_adho\_df = tapor\_adho\_df.melt(id\_vars=['Tool'], var\_name='year', value\_name='counts')  
  
 # Create a 'date' column by concatenating the 'year' column with '-01-01' to represent the first day of the year  
 melted\_tapor\_adho\_df['date'] = melted\_tapor\_adho\_df['year'].astype(str) + '-01-01'  
  
 # Convert the 'date' column to datetime format  
 melted\_tapor\_adho\_df['date'] = pd.to\_datetime(melted\_tapor\_adho\_df['date'])  
  
 # Group the DataFrame by 'Tool' and calculate the sum of 'counts' for each tool, then reset the index  
 grouped\_dh\_tools = melted\_tapor\_adho\_df.groupby(['year'])['counts'].sum().reset\_index().rename(columns={"counts": "total\_tokens"})  
  
 # Merge the melted DataFrame with the grouped DataFrame on the 'year' column  
 merged\_tapor\_adho\_df = pd.merge(melted\_tapor\_adho\_df, grouped\_dh\_tools, on='year', how='outer')  
  
 # Create a subset of the DataFrame that only includes rows where 'Tool' is in 'network\_tools'  
 subset\_tapor\_adho\_df = merged\_tapor\_adho\_df[merged\_tapor\_adho\_df.Tool.isin(network\_tools)]  
  
 # Drop the 'year' column from the subset DataFrame  
 subset\_tapor\_adho\_df = subset\_tapor\_adho\_df.drop(columns=["year"])  
  
 # Rename the 'Tool' column to 'tool'  
 subset\_tapor\_adho\_df = subset\_tapor\_adho\_df.rename(columns={"Tool": "tools"})  
  
 # Add a 'data\_origin' column with the value 'TaPoR'  
 subset\_tapor\_adho\_df["data\_origin"] = "TaPoR"  
  
 # Add a 'title' column with the value 'TaPoR and DH Conferences Data by Barbot et al. (2019)'  
 subset\_tapor\_adho\_df["title"] = "TaPoR and DH Conferences Data by Barbot et al. (2019)"  
  
 # Set scaled counts to None  
 subset\_tapor\_adho\_df['scaled\_counts'] = subset\_tapor\_adho\_df['counts'] / subset\_tapor\_adho\_df['total\_tokens'] \* 100  
  
#END HERMENEUTICS

START HERMENEUTICS

Next we have Weingart, S.B., Eichmann-Kalwara, N., Lincoln, M., et al. 'DH Conferences Data Extract' in *The Index of Digital Humanities Conferences*. Carnegie Mellon University, 2020. Data last updated 2023-09-20. <https://dh-abstracts.library.cmu.edu>. <https://doi.org/10.34666/k1de-j489>, which we downloaded from <https://dh-abstracts.library.virginia.edu/downloads>. Technically this data somewhat overlaps with the previous dataset, but this one covers a larger time period and includes more conferences, so we have included both to enable the most comprehensive analysis and to allow comparison between the two since they were compiled using differing mechanisms.

END HERMENEUTICS

#START HERMENEUTICS  
  
# Check if the files exist and if the code should be rerun  
if (os.path.exists(network\_tools\_over\_time\_chart\_output\_path) == False) or (os.path.exists(relative\_popularity\_chart\_output\_path) == False) or rerun\_chart\_code:  
 grouped\_index\_conferences\_df = process\_data(  
 file\_path=f"{article\_data\_directory\_path}/weingart\_et\_al\_data/dh\_conferences\_works.csv",  
 text\_column="full\_text",  
 date\_column="conference\_year",  
 terms\_list=network\_tools,  
 data\_origin="Index of DH Conferences",  
 title="Index of DH Conferences by Weingart et al (2023)",  
 term\_type="tools",  
 term\_mapping=True,  
 joined\_term=None  
 )  
  
#END HERMENEUTICS

START HERMENEUTICS

The final dataset is the entire run of *Digital Humanities Quarterly* (DHQ). This dataset comes from the DHQ website (<https://digitalhumanities.org/dhq/>), and the data was scraped and compiled from XML files available here <https://github.com/Digital-Humanities-Quarterly/dhq-journal/tree/main/data/dhq-xml>. All the code and data is available in this GitHub repository <https://github.com/ZoeLeBlanc/dhq_scraper>.

END HERMENEUTICS

#START HERMENEUTICS  
  
# Check if the files exist and if the code should be rerun  
if (os.path.exists(network\_tools\_over\_time\_chart\_output\_path) == False) or (os.path.exists(relative\_popularity\_chart\_output\_path) == False) or rerun\_chart\_code:  
 grouped\_dhq\_df = process\_data(  
 file\_path=f"{article\_data\_directory\_path}scraped\_dhq\_data/dhq\_articles.csv",  
 text\_column="body\_text",  
 date\_column="date\_processed",  
 terms\_list=network\_tools,  
 data\_origin="DHQ",  
 title="DHQ Articles (2023)",  
 term\_type="tools",  
 term\_mapping=True,  
 joined\_term=None  
 )  
   
  
#END HERMENEUTICS

START HERMENEUTICS

Now that we have the datasets processed, we can start looking at trends in network analysis tools over time.

END HERMENEUTICS

#START HERMENEUTICS  
  
# Define colors for each tool to use the same colors across charts  
color\_network\_map = {  
 'Gephi': '#4c78a8',   
 'Palladio': '#aec7e8',   
 'nodegoat': '#ff7f0e',   
 'igraph': '#ffbb78',   
 'Textexture': '#2ca02c',   
 'Netlytic': '#98df8a',   
 'sigma.js': '#d62728',   
 'Neo4j': '#ff9896',   
 'NetworkX': '#9467bd',   
 'NodeXL': '#c5b0d5',   
 'Graphviz': '#8c564b',   
 'Cytoscape': '#c49c94'  
 }  
  
#END HERMENEUTICS

#START HERMENEUTICS  
  
# Check if the chart file exists  
if os.path.exists(network\_tools\_over\_time\_chart\_output\_path) and rerun\_chart\_code == False:  
 # Load the chart from the JSON file  
 with open(network\_tools\_over\_time\_chart\_output\_path) as json\_file:  
 network\_tools\_over\_time\_chart\_dict = json.load(json\_file)  
 network\_tools\_over\_time\_chart = alt.Chart.from\_dict(network\_tools\_over\_time\_chart\_dict)  
else:  
 # Concatenate the three DataFrames into one  
 combined\_df = pd.concat([subset\_tapor\_adho\_df, grouped\_index\_conferences\_df, grouped\_dhq\_df])  
 combined\_df = combined\_df.reset\_index(drop=True)  
  
 combined\_df.date = combined\_df.date.astype(str)  
 combined\_df.date = combined\_df.date.str.split(" ").str[0]  
 combined\_df['date'] = pd.to\_datetime(combined\_df.date, format='%Y-%m-%d')  
 combined\_df = combined\_df.rename(columns={"counts": "Raw Frequency of Mentions", "scaled\_counts": "Frequency of Mentions Scaled by Total Tokens"})  
 # Get a list of unique data origins  
 data\_origins = combined\_df.data\_origin.unique().tolist()  
  
 # Initialize an empty list to store the charts  
 charts = []  
  
 # Get a list of unique tools  
 list\_tools = combined\_df.tools.unique().tolist()  
  
 # Create a list of colors for each tool  
 color\_list = [color\_network\_map[tool] for tool in list\_tools]  
  
 # Create a dropdown selection  
 dropdown = alt.binding\_select(  
 options=['Raw Frequency of Mentions', 'Frequency of Mentions Scaled by Total Tokens'],   
 name='Select Counting Method on Y Axis'  
 )  
 ycol\_param = alt.param(  
 value="Raw Frequency of Mentions",   
 bind=dropdown  
 )  
 # Create a selection that allows the user to select points in the legend  
 selection = alt.selection\_multi(fields=['tools'], bind='legend')  
 # Loop over each data origin  
 # Create a color scale for the chart  
 color\_scale = alt.Scale(  
 domain=list\_tools,  
 range=color\_list  
 )  
 for data\_origin in data\_origins:  
 # Create a subset of the DataFrame that only includes rows where 'data\_origin' is the current data origin  
 subset\_df = combined\_df[combined\_df.data\_origin == data\_origin]  
 # Set the width of the chart depending on the data origin  
 width = 500 if data\_origin != "TaPoR" else 300  
  
 # Get the title of the chart from the 'title' column of the subset DataFrame  
 title = subset\_df.title.unique()[0]  
  
 new\_chart = alt.Chart(subset\_df).mark\_bar().encode(  
 # Encode the 'year' column as the x-axis  
 x=alt.X('year(date):T', axis=alt.Axis(title="Year")),  
 # Encode the 'counts' column as the y-axis  
 y=alt.Y('y:Q', axis=alt.Axis(title="Frequency of Mentions")),  
 # Encode the 'tool' column as the color  
 color=alt.Color('tools:N', legend=alt.Legend(title="Tool"), scale=color\_scale),  
 # Set the opacity of the bars depending on the selection  
 opacity=alt.condition(selection, alt.value(1), alt.value(0.2)),  
 # Set the tooltip to show the 'tool', 'counts', 'date', and 'Frequency of Mentions Scaled by Total Tokens' columns  
 tooltip=[  
 alt.Tooltip('tools', title='Tool'),  
 alt.Tooltip('Raw Frequency of Mentions'),  
 alt.Tooltip('date:T', title='Date', format='%Y'),  
 alt.Tooltip('Frequency of Mentions Scaled by Total Tokens', title="Frequency of Mentions Scaled by Total Tokens (Percentage)", format='.2f'),  
 alt.Tooltip('total\_tokens', title='Total Tokens'),  
 ]  
 ).transform\_calculate(  
 y=f'datum[{ycol\_param.name}]'  
 ).add\_params(  
 ycol\_param, selection  
 ).properties(  
 # Set the title and width of the chart  
 title=f"{title}",  
 width=width  
 )  
   
  
 # Add the chart to the list of charts  
 charts.append(new\_chart)  
  
 # Concatenate the charts horizontally  
 network\_tools\_over\_time\_chart = alt.hconcat(\*charts).properties(  
 # Set the title of the concatenated chart  
 title={  
 "text": "Popularity of Network Analysis Tools Over Time and Across Multiple Datasets",  
 "subtitle": ["The Y-Axis can be changed to show either the raw frequency of a tool name in each respective dataset or the frequency of mentions scaled by all tokens for that year.", "You can hover over individual tools for more information and the legend is also interactive, allowing you to select specific tools to highlight."],  
 }  
 ).configure(  
 # Configure the title, axis, and legend of the chart  
 title=alt.TitleConfig(fontSize=20, font='Garamond', anchor='middle'),  
 axis=alt.AxisConfig(labelFont='Garamond', titleFont='Garamond'),  
 legend=alt.LegendConfig(labelFont='Garamond', titleFont='Garamond')  
 )  
 network\_tools\_over\_time\_chart.save(network\_tools\_over\_time\_chart\_output\_path)  
  
#END HERMENEUTICS

#START HERMENEUTICS  
  
# Display the concatenated chart  
network\_tools\_over\_time\_chart  
  
#END HERMENEUTICS

alt.HConcatChart(...)

START HERMENEUTICS

We can also consider the relative popularity of network analysis tools compared to other most popular tools in DH, using the list from TAPoR.

END HERMENEUTICS

#START HERMENEUTICS  
  
# Check if the chart file exists  
if os.path.exists(relative\_popularity\_chart\_output\_path) and rerun\_chart\_code == False:  
 # Load the chart from the JSON file  
 with open(relative\_popularity\_chart\_output\_path) as json\_file:  
 relative\_popularity\_chart\_dict = json.load(json\_file)  
 relative\_popularity\_chart = alt.Chart.from\_dict(relative\_popularity\_chart\_dict)  
else:  
 # Get a list of unique tools  
 grouped\_merged\_dh\_tools = merged\_tapor\_adho\_df.groupby(['Tool'])['counts'].sum().reset\_index()  
  
 total\_tokens = grouped\_merged\_dh\_tools['counts'].sum()  
 grouped\_merged\_dh\_tools['scaled\_counts'] = grouped\_merged\_dh\_tools['counts'] / total\_tokens \* 100  
 all\_dh\_tools = grouped\_merged\_dh\_tools.Tool.unique().tolist()  
  
 # Get the count of the second most frequent network tool  
 second\_frequent\_network\_tool = grouped\_merged\_dh\_tools[grouped\_merged\_dh\_tools.Tool.isin(network\_tools)].sort\_values(by='counts', ascending=False).head(2)[1:2].counts.values[0]  
  
 # Get a list of tools that have a count greater than or equal to the count of the second most frequent network tool  
 updated\_network\_tools = grouped\_merged\_dh\_tools[grouped\_merged\_dh\_tools.counts >= second\_frequent\_network\_tool].sort\_values(by='counts', ascending=False).Tool.unique().tolist()  
  
 # Create a list of colors for each tool  
 updated\_network\_colors = [color\_network\_map[tool] if tool in color\_network\_map else '#bab0ac' for tool in updated\_network\_tools]  
 grouped\_merged\_dh\_tools = grouped\_merged\_dh\_tools.rename(columns={"counts": "Raw Frequency of Mentions", "scaled\_counts": "Frequency of Mentions Scaled by All Tool Mentions"})  
 # Create a dropdown selection  
 dropdown = alt.binding\_select(  
 options=["Raw Frequency of Mentions", "Frequency of Mentions Scaled by All Tool Mentions"],   
 name='Select Counting Method on Y Axis'  
 )  
 xcol\_param = alt.param(  
 value="Raw Frequency of Mentions",  
 bind=dropdown  
 )  
  
 # Create a bar chart of the total tools  
 total\_tools = alt.Chart(grouped\_merged\_dh\_tools[grouped\_merged\_dh\_tools["Raw Frequency of Mentions"] >= second\_frequent\_network\_tool]).mark\_bar().encode(  
 x=alt.X('x:Q', title='Number of mentions'),  
 y=alt.Y('Tool:N', title='Tool', sort='-x'),  
 color=alt.Color('Tool:N', legend=None, scale=alt.Scale(domain=updated\_network\_tools, range=updated\_network\_colors)),  
 tooltip=[  
 alt.Tooltip('Tool', title='Tool'),  
 alt.Tooltip('Raw Frequency of Mentions', title='Raw Frequency of Mentions'),  
 alt.Tooltip('Frequency of Mentions Scaled by All Tool Mentions', title='Frequency of Mentions Scaled by All Tool Mentions (Percentage)', format='.2f'),  
 ]  
 ).transform\_calculate(  
 x=f'datum[{xcol\_param.name}]'  
 ).add\_params(xcol\_param).properties(  
 title="TaPoR and DH Conferences Data by Barbot et al. (2019)",  
 )  
  
 # Process the DataFrame to calculate term frequencies and tokenize text  
 grouped\_index\_conferences\_df = process\_data(  
 file\_path=f"{article\_data\_directory\_path}weingart\_et\_al\_data/dh\_conferences\_works.csv",  
 text\_column="full\_text",  
 date\_column="conference\_year",  
 terms\_list=all\_dh\_tools,  
 data\_origin="Index of DH Conferences",  
 title="Index of DH Conferences by Weingart et al (2023)",  
 term\_type="all\_tools",  
 term\_mapping=True,  
 joined\_term=None  
 )  
 all\_tools\_index\_conferences\_df = grouped\_index\_conferences\_df.groupby('all\_tools').agg({'counts': 'sum'}).reset\_index()  
  
 total\_index\_tools = all\_tools\_index\_conferences\_df['counts'].sum()  
 all\_tools\_index\_conferences\_df['scaled\_counts'] = all\_tools\_index\_conferences\_df['counts'] / total\_index\_tools \* 100  
  
 # Get the frequency of the second most frequent network tool  
 conference\_second\_frequent\_network\_tool = all\_tools\_index\_conferences\_df[all\_tools\_index\_conferences\_df.all\_tools.isin(network\_tools)].sort\_values(by='counts', ascending=False).head(2)[1:2].counts.values[0]  
  
 # Get a list of tools that have a frequency greater than or equal to the frequency of the second most frequent network tool  
 conference\_updated\_network\_tools = all\_tools\_index\_conferences\_df[all\_tools\_index\_conferences\_df.counts >= conference\_second\_frequent\_network\_tool].sort\_values(by='counts', ascending=False).all\_tools.unique().tolist()  
  
 # Create a list of colors for each tool  
 conference\_updated\_network\_colors = [color\_network\_map[tool] if tool in color\_network\_map else '#bab0ac' for tool in conference\_updated\_network\_tools]  
  
  
 all\_tools\_index\_conferences\_df = all\_tools\_index\_conferences\_df.rename(columns={"all\_tools": "Tool", 'counts': 'Raw Frequency of Mentions', 'scaled\_counts': 'Frequency of Mentions Scaled by All Tool Mentions'})  
  
 # Create a bar chart of the total tools  
 conference\_total\_tools = alt.Chart(all\_tools\_index\_conferences\_df[all\_tools\_index\_conferences\_df["Raw Frequency of Mentions"] >= conference\_second\_frequent\_network\_tool]).mark\_bar().encode(  
 x=alt.X('x:Q', title='Number of mentions'),  
 y=alt.Y('Tool:N', sort='-x'),  
 color=alt.Color('Tool:N', legend=None, scale=alt.Scale(domain=conference\_updated\_network\_tools, range=conference\_updated\_network\_colors)),  
 tooltip=[  
 alt.Tooltip('Tool', title='Tool'),  
 alt.Tooltip('Raw Frequency of Mentions', title='Raw Frequency of Mentions'),  
 alt.Tooltip('Frequency of Mentions Scaled by All Tool Mentions', title='Frequency of Mentions Scaled by All Tool Mentions (Percentage)', format='.2f'),  
 ]  
 ).transform\_calculate(  
 x=f'datum[{xcol\_param.name}]'  
 ).add\_params(xcol\_param).properties(  
 title="Index of DH Conferences by Weingart et al (2023)",  
 )  
  
 # Process the DataFrame to calculate term frequencies and tokenize text  
 grouped\_dhq\_df = process\_data(  
 file\_path=f"{article\_data\_directory\_path}scraped\_dhq\_data/dhq\_articles.csv",  
 text\_column="body\_text",  
 date\_column="date\_processed",  
 terms\_list=all\_dh\_tools,  
 data\_origin="DHQ",  
 title="DHQ Articles (2023)",  
 term\_type="all\_tools",  
 term\_mapping=True,  
 joined\_term=None  
 )  
  
 all\_tools\_dhq\_df = grouped\_dhq\_df.groupby('all\_tools').agg({'counts': 'sum'}).reset\_index()  
 total\_dhq\_tools = all\_tools\_dhq\_df['counts'].sum()  
 all\_tools\_dhq\_df['scaled\_counts'] = all\_tools\_dhq\_df['counts'] / total\_dhq\_tools \* 100  
  
 # Get the frequency of the second most frequent network tool  
 dhq\_second\_frequent\_network\_tool = all\_tools\_dhq\_df[all\_tools\_dhq\_df.all\_tools.isin(network\_tools)].sort\_values(by='counts', ascending=False).head(2)[1:2].counts.values[0]  
  
 # Get a list of tools that have a frequency greater than or equal to the frequency of the second most frequent network tool  
 dhq\_updated\_network\_tools = all\_tools\_dhq\_df[all\_tools\_dhq\_df.counts >= dhq\_second\_frequent\_network\_tool].sort\_values(by='counts', ascending=False).all\_tools.unique().tolist()  
  
 # Create a list of colors for each tool  
 dhq\_updated\_network\_colors = [color\_network\_map[tool] if tool in color\_network\_map else '#bab0ac' for tool in dhq\_updated\_network\_tools]  
  
 all\_tools\_dhq\_df = all\_tools\_dhq\_df.rename(columns={"all\_tools": "Tool", 'counts': 'Raw Frequency of Mentions', 'scaled\_counts': 'Frequency of Mentions Scaled by All Tool Mentions'})  
 # Create a bar chart of the total tools  
 dhq\_total\_tools = alt.Chart(all\_tools\_dhq\_df[all\_tools\_dhq\_df["Raw Frequency of Mentions"] >= dhq\_second\_frequent\_network\_tool]).mark\_bar().encode(  
 x=alt.X('x:Q', title='Number of mentions'),  
 y=alt.Y('Tool:N', sort='-x'),  
 color=alt.Color('Tool:N', legend=None, scale=alt.Scale(domain=dhq\_updated\_network\_tools, range=dhq\_updated\_network\_colors)),  
 tooltip=[  
 alt.Tooltip('Tool', title='Tool'),  
 alt.Tooltip('Raw Frequency of Mentions', title='Raw Frequency of Mentions'),  
 alt.Tooltip('Frequency of Mentions Scaled by All Tool Mentions', title='Frequency of Mentions Scaled by All Tool Mentions (Percentage)', format='.2f'),  
 ]  
 ).transform\_calculate(  
 x=f'datum[{xcol\_param.name}]'  
 ).add\_params(xcol\_param).properties(  
 title="DHQ Articles (2023)",  
 )  
  
 # Concatenate the charts horizontally  
 relative\_popularity\_chart = alt.hconcat(total\_tools, conference\_total\_tools, dhq\_total\_tools).configure(  
 title=alt.TitleConfig(fontSize=20, font='Garamond', anchor='middle'),  
 axis=alt.AxisConfig(labelFont='Garamond', titleFont='Garamond'),  
 legend=alt.LegendConfig(labelFont='Garamond', titleFont='Garamond')  
 ).resolve\_scale(color='independent').properties(  
 title={  
 "text":"Ranking of Most Popular Network Analysis Tools Relative to Other DH Tools Across Multiple Datasets",  
 "subtitle": ["The Y-Axis can be changed to show either the raw frequency of a tool name in each respective dataset", "or the frequency of mentions scaled by total references to any tool for that dataset.", "You can also hover over individual tools for more information."],  
 }  
 )  
 relative\_popularity\_chart.save(relative\_popularity\_chart\_output\_path)  
  
#END HERMENEUTICS

#START HERMENEUTICS  
  
# Display the concatenated chart  
relative\_popularity\_chart  
  
#END HERMENEUTICS

alt.HConcatChart(...)

While these figures hint at the surging popularity of these tools (and network analysis more broadly), what is only starting to be studied is how this rise of network analysis in DH has been delimited by what these tools can produce (*van Geenen, Gray, Bounegru, Venturini, Jacomy, Meunier 2023*). Consider, for instance, the prevalence of force layout diagrams in DH, colloquially known as the ‘hairball graph.’ Although these diagrams are often criticized for being difficult to interpret in DH, we believe their widespread use is also indicative of the power these “tools” exert on what scholars think is possible with network analysis. Given this influence, we attempted to intervene in these practices by building *Network Navigator*, a browser-based tool for network analysis in DH.

START HERMENEUTICS

To give a sense of the background of *Network Navigator*, we can explore the history of the project via the GitHub repository <https://github.com/jrladd/network_navigator>.

END HERMENEUTICS

#START HERMENEUTICS  
  
nn\_chart\_output\_path = f"{article\_chart\_directory\_path}/nn\_chart.json"  
  
# Check if the CSV files exist  
if os.path.exists(nn\_chart\_output\_path) and rerun\_chart\_code == False:  
 with open(nn\_chart\_output\_path) as json\_file:  
 nn\_chart\_dict = json.load(json\_file)  
 nn\_chart = alt.Chart.from\_dict(nn\_chart\_dict)  
else:  
 # Load the CSV files into DataFrames  
 nn\_commits\_df = pd.read\_csv(f"{article\_data\_directory\_path}github\_data/network\_navigator\_commits.csv")  
  
 # Convert the 'commit.author.date' column to datetime format  
 nn\_commits\_df['commit.author.date'] = pd.to\_datetime(nn\_commits\_df['commit.author.date'], errors='coerce')  
  
 # Replace 'Zoe LeBlanc' with 'ZoeLeBlanc' in the 'commit.author.name' column  
 nn\_commits\_df.loc[nn\_commits\_df['commit.author.name'] == 'Zoe LeBlanc', 'commit.author.name'] = 'ZoeLeBlanc'  
  
 # Create a subset of the DataFrame that only includes certain columns  
 subset\_commits\_df = nn\_commits\_df[['commit.author.name', 'commit.author.date', 'stats.total', 'stats.additions', 'stats.deletions']]  
  
 # Replace '.' with '\_' in the column names  
 cols = subset\_commits\_df.columns  
 cols = [col.replace('.', '\_') for col in cols]  
 subset\_commits\_df.columns = cols  
  
 # Prepare data for symmetric bar chart  
 df\_additions = subset\_commits\_df[["commit\_author\_name", "commit\_author\_date", "stats\_additions"]].copy()  
 df\_additions["type"] = "additions"  
 df\_additions.rename(columns={"stats\_additions": "value"}, inplace=True)  
  
 df\_deletions = subset\_commits\_df[["commit\_author\_name", "commit\_author\_date", "stats\_deletions"]].copy()  
 df\_deletions["type"] = "deletions"  
 df\_deletions["stats\_deletions"] = -df\_deletions["stats\_deletions"]  
 df\_deletions.rename(columns={"stats\_deletions": "value"}, inplace=True)  
  
 # Concatenate the additions and deletions DataFrames  
 df\_graph = pd.concat([df\_additions, df\_deletions])  
  
 # Create the symmetric bar chart  
 project\_history = alt.Chart(df\_graph).mark\_bar().encode(  
 x=alt.X('commit\_author\_date:T', title="Date"),  
 y=alt.Y('value:Q', title="Additions/Deletions"),  
 color=alt.Color('type:N', title="Type of Change"),  
 tooltip=['commit\_author\_name:N', 'commit\_author\_date:T', 'value:Q', 'type:N'],  
 ).properties(  
 title="Additions and Deletions to Network Navigator",  
 )  
  
 # Create a selection that allows the user to select points in the legend  
 selection = alt.selection\_point(fields=['commit\_author\_name'], bind='legend')  
  
 # Create a bar chart of the total contributions by each author  
 author\_chart = alt.Chart(subset\_commits\_df).mark\_bar(opacity=0.7).encode(  
 x=alt.X('commit\_author\_date:T', title='Date'),  
 y=alt.Y('stats\_total:Q', title='Total Contributions').stack(None),  
 color=alt.Color('commit\_author\_name:N',  
 scale=alt.Scale(scheme='tableau20'),   
 legend=alt.Legend(title="Author"),  
 sort=alt.EncodingSortField(  
 field="stats\_total", # The field to use for the sort  
 op="sum", # The operation to run on the field prior to sorting  
 order="descending" # The sort order  
 )  
 ),  
 tooltip=['commit\_author\_name:N', 'commit\_author\_date:T', 'stats\_total:Q'],  
 opacity=alt.condition(selection, alt.value(1), alt.value(0.2)),  
 ).add\_params(selection).interactive(bind\_y=False).properties(  
 title="Contributors to Network Navigator",  
 )  
  
 # Concatenate the charts horizontally  
 nn\_chart = alt.hconcat(project\_history, author\_chart).resolve\_scale(y='independent', color='independent').properties(  
 title="History of Network Navigator via GitHub"  
 ).configure(  
 title=alt.TitleConfig(fontSize=20, font='Garamond', anchor='middle'),  
 axis=alt.AxisConfig(labelFont='Garamond', titleFont='Garamond'),  
 legend=alt.LegendConfig(labelFont='Garamond', titleFont='Garamond')  
 )  
 nn\_chart.save(nn\_chart\_output\_path)  
  
#END HERMENEUTICS

#START HERMENEUTICS  
  
# Display the concatenated chart  
  
nn\_chart  
  
#END HERMENEUTICS

alt.HConcatChart(...)

Built initially from 2017-2018 by Ladd, and then redesigned with LeBlanc from 2020-2022, the original impetus for *Network Navigator* was to provide scholars and students a platform that emphasized a deeper understanding and exploration of networks—particularly in terms of metrics and visualizations. Ladd and LeBlanc had observed that for newcomers to networks in DH, most of them prioritized learning tools, like *Gephi* or *Palladio*, which unintentionally neglected the broader question of how to interpret networks, a phenomenon often referred to as ‘button-ology’(*Russell, Hensley 2017*). While *Network Navigator* aimed to overcome these challenges, its development was also impacted by the onset of the COVID-19 pandemic. This not only influenced how Ladd and LeBlanc collaborated but also necessitated DH tools that were not only easy to use, but that did not require extensive technical setup or installation since most users would be working remotely.

In making *Network Navigator*, we intentionally contributed to this existing and evolving ecosystem of tools, but we were also motivated to build in ways that intentionally prioritized the constraints of DH. Such an approach runs counter to most software development practices that often privilege maximizing user base and scale, which often requires complex infrastructure and the latest frameworks web development. Instead, we carefully considered not only what we could build, but also what we should exclude from this tool. The result is a tool that is not as full featured as other network analysis tools, but we believe that our choices, while still prioritizing users, challenge the default notion of user-friendliness in DH tools (a goal that in DH has been rarely defined or until recently, even studied). Thus, *Network Navigator* is intended to help researchers to understand networks as both an information design technique and a statistical methodology. But how to balance these goals is far from straightforward. Even more fraught is when these tools function as pedagogical instruments, introducing researchers to a methodology—whether that is network analysis, topic modeling, or even counting words, to name but a few of the most popular, current methods. In this scenario, the tool itself can have enormous influence on how scholars understand these methods, thus shaping not only how we research, but also make knowledge claims.

We are far from the first scholars to raise such questions or concerns regarding DH tools. Nonetheless, we believe sharing our journey with *Network Navigator*—from conception to (re)construction—can contribute to the ongoing critical reassessment of the very category of DH tools and their often difficult to discern role in shaping DH knowledge production. With the rise of subfields like Cultural Analytics and Computational Humanities, as well as the proliferation of data science methods, our very definitions of tools have expanded far beyond the early days of humanities computing, with code libraries, scripts, and complex modeling architectures increasingly integral to DH research. Yet, even in this data-driven landscape, we contend that there remains a role for what has often been termed “off-the-shelf” tools, that is those with a graphical user interface (GUI) and that require no programming knowledge.

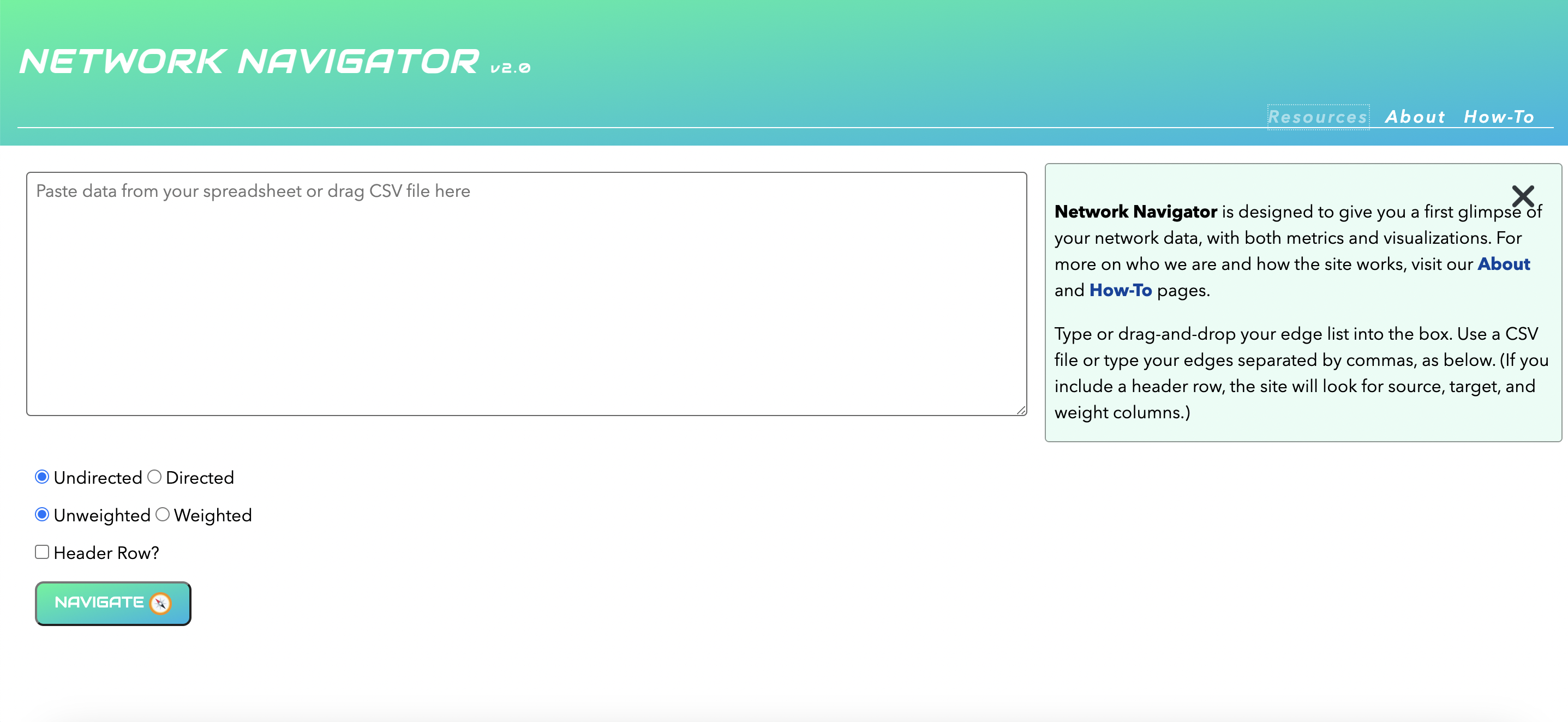
Indeed, given the importance of these GUI tools in introductory DH pedagogy, we believe the creation and maintenance of DH tools is profoundly influential in democratizing access to DH. Yet, how the design and development of these tools can further DH pedagogy remains under-theorized in scholarship, which tends to apply a critical lens to the use of tools, not their construction. Such a gap reflects the historical divide of who builds these tools (primarily developers) versus who uses and studies them (researchers and students), a distinction we hope continues to become blurrier in the future. But it is also symptomatic of the difficulties in studying DH tools. As Ramsay and Rockwell argued, “For tools to be theories in the way digital humanists want—in a way that makes them accessible to, for example, peer review—opacity becomes an almost insuperable problem” (*Ramsay, Rockwell 2012*, 80). Rather than accepting this opacity as inevitable or treating ‘tools’ as simply technical objects, we theorize our experiences in building *Network Navigator* as a window into both our intellectual choices and as an exemplar for future DH tools, especially those aimed at helping researchers learn new methods.

Thus, we hope that by sharing not only our source code, but also our sources of inspiration and interpretation, we can contribute to the growing scholarship that seeks to complicate our understanding of “tool”-building and make visible the obscured interpretive work of DH infrastructure. We think such work not only has implications for the future of network analysis in DH, but also will hopefully chart how the next generation of these tools can not only leverage new technologies, but also engage with DH-centric theoretical and design frameworks. Such goals are crucial for ensuring that DH tools support humanistic research in ways that are sustainable and accessible. Achieving this vision though first requires tracing the origins and making of *Network Navigator*, a trajectory that spans not only code commits but also the history of DH tools writ large.

## Edges: The Origins of Network Navigator and Critical Making of DH Tools

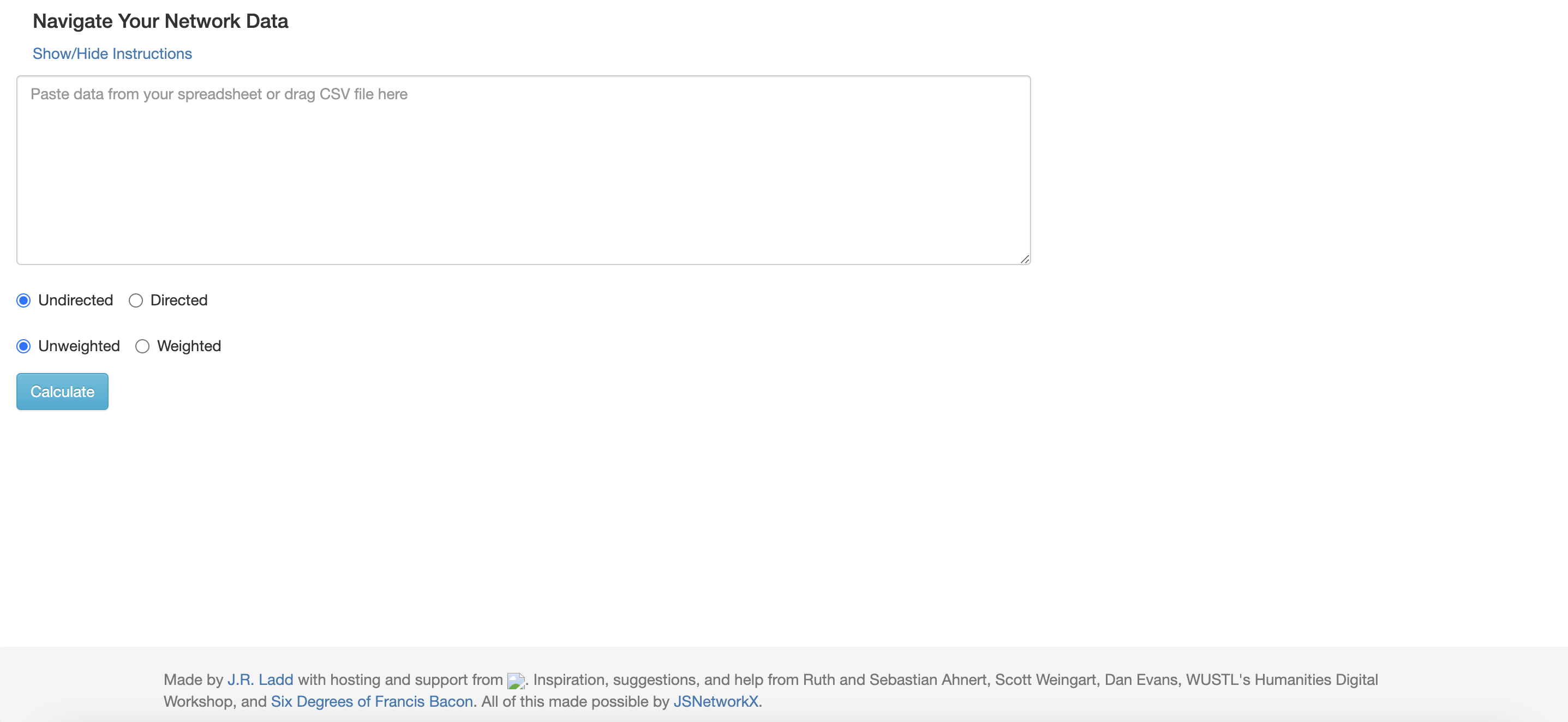
Today, when a user first navigates to *Network Navigator* at <https://networknavigator.jrladd.com/>, they encounter a web page divided into sections: a branded header; instructions on the right for how to use the site; a box to paste their data; buttons to customize their network; and lastly links to resources, about, and how-to page.

display(Image("media/nn\_homepage.png"))



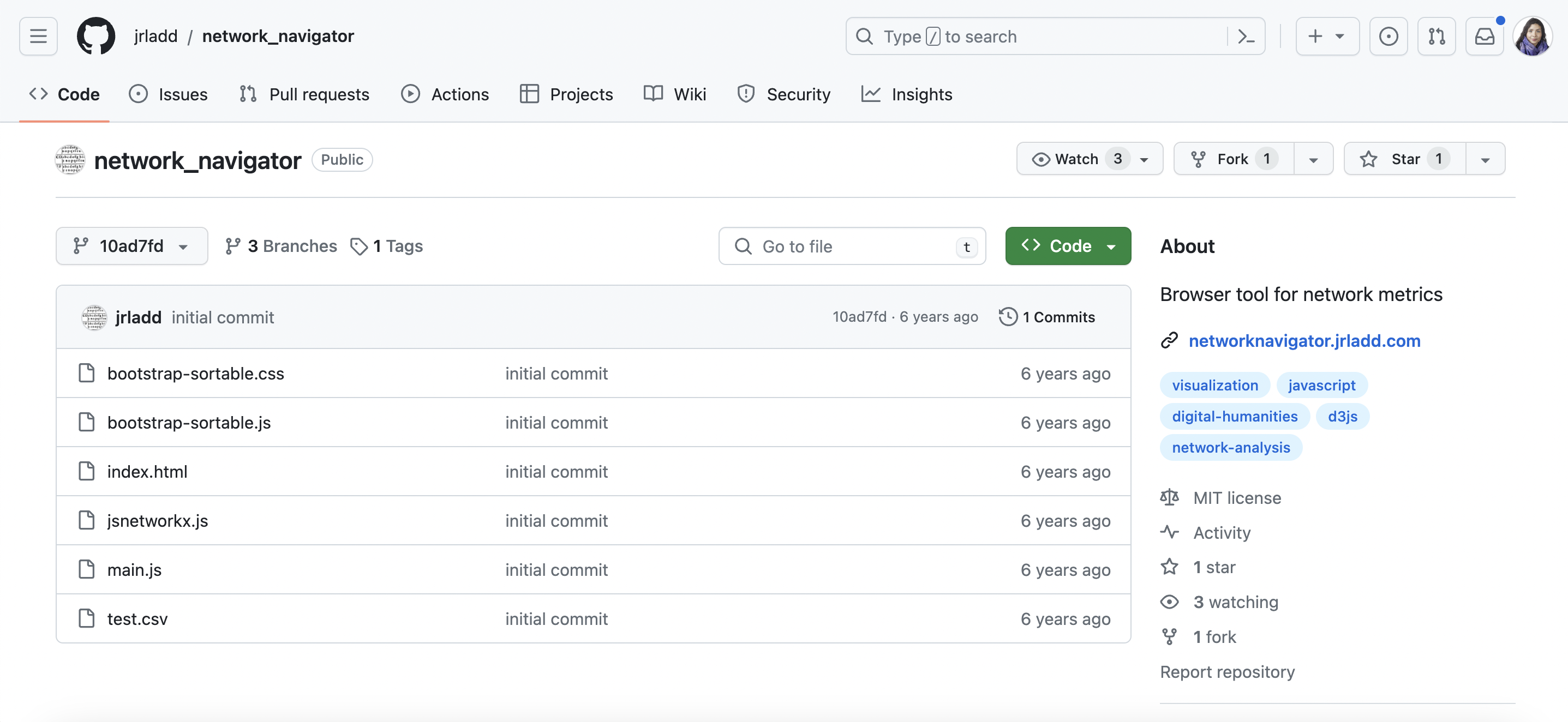
Such a layout is intended to be relatively straightforward for the user, and without many interactive features or pop-ups that have become widespread in web apps. But this design and layout is also a product of the longer history of the project.

display(Image("media/nn\_original\_homepage.png"))



Initially, *Network Navigator* was simply a table. The imagined user would drag-and-drop network data into the browser and receive in return cleanly-formatted tabular data, detailing some standard network metrics.

display(Image("media/nn\_first\_commit\_repo.png"))



Ladd first created the GitHub repository for *Network Navigator* on August 8, 2017, and his first commit consisted of six files: a homepage through an index HTML file, an initial Javascript file to build networks, an example spreadsheet, and then imports from two libraries: JSNetworkX for the network analysis and the web design framework Bootstrap for styling of the website -- libraries we discuss more in-depth later in this article. The idea for the tool was born from Ladd’s contributions to the *Six Degrees of Francis Bacon* (SDFB) project, which visualized the social networks of early modern England in a crowdsourced web application. The SDFB site included many D3 visualizations for networks, drawing on a range of then-new visualization paradigms for networks in the humanities. However, the scope of that project and its design didn’t allow for some of the more advanced metrics Ladd was using internally for the project at the time, and he saw the possibilities for a web tool that could provide this functionality. Drawing from these experiences, Ladd built the first prototype of *Network Navigator* over the remainder of 2017, and while initially hosted via Ladd’s website and GitHub pages, in 2018, with assistance from Dan Evans, the tool was hosted by Carnegie Mellon University Libraries’ servers.

START HERMENEUTICS

Evans was succeeded in his role by Matt Lincoln, who contributed some fixes and oversaw the site while it was hosted at CMU, which is reflected in our earlier project history visualizations.

END HERMENEUTICS

Beyond his experience on SDFB, one of the initial motivating factors for Ladd in creating *Network Navigator* was the desire to foreground network metrics, in response to humanists’ increasing interest in networks. This shift was also due to the emergence of Cultural Analytics and data science, where researchers were exploring the potential for networks not just as visualizations or databases but also applying network statistics and graph theory to study culture. At the time, only *Gephi* had these types of metrics–*Palladio* and other network analysis tools had none–and the metrics in *Gephi* were presented in a way that was often overwhelming to novices. With a few clicks, users in *Gephi* could generate an enormous amount of statistical inferences about their networks, but these buttons included few explanations of how to interpret or understand these metrics. While scholars like Ben Schmidt have argued that humanists do not need to necessarily understand these algorithms, Ladd felt that similar to Rebecca Koeser’s assessment on the dangers of trusting others to do the math, tools like *Gephi* unintentionally made it difficult for humanists to understand the principles of network analysis, at times resulting in an over-interpretation of the visual features of graph layouts rather than engagement with network statistics (*Schmidt 2016*,*Koeser 2015*). Therefore, *Network Navigator* initially combined a handful of the most popular introductory metrics in an interface that would help users learn how these metrics could inform their research questions, with the goal of helping the field move beyond networks as only a critical visualization practice towards what networks could tell us statistically and computationally.

This core ethos has remained central to Network Navigator even as it was rebuilt by Ladd and LeBlanc, but before detailing their process, it is crucial to situate the tool within the longer history of DH tools and what Leigh Star and Karen Ruhleder termed an “ecology of infrastructure” (*Star, Ruhleder 1996*). Infrastructure, though a vast and sometimes nebulous concept—as anthropologist Brian Larkin notes, it can be “conceptually unruly”—is pivotal in understanding that digital tools transcend mere outputs (*Larkin 2013*, 329). Such an approach to studying these tools is in line with the recent rise of Critical Infrastructure Studies in DH, which seeks to bridge fields like Science and Technology Studies (STS) and Library and Information Sciences (LIS) with how we study the underlying systems and structures that support and shape DH research, tools, and methodologies. How Network Navigator constitutes infrastructure then is partially a question of its relation to both users and other existing network tools, which we explore later in this article. But it is also a helpful framework to contextualize the choices that shaped this tool within the longer trends and debates over DH tools, as Yrjö Engeström has argued, when studying a tool, we need to ask “when is a tool”, rather than what is a tool”: in essence, not only exploring its functionality, but also its histories (*Engeström 1990*).

To help understand the growing interest in DH tools and infrastructure, we can also explore these terms in DH academic writing. This approach is far from perfect, but can offer a helpful additional line of evidence to scholarly analysis for how these terms and the ideas they represent have evolved in DH. Using DH journals and other academic writings is an increasingly popular approach to study trends in the field, such as studies of the term theory in DH journals (*Kleymann, Niekler, Burghardt 2022*) or using DH journal articles to situate it across disciplines (*Luhmann, Burghardt 2022*).

To do this, we use the DH Conference Abstracts dataset from the *Index of DH Conferences*, the articles in *Digital Humanities Quarterly*, as well as data from DH journals that are available in the JSTOR Constellate platform, specifically *Computers and the Humanities* and *Languague Resources and Evaluation*. The choice of these journals is partially driven by what's available, but is also informed by growing research into which journals most represent DH (*Spinaci, Colavizza, Peroni 2022*,*Li, Ma, Fang 2023*) .

START HERMENEUTICS

First, we define a list of terms related to tools and infrastructure, and then we use the same functions we used earlier to identify these terms in the datasets.

END HERMENEUTICS

#START HERMENEUTICS  
  
terms = ['tool', 'tools', 'infrastructure']  
tool\_infrastructure\_chart\_output\_path = f"{article\_chart\_directory\_path}tool\_infrastructure\_chart.json"  
rerun\_chart\_code = False # Set to True to rerun the code to generate the charts  
  
#END HERMENEUTICS

START HERMENEUTICS

We also define new functions for processing JSTOR data. To collect this data, we selected the full run of the journals and downloaded the metadata, as well as unigram, bigram, and trigram data. Because of the size of this dataset (approximately ~400 MB) we cannot include it in this repository, but it is available on request. Instead, we have preprocessed this data and save it as a csv file, which we can load in here.

END HERMENEUTICS

#START HERMENEUTICS  
  
def process\_jstor\_data(df: pd.DataFrame, terms: List[str], date\_column: str, id\_column: str, origin\_column: str, word\_count\_column: str, joined\_term: Optional[str] = None) -> pd.DataFrame:  
 """  
 Process and group a DataFrame to calculate term frequencies.  
  
 Parameters:  
 df (pd.DataFrame): The input DataFrame.  
 terms (List[str]): A list of terms for which to calculate frequencies.  
 date\_column (str): The name of the column in df that contains the date.  
 id\_column (str): The name of the column in df that contains the ID.  
 origin\_column (str): The name of the column in df that contains the origin of the data.  
 word\_count\_column (str): The name of the column in df that contains the word count.  
 joined\_term (str, optional): A term that we've searched for in multiple formats that we want to normalize into one term. Defaults to None.  
  
 Returns:  
 final\_grouped\_df (pd.DataFrame): A DataFrame containing the terms and their frequencies, grouped by date and origin.  
 """  
 def process\_counts(row):  
 for term in terms:  
 split\_term = term.split()  
 column\_length = 'unigramCount' if len(split\_term) == 1 else 'bigramCount' if len(split\_term) == 2 else 'trigramCount'  
 counts = row[column\_length][term] if term in row[column\_length] else 0  
 row[term] = counts  
 return row  
  
 df = df.apply(process\_counts, axis=1)  
  
 subset\_df = df[[date\_column, id\_column, origin\_column, word\_count\_column] + terms]  
 # If a tool column is provided, create the 'finalized\_tool' column  
 if joined\_term:  
 plural\_joined\_term = joined\_term + "s"  
 subset\_df[f'finalized\_{joined\_term}'] = np.where(subset\_df[joined\_term].notnull(), subset\_df[joined\_term], subset\_df[plural\_joined\_term])  
 subset\_df = subset\_df.drop(columns=[joined\_term, plural\_joined\_term])  
 subset\_df = subset\_df.rename(columns={f"finalized\_{joined\_term}": joined\_term})  
 melted\_df = pd.melt(subset\_df, id\_vars=[date\_column, id\_column, origin\_column, word\_count\_column], var\_name='terms', value\_name='counts')  
 melted\_df['scaled\_counts'] = melted\_df['counts'] / melted\_df[word\_count\_column]  
 total\_df = melted\_df.groupby([origin\_column, date\_column])[word\_count\_column].sum().reset\_index().rename(columns={word\_count\_column: 'total\_tokens'})  
 grouped\_df = melted\_df.groupby([origin\_column, date\_column, 'terms']).counts.sum().reset\_index()  
 scaled\_grouped\_df = melted\_df.groupby([origin\_column, date\_column, 'terms']).scaled\_counts.sum().reset\_index()  
 final\_grouped\_df = pd.merge(grouped\_df, scaled\_grouped\_df, on=[origin\_column, date\_column, 'terms'])  
 final\_grouped\_df = pd.merge(final\_grouped\_df, total\_df, on=[origin\_column, date\_column])  
 final\_grouped\_df.scaled\_counts = final\_grouped\_df.scaled\_counts \* 100  
 final\_grouped\_df = final\_grouped\_df.rename(columns={origin\_column: 'data\_origin', date\_column: 'date'})  
 final\_grouped\_df["title"] = final\_grouped\_df.data\_origin  
  
 return final\_grouped\_df  
  
#END HERMENEUTICS

# Check if the chart file exists  
if os.path.exists(tool\_infrastructure\_chart\_output\_path) and rerun\_chart\_code == False:  
 # Load the chart from the JSON file  
 with open(tool\_infrastructure\_chart\_output\_path) as json\_file:  
 infrastructure\_tool\_chart\_dict = json.load(json\_file)  
 infrastructure\_tool\_chart = alt.Chart.from\_dict(infrastructure\_tool\_chart\_dict)  
else:  
  
 # Process the DataFrame to calculate term frequencies and tokenize text  
 grouped\_term\_index\_conferences\_df = process\_data(  
 file\_path=f"{article\_data\_directory\_path}/weingart\_et\_al\_data/dh\_conferences\_works.csv",  
 text\_column="full\_text",  
 date\_column="conference\_year",  
 terms\_list=terms,  
 data\_origin="Index of DH Conferences",  
 title="Index of DH Conferences by Weingart et al (2023)",  
 term\_type="terms",  
 term\_mapping=False,  
 joined\_term="tool"  
 )  
  
 # Process the DataFrame to calculate term frequencies and tokenize text  
 grouped\_term\_dhq\_df = process\_data(  
 file\_path=f"{article\_data\_directory\_path}/scraped\_dhq\_data/dhq\_articles.csv",  
 text\_column="body\_text",  
 date\_column="date\_processed",  
 terms\_list=terms,  
 data\_origin="DHQ",  
 title="DHQ Articles (2023)",  
 term\_type="terms",  
 term\_mapping=False,  
 joined\_term="tool"  
 )  
 if os.path.exists(f"{article\_data\_directory\_path}constellate\_data/jstor\_data\_infrastructure\_tool.csv") == False:  
 jstor\_data = pd.read\_json(f"{article\_data\_directory\_path}constellate\_data/private\_data/jstor\_data-jsonl.jsonl", lines=True)  
 jstor\_data['datePublished'] = pd.to\_datetime(jstor\_data['datePublished'])  
 grouped\_jstor\_data = process\_jstor\_data(jstor\_data, terms, 'datePublished', 'id', 'isPartOf', 'wordCount', joined\_term='tool')  
 else:  
 grouped\_jstor\_data = pd.read\_csv(f"{article\_data\_directory\_path}constellate\_data/jstor\_data\_infrastructure\_tool.csv")  
  
 term\_combined\_df = pd.concat([grouped\_term\_index\_conferences\_df, grouped\_term\_dhq\_df, grouped\_jstor\_data])  
 term\_combined\_df = term\_combined\_df.reset\_index(drop=True)  
 term\_combined\_df = term\_combined\_df.rename(columns={"counts": "Raw Frequency of Mentions", "scaled\_counts": "Frequency of Mentions Scaled by Total Tokens"})  
  
 term\_combined\_df.date = term\_combined\_df.date.astype(str)  
 term\_combined\_df.date = term\_combined\_df.date.str.split(" ").str[0]  
 term\_combined\_df['date'] = pd.to\_datetime(term\_combined\_df.date, format='%Y-%m-%d')  
  
  
 dropdown = alt.binding\_select(  
 options=['Raw Frequency of Mentions', 'Frequency of Mentions Scaled by Total Tokens'],   
 name='Select Counting Method on Y Axis'  
 )  
 ycol\_param = alt.param(  
 value="Raw Frequency of Mentions",  
 bind=dropdown  
 )  
  
 infrastructure\_term = term\_combined\_df[term\_combined\_df.terms == 'infrastructure']  
 selection = alt.selection\_point(fields=['data\_origin'], bind='legend')  
 infrastructure\_chart = alt.Chart(infrastructure\_term).mark\_bar().encode(  
 x=alt.X('year(date):T', axis=alt.Axis(title="Year")),  
 y=alt.Y('y:Q', axis=alt.Axis(title="Frequency of Mentions")),  
 color=alt.Color('data\_origin:N', legend=alt.Legend(title="Data Origin", symbolLimit=0, labelLimit=0), scale=alt.Scale( scheme='tableau20'), sort=alt.EncodingSortField(field="counts", op="sum", order='descending')),  
 tooltip=[  
 alt.Tooltip('data\_origin', title='Data Origin'),  
 alt.Tooltip('Raw Frequency of Mentions'),  
 alt.Tooltip('date:T', title='Date', format='%Y'),  
 alt.Tooltip('Frequency of Mentions Scaled by Total Tokens', title="Frequency of Mentions Scaled by Total Tokens (Percentage)", format='.2f'),  
 alt.Tooltip('total\_tokens', title='Total Tokens'),  
 ],  
 opacity=alt.condition(selection, alt.value(1), alt.value(0.2)),  
 ).transform\_calculate(  
 y=f'datum[{ycol\_param.name}]'  
 ).add\_params(  
 ycol\_param, selection  
 ).properties(  
 title="Popularity of the Term 'Infrastructure' Over Time and Across Multiple Datasets",  
 width=500  
 )  
  
 tool\_term = term\_combined\_df[term\_combined\_df.terms == 'tool']  
  
 selection = alt.selection\_point(fields=['data\_origin'], bind='legend')  
  
 tool\_chart = alt.Chart(tool\_term).mark\_bar().encode(  
 x=alt.X('year(date):T', axis=alt.Axis(title="Year")),  
 y=alt.Y('y:Q', axis=alt.Axis(title="Frequency of Mentions")),  
 color=alt.Color('data\_origin:N', legend=alt.Legend(title="Data Origin", symbolLimit=0, labelLimit=0), scale=alt.Scale( scheme='tableau20'), sort=alt.EncodingSortField(field="counts", op="sum", order='descending')),  
 tooltip=[  
 alt.Tooltip('data\_origin', title='Data Origin'),  
 alt.Tooltip('Raw Frequency of Mentions'),  
 alt.Tooltip('date:T', title='Date', format='%Y'),  
 alt.Tooltip('Frequency of Mentions Scaled by Total Tokens', title="Frequency of Mentions Scaled by Total Tokens (Percentage)", format='.2f'),  
 alt.Tooltip('total\_tokens', title='Total Tokens'),  
 ],  
 opacity=alt.condition(selection, alt.value(1), alt.value(0.2)),  
 ).transform\_calculate(  
 y=f'datum[{ycol\_param.name}]'  
 ).add\_params(  
 ycol\_param, selection  
 ).properties(  
 title=f"Popularity of the Term 'Tool' Over Time and Across Multiple Datasets",  
 width=500  
 )  
  
 infrastructure\_tool\_chart = alt.vconcat(infrastructure\_chart, tool\_chart).configure(  
 title=alt.TitleConfig(fontSize=20, font='Garamond', anchor='middle'),  
 axis=alt.AxisConfig(labelFont='Garamond', titleFont='Garamond'),  
 legend=alt.LegendConfig(labelFont='Garamond', titleFont='Garamond')  
 )  
 infrastructure\_tool\_chart.save(tool\_infrastructure\_chart\_output\_path)

#START HERMENEUTICS  
  
infrastructure\_tool\_chart  
  
#END HERMENEUTICS

alt.VConcatChart(...)

START HERMENEUTICS

These charts show that whether using raw frequencies or scaling by length of articles over time, both tools and infrastructure have become increasingly popular in DH. You can also select each publication in the legend to see these trends more clearly.

While this is not surprising, it does suggest that these terms are worth further investigation, especially since they are often used in DH without being defined or even interrogated. Indeed, while tool has been present from the outset of DH writing (though increasingly prevalent since 2005), infrastructure seems to be a more recent discourse, only emerging around 2000 and then becoming very popular; a finding we hope to explore more in-depth in the future.

END HERMENEUTICS

While histories of DH have started to proliferate, there remain few that foreground individual tools or methodologies, though certainly scholars like Julia Nyhan and Melissa Terras have recovered parts of this history, whether through complicating our histories of Robert Busa’s concordance programs or tracing the careers of the first DH developers, like John Bradley who started developing tools for text analysis in the 1970s (*Terras, Nyhan 2016*,*Nyhan, Flinn 2016*). If we primarily focus then on tools, unsurprisingly we find that the advent of mass digitization and the web in the 1990s and early 2000s was a moment of intense debate over tools (as indicated in our visualizations). Indeed, in 2003, John Unsworth, reflecting on the last decade of humanities computing, questioned whether the field was “going around in circles” because of the consistent refrain for needing better tools for humanities research (*Unsworth 2003*). To overcome these obstacles, Unsworth proposed the framing of ‘scholarly primitives’ to elucidate the theoretical and shared functions that undergird humanities research “that could be the basis for a manageable but also useful tool-building enterprise in humanities computing.” Part of Unsworth’s primitives was trying to envision the potential of DH tools, but it was also intended to argue for the necessity of these tools: both to grant funders and especially to skeptical humanists (*Unsworth 2000*). Such a focus was also central to the 2006 Summit on Digital Tools for the Humanities, also chaired by Unsworth, which argued “that only about six percent of humanist scholars go beyond general purpose information technology and use digital resources and more complex digital tools in their scholarship” (14009734/8DC6SYTG). This claim was further supported by Patrick Juola, who through citation analysis showed that very few high profile humanists had published or even engaged with humanities computing, though journals like *Computers and the Humanities* had been in print since the 1960s. This lack of engagement, especially given the growing abundance of born digital and digitized materials, led many to conclude that most DH tools were too bespoke for the average humanist, and that the solution was to focus on creating shared, large-scale infrastructure, or what Juola termed a “Killer App” (*Juola 2008*). Realizing this dream of wide-spread uptake of DH tools was also often tied to calls for standardizing tool infrastructures that was partially realized in projects like TEI and the maturing of web technologies (*Sinclair 2005*). But even without these shared standards or killer apps, this sense that large scale software development as the solution was undoubtedly influential in the development of tools like *Zotero* and *Voyant Tools*, along with network analysis tools like *Gephi*, whose developers described being “inspired by WYSIWYG editors like Adobe Photoshop” (*Bastian, Heymann, Jacomy 2009*, 361).

However, *Gephi* itself was not initially conceived of or built as primarily a tool for digital humanities research. According to Clément Levallois, the idea for *Gephi* was first developed in 2006 by Mathieu Jacomy, as part of his work with the e-Diasporas Atlas project and, and by Eytan Adar, as part of his work on GUESS, the Graph Exploration System Software (*Levallois 2017*). Together with researchers from Université Technologique de Compiègne, the team worked on developing the initial version of *Gephi*, releasing it for download on September 8, 2008, and then making the software public on GitHub on March 2, 2009. Part of the impetus for *Gephi*, according to one of the developers Sebastien Heymann, was that “other network visualization software is either a poor visualization module or a stunning aspect but not efficient.” For example, one of the earliest network analysis tools, Pajek, first released in 1997, would only let you visualize a few hundred nodes, and GUESS, released in 2007, could only display up to 2000 nodes. Thus, the creators of *Gephi* initially focused on leveraging new graphical technologies like GPUs to render much larger networks; the initial release could do up to 50000 nodes (*Heymann 2008*). But the very existence of this software was also indicative of the massive growth of network science as a field during this period. As the authors of The Network Turn have argued, the 1998 publication “Collective Dynamics of Small World Networks” by Duncan J. Watts and Steven Strogatz and the 1999 publication “Emergence of Scaling in Random Networks” by Albert-László Barabási and Réka Albert “ushered in the field of network science,” in many ways parallel to the coterminous growth of digital humanities (*Ahnert, Ahnert, Coleman, Weingart 2020*, 21). And yet, much of this initial debate over DH tools was focused almost exclusively on tools for working with digitized texts (whether editing and annotating or analyzing them).

START HERMENEUTICS

To see more on the history of *Gephi*, we also explored the early commit history <https://github.com/gephi/gephi/commit/c1a1b8316427b1e5c6e0ea5ea8031a016b7ba2c1> and web archived versions of their website <https://web.archive.org/web/20080912001706/https://gephi.org/> .

We can also get a sense of this transformation through exploring the rise of networks and network analysis in DH writing, using our same methods as before.

END HERMENEUTICS

#START HERMENEUTICS  
  
terms = ['network', 'networks', 'network analysis']  
network\_analysis\_chart\_output\_path = f"{article\_chart\_directory\_path}network\_analysis\_chart.json"  
rerun\_chart\_code = False # Set to True to rerun the code to generate the charts  
  
#END HERMENEUTICS

#START HERMENEUTICS  
  
# Check if the chart file exists  
if os.path.exists(network\_analysis\_chart\_output\_path) and rerun\_chart\_code == False:  
 # Load the chart from the JSON file  
 with open(network\_analysis\_chart\_output\_path) as json\_file:  
 network\_analysis\_chart\_dict = json.load(json\_file)  
 network\_analysis\_chart = alt.Chart.from\_dict(network\_analysis\_chart\_dict)  
else:  
  
 # Process the DataFrame to calculate term frequencies and tokenize text  
 grouped\_term\_index\_conferences\_df = process\_data(  
 file\_path=f"{article\_data\_directory\_path}/weingart\_et\_al\_data/dh\_conferences\_works.csv",  
 text\_column="full\_text",  
 date\_column="conference\_year",  
 terms\_list=terms,  
 data\_origin="Index of DH Conferences",  
 title="Index of DH Conferences by Weingart et al (2023)",  
 term\_type="terms",  
 term\_mapping=False,  
 joined\_term="network"  
 )  
  
 # Process the DataFrame to calculate term frequencies and tokenize text  
 grouped\_term\_dhq\_df = process\_data(  
 file\_path=f"{article\_data\_directory\_path}scraped\_dhq\_data/dhq\_articles.csv",  
 text\_column="body\_text",  
 date\_column="date\_processed",  
 terms\_list=terms,  
 data\_origin="DHQ",  
 title="DHQ Articles (2023)",  
 term\_type="terms",  
 term\_mapping=False,  
 joined\_term="network"  
 )  
  
 if os.path.exists(f"{article\_data\_directory\_path}constellate\_data/jstor\_data\_network\_analysis.csv") == False:  
 jstor\_data = pd.read\_json(f"{article\_data\_directory\_path}constellate\_data/jstor\_data-jsonl.jsonl", lines=True)  
 jstor\_data['datePublished'] = pd.to\_datetime(jstor\_data['datePublished'])  
 grouped\_jstor\_data = process\_jstor\_data(jstor\_data, terms, 'datePublished', 'id', 'isPartOf', 'wordCount', joined\_term='network')  
 else:  
 grouped\_jstor\_data = pd.read\_csv(f"{article\_data\_directory\_path}constellate\_data/jstor\_data\_network\_analysis.csv")  
  
 term\_combined\_df = pd.concat([grouped\_term\_index\_conferences\_df, grouped\_term\_dhq\_df, grouped\_jstor\_data])  
 term\_combined\_df = term\_combined\_df.reset\_index(drop=True)  
 term\_combined\_df = term\_combined\_df.rename(columns={"counts": "Raw Frequency of Mentions", "scaled\_counts": "Frequency of Mentions Scaled by Total Tokens"})  
  
 term\_combined\_df.date = term\_combined\_df.date.astype(str)  
 term\_combined\_df.date = term\_combined\_df.date.str.split(" ").str[0]  
 term\_combined\_df['date'] = pd.to\_datetime(term\_combined\_df.date, format='%Y-%m-%d')  
  
  
 dropdown = alt.binding\_select(  
 options=['Raw Frequency of Mentions', 'Frequency of Mentions Scaled by Total Tokens'],   
 name='Select Counting Method on Y Axis'  
 )  
 ycol\_param = alt.param(  
 value="Raw Frequency of Mentions",  
 bind=dropdown  
 )  
  
 network\_term = term\_combined\_df[term\_combined\_df.terms == 'network']  
 selection = alt.selection\_point(fields=['data\_origin'], bind='legend')  
 network\_chart = alt.Chart(network\_term).mark\_bar().encode(  
 x=alt.X('year(date):T', axis=alt.Axis(title="Year")),  
 y=alt.Y('y:Q', axis=alt.Axis(title="Frequency of Mentions")),  
 color=alt.Color('data\_origin:N', legend=alt.Legend(title="Data Origin", symbolLimit=0, labelLimit=0), scale=alt.Scale( scheme='tableau20'), sort=alt.EncodingSortField(field="counts", op="sum", order='descending')),  
 tooltip=[  
 alt.Tooltip('data\_origin', title='Data Origin'),  
 alt.Tooltip('Raw Frequency of Mentions'),  
 alt.Tooltip('date:T', title='Date', format='%Y'),  
 alt.Tooltip('Frequency of Mentions Scaled by Total Tokens', title="Frequency of Mentions Scaled by Total Tokens (Percentage)", format='.2f'),  
 alt.Tooltip('total\_tokens', title='Total Tokens'),  
 ],  
 opacity=alt.condition(selection, alt.value(1), alt.value(0.2)),  
 ).transform\_calculate(  
 y=f'datum[{ycol\_param.name}]'  
 ).add\_params(  
 ycol\_param, selection  
 ).properties(  
 title="Popularity of the Term 'Network' Over Time and Across Multiple Datasets",  
 width=500  
 )  
  
 network\_analysis\_term = term\_combined\_df[term\_combined\_df.terms == 'network analysis']  
  
 selection = alt.selection\_point(fields=['data\_origin'], bind='legend')  
  
 network\_analysis\_chart = alt.Chart(network\_analysis\_term).mark\_bar().encode(  
 x=alt.X('year(date):T', axis=alt.Axis(title="Year")),  
 y=alt.Y('y:Q', axis=alt.Axis(title="Frequency of Mentions")),  
 color=alt.Color('data\_origin:N', legend=alt.Legend(title="Data Origin", symbolLimit=0, labelLimit=0), scale=alt.Scale( scheme='tableau20'), sort=alt.EncodingSortField(field="counts", op="sum", order='descending')),  
 tooltip=[  
 alt.Tooltip('data\_origin', title='Data Origin'),  
 alt.Tooltip('Raw Frequency of Mentions'),  
 alt.Tooltip('date:T', title='Date', format='%Y'),  
 alt.Tooltip('Frequency of Mentions Scaled by Total Tokens', title="Frequency of Mentions Scaled by Total Tokens (Percentage)", format='.2f'),  
 alt.Tooltip('total\_tokens', title='Total Tokens'),  
 ],  
 opacity=alt.condition(selection, alt.value(1), alt.value(0.2)),  
 ).transform\_calculate(  
 y=f'datum[{ycol\_param.name}]'  
 ).add\_params(  
 ycol\_param, selection  
 ).properties(  
 title=f"Popularity of the Term 'Network Analysis' Over Time and Across Multiple Datasets",  
 width=500  
 )  
  
 network\_analysis\_chart = alt.vconcat(network\_chart, network\_analysis\_chart).configure(  
 title=alt.TitleConfig(fontSize=20, font='Garamond', anchor='middle'),  
 axis=alt.AxisConfig(labelFont='Garamond', titleFont='Garamond'),  
 legend=alt.LegendConfig(labelFont='Garamond', titleFont='Garamond')  
 )  
 network\_analysis\_chart.save(network\_analysis\_chart\_output\_path)  
  
#END HERMENEUTICS

#START HERMENEUTICS  
  
network\_analysis\_chart  
  
#END HERMENEUTICS

alt.VConcatChart(...)

START HERMENEUTICS

Similar to our graph above, these charts visualize how networks and network analysis have increasingly become popular in DH, even when scaled by the length of articles. You can also select each publication in the legend to see these trends more clearly. While there seems to be a post-2020 decline, this decrease is likely an artefact of the pandemic given the rebound in 2022 (2023 data was not complete at the time of writing).

Also of note is that while networks, like tools, have been present in DH writing since the beginning, network analysis is a more recent phenomenon, only emerging in the mid-2000s and then becoming increasingly popular. This trend seems to correspond with the growing popularity of network analysis tools, like *Gephi*, which we explored in the introduction.

END HERMENEUTICS

Since then, tools for network analysis, like *Gephi*, have become very popular in the field. But we believe that part of this shift is not only the availability of these tools, but also the shift in how DH as a field approaches these tools, both in terms of making and theorizing them, and that understanding these shifts is critical for contextualizing *Network Navigator*.

To understand the growth then of network analysis tools, we need to return to the aftermath of the 2006 Summit on DH Tools, where one of the main takeaways was that these digital tools needed further theorizing to uncover why they were not being more widely adopted or why more were not being built in the first place. To that end, Susan Schreibman and Ann Hanlon completed the first survey of DH tool developers in 2008 to understand how the creators of these tools conceived of their efforts. While the survey focused on a mix of “branded” and “unbranded” DH tools, the main finding was that “overwhelming majority of respondents (94%) considered tool development a scholarly activity, although the range of responses to this question made it clear that many departments and institutions do not” (*Schreibman, Hanlon 2010*). Part of the solution to this disconnect was increased advocacy on evaluating and crediting digital scholarship through scholarly associations like the MLA and AHA, but there was also a growing debate over whether tools should be considered not just scholarship, but also as theoretical objects into themselves.

For DH developers like John Bradley, there was a sense that “in early discussions on the Humanist discussion list (for example in the late 1980s and early 1990s), the term ‘tool’ often carried a pejorative tone, with the implication of considering something as ‘merely a tool’” (*Bradley 2023*, 153-154). But the proliferation of new digital tools led scholars like Willard McCarty in a Humanist exchange to question whether “digital artifacts” could be interpreted without some paratextual explanation: in essence, could digital tools function as other forms of scholarly objects or were they simply a means to an end. Stephen Ramsay and Geoffrey Rockwell sought to directly answer this question, arguing that if building functioned similarly to writing technologies, then “the quality of the interventions that occur as a result of building are as interesting as those that are typically established through writing, then that activity is, for all intents and purposes, scholarship” (*Ramsay, Rockwell 2012*, 83). Ramsay had proposed a similar assessment earlier in 2011 at the MLA arguing for building and making as “a new kind of hermeneutic,” though his assertion became somewhat overshadowed by his framing of defining who ‘counted’ as a digital humanist (*Ramsay 2016*). But even before what eventually became the “hack versus yack” debates, there was growing awareness but also anxiety over digital tools, as Julia Flanders articulated, “Digital tools add a challenge and give us a new set of terms — like a new planet in the system, they change the vectors of all the other things we have in our universe. They will probably change the way humanities research is done” (*Flanders 2009*).

Assessing this “new planet” of digital tools then would require not only new forms of credit for creating tools, but also the development of new scholarly infrastructure and frameworks that centered tools as objects of study. Part of this transformation was evidenced in growing calls for peer review of these tools. Initially advocated by Alan Galey and Stan Ruecker who, building from Lev Manovich’s response to McCarthy that “a prototype is a theory”, proposed that prototypes should be peer reviewed like other forms of scholarship. For Galey and Ruecker, prototypes were a capacious category that included tools, along with many ‘digital objects.’ But regardless of terminology, the authors argued that peer reviewing would serve as a “useful bridge between the tool-building tradition of the digital humanities on one hand, and interpretive and critical traditions like book history and science and technology studies on the other,” helping move tools from being seen as service-based outputs towards inquiry-based ones (*Galey, Ruecker 2010*, 406). The necessity of such peer review mechanisms was further reinforced by Annie Swafford with her analysis of Matthew Jockers’ sentiment analysis library “Syuzhet,” which eventually became the subject of a protracted Twitter debate over the assumptions built into the tool, something Swafford felt that formal peer review would have been better suited to resolving. To avoid future Syuzhet debates and to realize Galey and Ruecker’s vision, many academic journals have increasingly developed mechanisms for reviewing digital projects and even tools (*Swafford 2016*). But even more profound has been the development of more robust theoretical frameworks for analyzing tools that not only detail the functionalities of these tools, but also explore the socio-political assumptions built into these tools.

The most prominent framework to this end has been the concept of “tool criticism”, which was inspired by earlier work on digital source criticism and algorithmic criticism, as well as the emergence of Critical Code Studies and Software Studies both of which treat code as an object for scholarship. As far as we have been able to determine, the first call for this type of work was in a Master’s thesis by Anastasia Dorofeeva titled “Towards Digital Humanities Tool Criticism” published in 2014 from the University of Leiden (*Dorofeeva 2014*). Since then, the concept has quickly become popular as a methodology that reflexively attends to both the technical and socio-political nature of DH tools (*van Es, Es 2023*,*Koolen, van Gorp, van Ossenbruggen 2019*). In defining this approach, Karin Van Es, Maranke Wieringa, and Mirko Tobias Schäfer argue that it not only embodies a critical approach to computation, like the ones advocated by CCS scholars like David M. Berry, but also “incorporates reflection on the interaction between the researcher and the tool” and at the same time collapses “distinction of tool builder and researcher” (*Van Es, Wieringa, Schäfer 2018*, 26).

START HERMENEUTICS

To explore the emergence of these theoretical approaches to both computational methods and tools, we can once again turn to DH writing, using our same methods as before.

END HERMENEUTICS

#START HERMENEUTICS  
  
terms = ['tool criticism', 'algorithmic criticism']  
tool\_algo\_criticism\_chart\_output\_path = f"{article\_chart\_directory\_path}tool\_algo\_criticism\_chart.json"  
rerun\_chart\_code = False # Set to True to rerun the code to generate the charts  
  
#END HERMENEUTICS

#START HERMENEUTICS  
  
# Check if the chart file exists  
if os.path.exists(tool\_algo\_criticism\_chart\_output\_path) and rerun\_chart\_code == False:  
 # Load the chart from the JSON file  
 with open(tool\_algo\_criticism\_chart\_output\_path) as json\_file:  
 tool\_algo\_criticism\_chart\_dict = json.load(json\_file)  
 tool\_algo\_criticism\_chart = alt.Chart.from\_dict(tool\_algo\_criticism\_chart\_dict)  
else:  
  
 # Process the DataFrame to calculate term frequencies and tokenize text  
 grouped\_term\_index\_conferences\_df = process\_data(  
 file\_path=f"{article\_data\_directory\_path}weingart\_et\_al\_data/dh\_conferences\_works.csv",  
 text\_column="full\_text",  
 date\_column="conference\_year",  
 terms\_list=terms,  
 data\_origin="Index of DH Conferences",  
 title="Index of DH Conferences by Weingart et al (2023)",  
 term\_type="terms",  
 term\_mapping=False,  
 joined\_term=None  
 )  
  
 # Process the DataFrame to calculate term frequencies and tokenize text  
 grouped\_term\_dhq\_df = process\_data(  
 file\_path=f"{article\_data\_directory\_path}scraped\_dhq\_data/dhq\_articles.csv",  
 text\_column="body\_text",  
 date\_column="date\_processed",  
 terms\_list=terms,  
 data\_origin="DHQ",  
 title="DHQ Articles (2023)",  
 term\_type="terms",  
 term\_mapping=False,  
 joined\_term=None  
 )  
  
 if os.path.exists(f"{article\_data\_directory\_path}constellate\_data/jstor\_data\_tool\_algorithmic\_criticism.csv") == False:  
 jstor\_data = pd.read\_json(f"{article\_data\_directory\_path}constellate\_data/jstor\_data-jsonl.jsonl", lines=True)  
 jstor\_data['datePublished'] = pd.to\_datetime(jstor\_data['datePublished'])  
 grouped\_jstor\_data = process\_jstor\_data(jstor\_data, terms, 'datePublished', 'id', 'isPartOf', 'wordCount', joined\_term=None)  
 else:  
 grouped\_jstor\_data = pd.read\_csv(f"{article\_data\_directory\_path}constellate\_data/jstor\_data\_tool\_algorithmic\_criticism.csv")  
  
 term\_combined\_df = pd.concat([grouped\_term\_index\_conferences\_df, grouped\_term\_dhq\_df, grouped\_jstor\_data])  
 term\_combined\_df = term\_combined\_df.reset\_index(drop=True)  
 term\_combined\_df = term\_combined\_df.rename(columns={"counts": "Raw Frequency of Mentions", "scaled\_counts": "Frequency of Mentions Scaled by Total Tokens"})  
  
 term\_combined\_df.date = term\_combined\_df.date.astype(str)  
 term\_combined\_df.date = term\_combined\_df.date.str.split(" ").str[0]  
 term\_combined\_df['date'] = pd.to\_datetime(term\_combined\_df.date, format='%Y-%m-%d')  
  
  
 dropdown = alt.binding\_select(  
 options=['Raw Frequency of Mentions', 'Frequency of Mentions Scaled by Total Tokens'],   
 name='Select Counting Method on Y Axis'  
 )  
 ycol\_param = alt.param(  
 value="Raw Frequency of Mentions",  
 bind=dropdown  
 )  
  
 tool\_criticism\_term = term\_combined\_df[term\_combined\_df.terms == 'tool criticism']  
 selection = alt.selection\_point(fields=['data\_origin'], bind='legend')  
 tool\_criticism\_chart = alt.Chart(tool\_criticism\_term).mark\_bar().encode(  
 x=alt.X('year(date):T', axis=alt.Axis(title="Year")),  
 y=alt.Y('y:Q', axis=alt.Axis(title="Frequency of Mentions")),  
 color=alt.Color('data\_origin:N', legend=alt.Legend(title="Data Origin", symbolLimit=0, labelLimit=0), scale=alt.Scale( scheme='tableau20'), sort=alt.EncodingSortField(field="counts", op="sum", order='descending')),  
 tooltip=[  
 alt.Tooltip('data\_origin', title='Data Origin'),  
 alt.Tooltip('Raw Frequency of Mentions'),  
 alt.Tooltip('date:T', title='Date', format='%Y'),  
 alt.Tooltip('Frequency of Mentions Scaled by Total Tokens', title="Frequency of Mentions Scaled by Total Tokens (Percentage)", format='.2f'),  
 alt.Tooltip('total\_tokens', title='Total Tokens'),  
 ],  
 opacity=alt.condition(selection, alt.value(1), alt.value(0.2)),  
 ).transform\_calculate(  
 y=f'datum[{ycol\_param.name}]'  
 ).add\_params(  
 ycol\_param, selection  
 ).properties(  
 title="Popularity of the Term 'Tool Criticism' Over Time and Across Multiple Datasets",  
 width=500  
 )  
  
 algorithmic\_criticism\_term = term\_combined\_df[term\_combined\_df.terms == 'algorithmic criticism']  
  
 selection = alt.selection\_point(fields=['data\_origin'], bind='legend')  
  
 algorithmic\_criticism\_chart = alt.Chart(algorithmic\_criticism\_term).mark\_bar().encode(  
 x=alt.X('year(date):T', axis=alt.Axis(title="Year")),  
 y=alt.Y('y:Q', axis=alt.Axis(title="Frequency of Mentions")),  
 color=alt.Color('data\_origin:N', legend=alt.Legend(title="Data Origin", symbolLimit=0, labelLimit=0), scale=alt.Scale( scheme='tableau20'), sort=alt.EncodingSortField(field="counts", op="sum", order='descending')),  
 tooltip=[  
 alt.Tooltip('data\_origin', title='Data Origin'),  
 alt.Tooltip('Raw Frequency of Mentions'),  
 alt.Tooltip('date:T', title='Date', format='%Y'),  
 alt.Tooltip('Frequency of Mentions Scaled by Total Tokens', title="Frequency of Mentions Scaled by Total Tokens (Percentage)", format='.2f'),  
 alt.Tooltip('total\_tokens', title='Total Tokens'),  
 ],  
 opacity=alt.condition(selection, alt.value(1), alt.value(0.2)),  
 ).transform\_calculate(  
 y=f'datum[{ycol\_param.name}]'  
 ).add\_params(  
 ycol\_param, selection  
 ).properties(  
 title=f"Popularity of the Term 'Algorithmic Criticism' Over Time and Across Multiple Datasets",  
 width=500  
 )  
  
 tool\_algo\_criticism\_chart = alt.vconcat(tool\_criticism\_chart, algorithmic\_criticism\_chart).configure(  
 title=alt.TitleConfig(fontSize=20, font='Garamond', anchor='middle'),  
 axis=alt.AxisConfig(labelFont='Garamond', titleFont='Garamond'),  
 legend=alt.LegendConfig(labelFont='Garamond', titleFont='Garamond')  
 )  
 tool\_algo\_criticism\_chart.save(tool\_algo\_criticism\_chart\_output\_path)  
  
#END HERMENEUTICS

#START HERMENEUTICS  
  
tool\_algo\_criticism\_chart  
  
#END HERMENEUTICS

alt.VConcatChart(...)

START HERMENEUTICS

Our results in these charts emphasize the relative novelty of tool and algorithmic criticism, but also the growing popularity of these terms in DH. You can also select each publication in the legend to see these trends more clearly.

END HERMENEUTICS

In the case of *Network Navigator*, tool criticism has been particularly helpful for reflecting on our design and development choices, though we were not directly familiar with it when building *Network Navigator*. But tool criticism is also broadly relevant since many scholars have focused on *Gephi* and network analysis as a case study for this framework. For example, Van Es et al utilize tool criticism to critically theorize *Gephi*’s popularity and also elucidate its constraints, such as the ways in which the tool “gives a large audience the ability to produce network diagrams without understanding the ‘layers of mediation’ involved in its production.” To help make these layers visible, the authors detail creating the “Gephi field notes plugin,” which takes snapshots of a network in *Gephi*, along with the transformations required to create it and saves it as a text file (*Van Es, Wieringa, Schäfer 2018*, 25). In many ways, the creation of this plugin corresponds with our desire to create *Network Navigator*, as attempts to intervene in scholarly practices. And similarly, both our projects also build from Johanna Drucker and Patrick Svensson’s concept of “middleware.” Though the concept of middleware is somewhat more narrow than tool criticism, in a 2016 DHQ article, Drucker and Svensson argued that greater scholarly attention was needed to uncover how software (whether tools or platforms) shape scholarly arguments. The authors highlighted the example of *Gephi* diagrams, describing how the popularity of force-layout diagrams elide how these visualizations are “not a natural expression of structured data, it is instead a distinct expression in a system authored by a programmer responsible for the coming-into-being of that visual form in a precise form” (*Drucker, Svensson 2016*). More recent work has continued to theorize *Gephi* through the lens of tool criticism, including work by Daniela Van Geenen, who has published two recent articles (the second of which includes some of the lead developers of *Gephi*). In the first, she uses tool criticism, along with the framing of “epistemological affordances,” to explore how *Gephi*’s interface and documentation enables users to follow the progression of their work and scrutinize the methods applied within the tool. Of particular concern to Van Geenen is how *Gephi* limits “reversibility and reproducibility,” and she argues that the lack of undo or redo functionality undermines any algorithmic accountability or critical use of *Gephi* (*van Geenen 2020*, 18-19). While we agree with Van Geenen on the importance of understanding these algorithms, building this proposed undo/redo functionality for any network algorithm would require significant computational resources, which is why we have not included such features in *Network Navigator*. Nonetheless, we believe that our tool shares a similar core ethos to this scholarship; in essence, the belief that these tools should promote a critical technical practice that goes beyond mere tool functionality and considers the diverse uses of networks across and beyond scholarly communities. Which is why in our development of *Network Navigator* we have focused primarily on DH pedagogical needs, but have also been inspired by growing critiques of large scale infrastructure for DH research.

Though both Ladd and LeBlanc had previous experience building complex web applications, as mentioned previously *Network Navigator* is largely composed of foundational web technologies: HTML, CSS, and Javascript. Such a relatively simple technological stack is in contrast to tools like *Gephi* or *Voyant Tools*, both of which are built in Java with far more complex design patterns and architecture. But while our technical choices diverge from other DH tools, we are far from the first to consider privileging less as more. As early as 2012, scholars like Joris Van Zundert were arguing that “big institutionally-based digital infrastructures are a dead end for information technology development and application in the humanities.” While Van Zundert was critiquing proposals like the ones for Killer Apps, his assessments on the difficulties, if not outright impossibility, of building widely generalizable tools for humanities research that go beyond digitization echo much of the criticisms of tools like *Gephi* and correspond with our experiences teaching these tools, all of which has informed *Network Navigator*. In particular, Van Zundert called for “super-lightweight humanities research microservices” that would have buy-in from developers and be low-cost to maintain as an alternative future for “tool building in the digital humanities [that] is still in the process of maturing” (*van Zundert 2012*, 182-184). In 2016, Dennis Tenen made a similar argument of how “big software” was increasingly unfit for the growing “computational turn” in DH and other disciplines, which with the rise of data analysis programming and coding libraries made building these tools not only somewhat unnecessary, but also ill-suited to understanding their underlying computational methodologies. Instead, Tenen argued that experimentation and iteration were critical, and that ultimately, “the best kind of tools are therefore the ones that we make ourselves” (*Tenen 2016*). While these critiques have yet to fully percolate into how DH tools are built, let alone funded, we believe these critiques were not isolated, but rather indicative and associated with the rise of minimal computing in DH.

First proposed as part of the <GO::DH> working group in the mid 2010s as the “application of minimalist principles to computing”, in a recent DHQ special issue dedicated to the topic, the term has evolved to become much more expansive to the point that the editors of the issue, Roopika Risam and Alex Gil argue that “defining minimal computing is as quixotic a task as defining digital humanities itself” (much like DH tools we would argue) (*Gil, Ortega 2016*,*Risam, Gil 2022*). But even if capacious, minimal computing has become a particularly influential framework, leading many scholars to pivot from complex web applications and databases towards simpler, more sustainable technologies like static site generators, exemplified by tools like Jekyll, which generate HTML and Javascript files without requiring a database. Such a shift is not without its tradeoffs, as Quinn Dombrowski has argued these technologies require “maximizing labor” since setting up a Jekyll site is much more labor intensive than a one click install of Wordpress (*Dombrowski 2022*). But even if it is not a panacea, minimal computing has been particularly important for critically considering the limits of GUI tools in DH, which as Risam and Gil argue often “hide the systems that drive that production, and by extension, the labor to maintain and sustain them” (*Risam, Gil 2022*). Their critique not only echoes tool criticism, but more profoundly, minimal computing also represents a path towards operationalizing these critiques in the process of building DH tools, through recentering not just technological considerations, but also the politics, labor, and material constraints inherent in building DH tools.

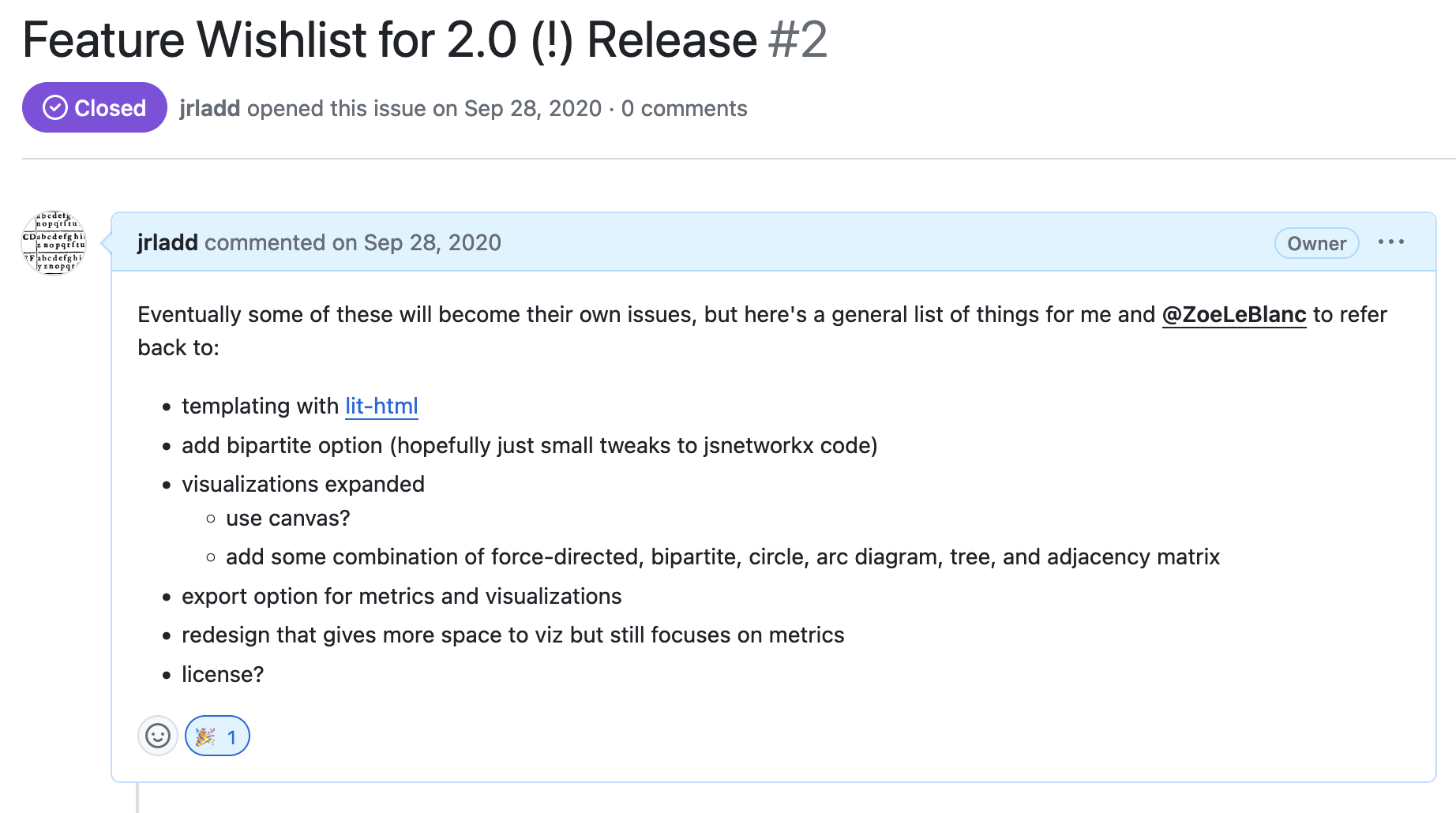
Stephen Ramsay has recently cautioned that even if we accept that our tools have politics, we should not conflate those politics with “acts of “resistance” or outright political activism” (*Ramsay 2023*). But even with this qualification, we believe that minimal computing, as a broader labor-centric framework rather than solely a technological one, is crucial for not only situating our approach with Network Navigator, but also the future political economy of DH tools. While scholars like Alan Liu, and even critics of DH such as David Golumbia, have long called for DH scholars to critically engage with scholarship on the neoliberal politics of our current techno-capitalist moment, such a perspective is only recently starting to be applied reflexively to consider DH labor practices and formations. As Andrew Pilsch and Shawna Ross have recently argued, DH scholarship too often leverages “passion and joy as an unconscious cover for exploitation,” rather than embodying and prioritizing the humanistic principles of “social justice, equity, criticality, and mutuality” in how we work (*Pilsch, Ross 2022*, 343). With *Network Navigator*, we have attempted to both bridge and implement tool criticism and minimal computing. This crucial perspective has helped us critically and intentionally build a DH tool that considers both the constraints of our labor (neither of us are full-time software developers) and the broader labor practices of DH when it comes to tools; that is the way these tools often function as the primary introduction to a computational method whether for those self-teaching or those in DH workshops and classes.

This perspective on the politics of DH labor is also imperative given the growing critique that DH tools have by-and-large failed. While some of this critique is centered on the financial inability of DH to support large-scale software, it is also a question of usability and fit for the research process. For instance, John Bradley writes, “it is natural that a toolmaker should want his/her tools to be used by others, and in this respect, my toolmaking for humanists (including Pliny) has to a large extent been a failure. My personal disappointment matches the experience of others too” (*Bradley 2019*, 13). For Bradley, this failure has been largely the result of a disconnect between tool functionality and how humanists want to create knowledge, an assessment that has increasingly become poignant through user studies of DH tools. In 2012, Fred Gibbs and Trevor Owens argued that rather than building more “sophisticated” tools, most DH scholars preferred ones that prioritized “ease of use and transparency” (*Gibbs, Owens 2012*). Similarly, Max Kemman and Martijn Kleppe have also argued that focusing on users of these tools makes visible how this fantasy of generalizability distorts how these tools need to be “compatible with specific research methods of scholars, taking a place in a wider research workflow.” For Kemman and Kleppe, this compatibility requires a shift in tool building practices moving “from developing tools for scholars, via developing tools with scholars, to scholars developing tools” – a shift that we have strived to realize with *Network Navigator* (*Kemman, Kleppe 2015*, 71). This question of usability is not simply about uptake of these tools though. As Paige Morgan has argued, “our framing for DH tools and resources shapes researchers’ emotions and expectations” and that this affective expectation of user-friendliness is not an obvious or uncomplicated goal but one that is often “abstract and intangible” (*Morgan 2018*, 211-213). Indeed, we have been inspired by these critiques, as well as the work of scholars of computing and STS like Michael Black who have started to historicize ideas of usability and user-friendliness, to consider how usability for DH scholars might mean more limited functionality in a tool (*Black 2022*). Such a perspective means that a tool like *Network Navigator* will never be the tool for network analysis in DH, and that our ultimate goal is not to prioritize number of users (or in business-speak: market-share) but rather to encourage the best use of these methods through our tool. To understand more how we implemented and debated these choices though, we need to return to our source code and the rebuilding of *Network Navigator*.

## Centrality: The (Re-) Making of Network Navigator and Moving Beyond The Hairball Visualization

As stated previously, the initial version of Network Navigator was built solely by Ladd. But after a chance meeting at the 2019 Association for Computers and the Humanities conference in Pittsburgh, where LeBlanc first met Ladd and shared her admiration for his project, as well as her concerns over how DH scholars were struggling to engage with network analysis more holistically, the two decided to collaborate on expanding Network Navigator to more directly intervene in these practices.

display(Image("media/nn\_feature\_wishlist.png"))



At their first meeting on September 28, 2020, the two developed a dream wish list of features that included using new HTML templating libraries like lit-html; adding functionality for bipartite networks; expanding visualizations; exporting metrics and visualizations; updated license; and most profoundly, a redesign that balanced visualizations with metrics. While these features may sound straightforward or obvious considering the initial *Network Navigator* interface, these were instead the result of much debate over how to leverage some of the latest technologies in web design and graphics, while also considering how to maintain a lightweight software design.

To that end, the first step was removing the Bootstrap styling library. Originally developed by Twitter in 2010 and publicly released in 2011, Bootstrap is a front-end framework that provides boilerplate code for styling web applications and websites (for instance, organizing columns, designing modal windows, implementing collapsible tables, or incorporating sizable buttons, etc.). Since its release Bootstrap has become incredibly popular, especially for single page applications (SPA), and is currently the eighteenth most starred repository on GitHub. This popularity is in part because Bootstrap makes it far easier to have responsive and aesthetic web applications, which previously required hand-coding many complicated styling elements. However, in our discussions we increasingly became concerned that Bootstrap was delimiting what was possible with the *Network Navigator* interface, rather than enabling our vision. Furthermore, adding Bootstrap to a project necessitates a massive styling library on each page load, which adds a lot of unnecessary weight to a small web project, especially considering that our interface was using few of its features. Part of our reconsideration of Bootstrap was also a larger discussion of whether we should use an even more robust web framework, like React or VueJS (both of which we had previous experience with). However, if Bootstrap was adding too much bloat, then these massively complex web frameworks were never going to be a great fit, especially considering our commitment to the principles of minimal computing and transparent tool design. Instead, though we considered using an HTML templating framework like lit-html, we decided to keep *Network Navigator* as lightweight as possible, hand coding our HTML and Javascript files (though we do utilize the JQuery library). We also decided to opt for lighter styling frameworks, in our case the library Tachyons, to have finer-grained control over the site’s appearance and distinguish it from other web interfaces, but without having to create custom CSS code that would be difficult to maintain over time. Ultimately, for us, it became important to interrogate the tool we were making, as well as the tools we were using to build the user interface itself. Our choices are also part of the growing un-adoption of modern web frameworks, as more and more developers realize that these frameworks, though helpful, are not the solution or a good fit for every web application or project (*Ferreira, Borges, Valente 2022*).

START HERMENEUTICS

For more on Bootstrap see <https://getbootstrap.com/docs/4.0/about/history/> and <https://github.com/EvanLi/Github-Ranking/blob/master/Top100/Top-100-stars.md>. While benchmarking the size of a library is somewhat dependent on internet bandwidth, earlier versions of Bootstrap could be up to ~116 KB when uncompressed (which does not include the Javascript files or other dependencies): <https://stackoverflow.com/questions/29288592/bootstrap-min-css-size-from-cdn-shows-a-surprising-size-in-devtools>.

For more on our adoption of Tachyons, see <https://github.com/jrladd/network_navigator/issues/3> and <https://github.com/jrladd/network_navigator/pull/7>.

END HERMENEUTICS

While the roads not taken with *Network Navigator* may be difficult to discern for users, we have attempted to build an interface for *Network Navigator* that intentionally makes users aware of itself as a structuring argument (in Drucker and Svensson’s sense). Specifically, the interface design is meant to guide the user to greater understanding of the capabilities of network analysis as a whole but also the intentional limitation of *Network Navigator* as a tool. Our emphasis on limitations may seem like a rationale, or dare we say even an excuse, for not building more functionality, but as stated previously, from the outset *Network Navigator* was designed to be more limited than a tool like *Gephi* since we see it as one part of a broader ecosystem. Therefore, the interface was crafted to encourage users to confront these limitations, which we believe helps them better understand what tools, libraries, and platforms for network analysis that they want to use next. For us then, assessing the usability of *Network Navigator* required thinking as much about what we wanted to omit or obscure from *Network Navigator* versus what we wanted to enable with this tool.

A central way in which *Network Navigator* calls attention to the many possibilities of network analysis is through its prominent presentation of a selected set of network metrics. Shortly before *Network Navigator* was first designed, the team members of *Six Degrees of Francis Bacon* created a tutorial for *The Programming Historian* that outlined how to calculate standard network metrics in Python (*Ladd, Otis, Warren, Weingart 2017*). The metrics included density, average shortest path length, diameter, and transitivity to measure the whole network, and degree, betweenness centrality, and eigenvector centrality to measure individual nodes. These metrics were chosen because they were already in regular use by the project team, and because the team noted these same metrics frequently occurring in other humanities network projects.

When first working with network data, humanities scholars are typically interested in two things: the importance of individual nodes and the overall network structure. Local metrics such as degree, betweenness, and eigenvector centrality are essential metrics for understanding a node’s importance. How central is a particular node in the network? Is a specific individual a hub (someone with a lot of connections) or a broker (someone who stands between several groups)? Global metrics such as density, average shortest path length, diameter, and transitivity all give a sense of the overall size and shape of a network. Is the community close-knit or spread out? Would information travel through the network quickly or slowly? The addition of these global metrics was the very first improvement to the site, made in its second commit on the next day, Aug. 9th. Below is a brief demonstration of how to calculate these metrics in Python, and the code outputs a table in a nearly identical way to the first version of *Network Navigator*.

#START HERMENEUTICS  
  
import networkx as nx  
import pandas as pd  
  
# Read CSV file  
edgelist = pd.read\_csv("https://raw.githubusercontent.com/melaniewalsh/sample-social-network-datasets/master/sample-datasets/quakers/quaker-edges.csv")  
  
# Create Graph Object  
G = nx.from\_pandas\_edgelist(edgelist, source="Source", target="Target")  
  
# Calculate Graph-level metrics  
print(f"""  
Data on the Six Degrees of Francis Bacon Quakers Group:  
  
{G}  
Density: {nx.density(G):.4f}  
Diameter: {nx.diameter(G)}  
Avg. Shortest Path Length: {nx.average\_shortest\_path\_length(G):.4f}  
Transitivity {nx.transitivity(G):.4f}  
""")  
  
# Calculate Node-level metrics  
degree = nx.degree\_centrality(G)  
betweenness = nx.betweenness\_centrality(G)  
eigenvector = nx.eigenvector\_centrality(G)  
  
# Add attributes to graph  
nx.set\_node\_attributes(G, degree, 'degree\_centrality')  
nx.set\_node\_attributes(G, betweenness, 'betweenness\_centrality')  
nx.set\_node\_attributes(G, eigenvector, 'eigenvector\_centrality')  
  
# Convert node table to Pandas DataFrame for display  
nodes = pd.DataFrame.from\_dict(G.nodes, orient='index')  
nodes.reset\_index(level=0,names="name",inplace=True)  
nodes.sort\_values("degree\_centrality", ascending=False)  
  
#END HERMENEUTICS

Data on the Six Degrees of Francis Bacon Quakers Group:  
  
Graph with 96 nodes and 162 edges  
Density: 0.0355  
Diameter: 8  
Avg. Shortest Path Length: 3.3789  
Transitivity 0.1695

name degree\_centrality betweenness\_centrality \  
3 George Fox 0.231579 0.366149   
4 William Penn 0.189474 0.371038   
34 James Nayler 0.168421 0.161498   
16 Margaret Fell 0.136842 0.187174   
2 George Whitehead 0.136842 0.195294   
.. ... ... ...   
39 Ellis Hookes 0.010526 0.000000   
43 William Mucklow 0.010526 0.000000   
46 John Whitehead 0.010526 0.000000   
55 Dorcas Erbery 0.010526 0.000000   
95 Lewis Morris 0.010526 0.000000   
  
 eigenvector\_centrality   
3 0.449175   
4 0.270317   
34 0.335301   
16 0.253171   
2 0.249744   
.. ...   
39 0.061067   
43 0.061067   
46 0.033954   
55 0.045585   
95 0.002182   
  
[96 rows x 4 columns]

The code above reflects the current approach that *Network Navigator* takes to calculating metrics. The site uses JSNetworkX, a JavaScript library that ports many features of NetworkX, a Python library for network analysis first built by Aric Hagberg, Dan Schult, and Pieter Swart and released in 2005, that has since become one of the main coding libraries for working networks (*Hagberg, Swart, Chult 2008*). JSNetworkX was built by Felix Kling from 2012 to 2015, and though Kling is no longer maintaining the project, we still decided to use his code for a few reasons. First, though network analysis is a very popular computational methodology, most projects use programming languages like Python or R to compute network metrics, rather than doing so in the browser. Consequently, there remain few alternatives to JSNetworkX for computing these calculations for projects that do not have any backend databases like ours. To limit the potential issues of using unmaintained code, we copied the minified version of JSNetworkX library, and loaded it directly from our project, rather than relying on a Content Delivery Network (CDN) version (which is how most web applications utilize external code libraries). Our second reason for utilizing JSNetworkX is primarily pedagogical. We believe keeping the network functionality consistent across programming languages and DH tools is crucial for enabling both users experienced with network analysis to quickly understand the tool’s functionality (for example, instructors in a DH course) and for helping those new to network analysis to learn about these methods in a way that is transferable (for example, should they decide to eventually learn Python and NetworkX) and consistent with scholarly practices, since the algorithms in NetworkX are widely used in DH and computational research.

START HERMENEUTICS

For more information on NetworkX, see <https://networkx.org/documentation/stable/developer/about_us.html>, and for more information on JSNetworkX, see <https://felix-kling.de/jsnetworkx/> .

END HERMENEUTICS

Using these established node-level metrics, with a familiar library, was part of the initial commit of the site’s code, the very first time the tool appeared on the web:

display(Image("media/nn\_first\_commit.png"))



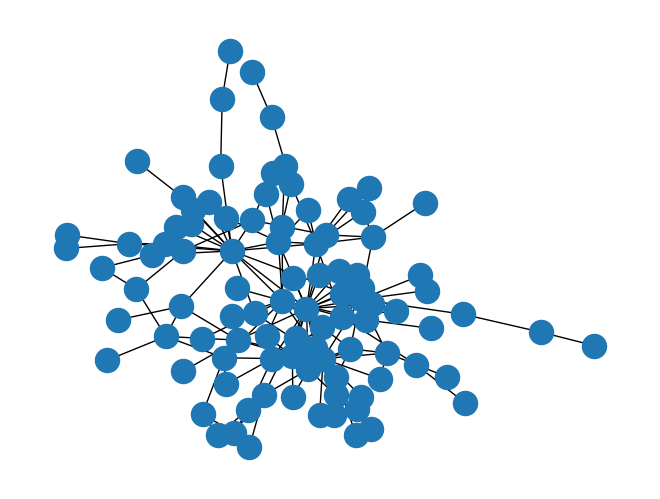
Early in the redesign, this sortable Bootstrap table was replaced with the DataTables library to enable easier search, sorting, and especially export. With this change users could explore these metrics outside of the tool, adding their own custom visualizations and pursuing new research questions as needed. Making it easy to get data both in and out of the tool was a central principle of the redesign. Since this wouldn’t be a single-tool approach to network analysis, we wanted users to be able to easily move output from one tool to another. DataTables also enabled a similar logic internally: we were able to create visualizations that would respond to users interacting with the table. If a user searched or filtered the table, the visualization would be filtered accordingly, reinforcing that visualization and metrics exist in conversation with one another. During the *Network Analysis+Digital Art History* workshop in Summer 2019, just one week after Ladd and LeBlanc first discussed redesigning *Network Navigator*, members of the *Palladio* team saw the 1.0 version of *Network Navigator* and proposed that the tool’s metrics be incorporated inside *Palladio*. While working on the *Network Navigator* redesign, Ladd helped adapt the *Palladio* code to include a metrics panel, still available in that tool today. To say that *Network Navigator* is part of a community of network tools is more than metaphoric, since parts of it are now literally embedded in other humanities network tools.

The partnership with *Palladio* throws another portion of the redesign process into relief: our choice not to create functionality for bipartite networks, or networks with more than one type of node. Bipartite network functionality is mentioned in that first feature wishlist screenshotted above: bipartite networks are very common in arts and humanities research, as these areas often deal with connections between people and artistic or historical objects (and thus two distinct node types). As the authors of *The Network Turn* point out, creating network data from humanities sources is not always straightforward, but modeling relationships as multipartite networks can be a useful way to proceed (*Ahnert, Ahnert, Coleman, Weingart 2020*). Because it is built for different kinds of humanities data, *Palladio* also allows users to visualize bipartite networks through prompting the user to select two different columns of data to link (though their actual network graph does not easily distinguish bipartite nodes whether through color or placement). While this choice makes perfect sense for *Palladio*, it was an ongoing question whether bipartite networks suited the pedagogical aims of *Network Navigator*. For the redesign, we had initially hoped to introduce users first to the simplest kinds of unipartite networks and increase complexity from there through including some bipartite functionality. However, we quickly realized that enabling bipartite networks would require an in-depth reworking of almost every part of the tool, and some of our libraries, including JSNetworkX, had not yet included bipartite network metric into their APIs. While we could allow users to visualize bipartite networks, without the ability to run algorithms for bipartite networks, any metrics we would present would ultimately be statistically misleading.

Thus, we decided not to include bipartite networks for three distinct reasons. First, bipartite options might distract or confuse the very researchers that were hoping to welcome into the field of network analysis, especially deciding how to represent a bipartite network as a unipartite one is an important choice that should be informed by the particular research question. Second, the tools and libraries we were working with were not yet ready to be used for bipartite networks, and given our time constraints as researchers and not full-time developers, we were hesitant to create custom Javascript libraries for bipartite networks that would require significant maintenance. And our third and final reason was that other tools, especially *Palladio*, were already providing researchers with a useful interface to visualize bipartite networks, allowing them to select source and target nodes, and we did not want to enable visualizations that would not correspond to our metrics. Indeed, many existing network analysis tools allow users to important bipartite graphs and run network analysis algorithms, but do not explain in-depth how these algorithms presume unipartite structures, leading to either confusion or misinterpretation of these metrics. We have come to recognize this decision as a pivotal one for the project, and it has illustrated for us the importance of focus and the careful consideration of what not to do. This decision was confirmed when, since building *Network Navigator*, other useful tools for creating unipartite networks from bipartite networks like *Table2Net* have emerged and joined the community of humanities network analysis applications (*Jacomy 2023*) .

The same kinds of design trade-offs and decision making drove the massive changes we made to visualization in the 2.0 version of *Network Navigator*. A goal of the 1.0 version was to subordinate visualization to metrics, though this needed serious revision in the site’s second version. Seeing more emphasis on visualization than metrics in many tools for beginners (*Gephi*, *Palladio*, *Cytoscape*, etc.), the site sought to make the opposite argument. Instead of creating visualization first, the site implicitly argued that the best way to first get acquainted with a network was through its metrics. To accomplish this, *Network Navigator 1.0* had only a small visualization that wasn’t very customizable, it used the built-in visualizer that came with JSNetworkX. The result was a very simple visualization like the one you see below:

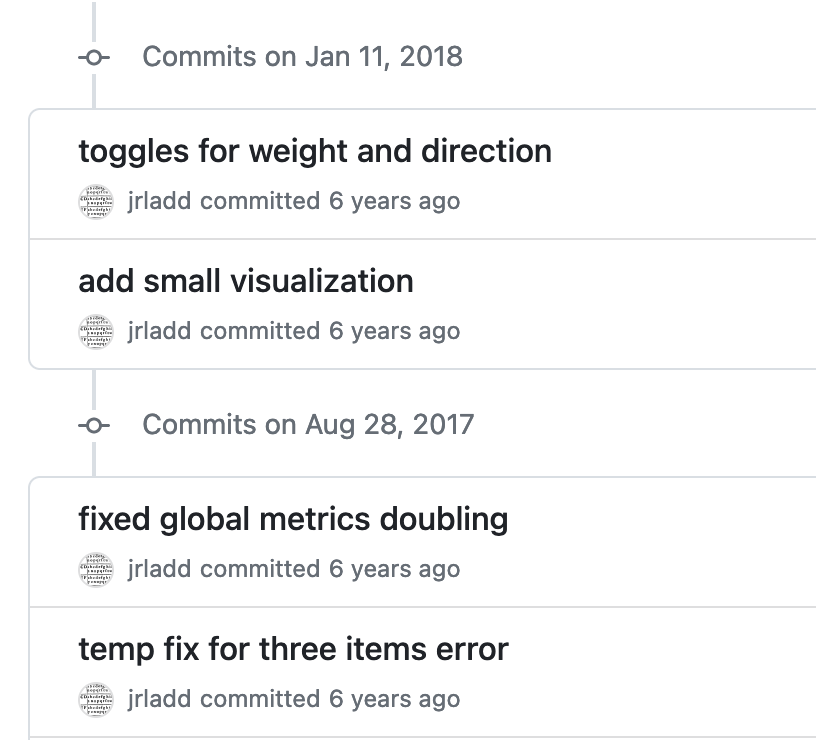
#START HERMENEUTICS  
  
nx.draw(G)  
  
#END HERMENEUTICS



Users could not update node or edge color, node size, position, or any of the typical customizations to which users of network tools had become accustomed. All of this was with the intention of steering the user away from treating a network as a primarily visual tool and toward the computational affordances of networks. Already thinking about the tool as infrastructure, Ladd began here what would become a theme of considering *Network Navigator*’s place within a larger ecosystem of humanities network tools. If other tools did visualization well but lacked detail and refinement for metrics, then *Network Navigator* could and should explore metrics more fully.

This is made evident by the fact that even the visualization, called the “small visualization” in the Github commit, wasn’t added until January of the following year, after Ladd’s work on the custom visualizations in *Six Degrees of Francis Bacon* was complete. The site had been live for six months making metrics available, but even then it was clear that metrics alone wouldn’t be enough for *Network Navigator* to serve as a piece of critical infrastructure.

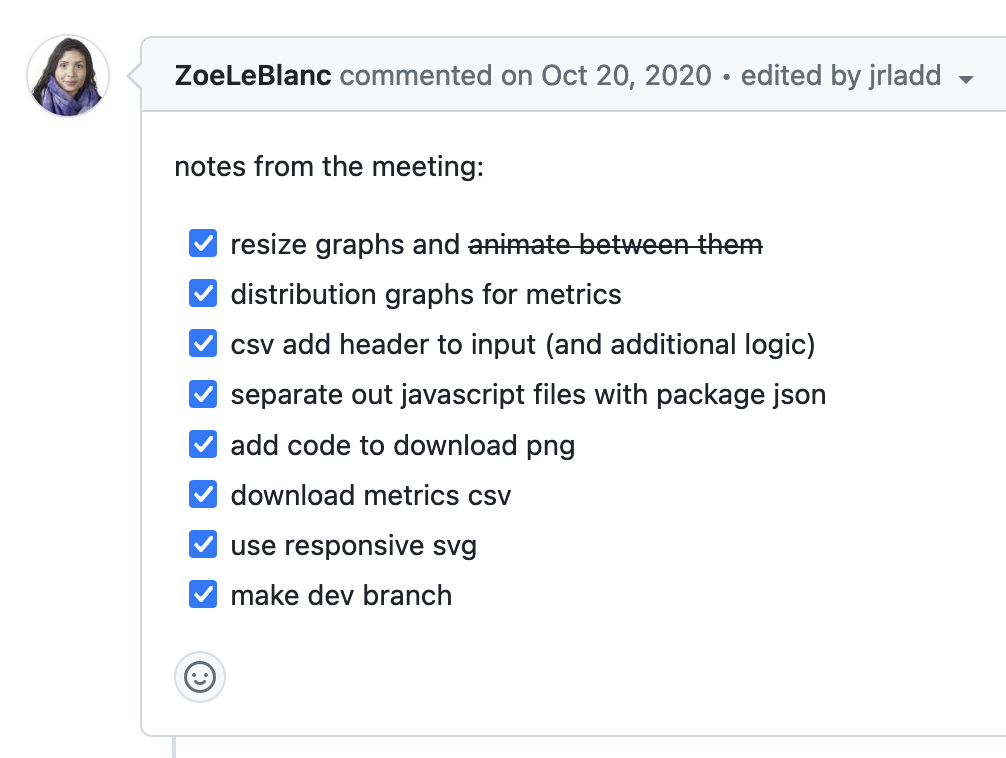
display(Image("media/nn\_commit\_list.png"))



By showing this small visualization, Ladd initially acknowledged that metrics alone weren’t sufficient for understanding a network, but in early redesign conversation our sense of what helps scholars understand networks evolved considerably. It became clear that many of the metrics, like density, were made much clearer when accompanied by even the simple visualization that the first version of the tool provided and that a more detailed visualization would be a boon to researchers. In becoming a part of the critical DH infrastructure, *Network Navigator* was in conversation with the tools around it, and in learning from our work with *Palladio* and other tools, it became clear that a new interface that could provide both metrics and visualization on equal footing would be the best option. *Network Navigator* could do more to guide researchers toward best practices around network visualization; it was becoming middleware, an essential learning point between general knowledge of network concepts and the ability to build complex models with more advanced tools. As such it was reaching a group of users with more needs for visualizations than the 1.0 site could provide.

As the authors of the original *Programming Historian* piece argue, when it comes to visualization and metrics in network analysis, it is impossible to have one without the other (*Ladd, Otis, Warren, Weingart 2017*). The same is true in other forms of exploratory data analysis as well—good visualization lets you see aspects of the data that are difficult to understand fully with statistics alone, and vice versa. *Network Navigator* has tried to embrace exploratory data analysis (EDA), which as Taylor Arnold and Lauren Tilton argue, “undergirds quantitative work in the digital humanities, particularly text analysis, and yet is rarely explicitly named as a key method or identified as a powerful approach in the field” (*Arnold, Tilton 2019*, 293). De-emphasizing visualization in *Network Navigator* 1.0 brought metrics to the foreground, but it made it difficult for users to fully engage in EDA and to understand their data. The central idea of *Network Navigator 2.0*, then, was to put visualization and metrics on equal footing and to show as much as possible how each relied on the other. Most of the items in the “Feature Wishlist” that began this section deal with creating a more flexible visualization structure that would respond to and enhance what *Network Navigator* was trying to show about metrics. This can be seen in more detail in the notes from our October 20th meeting, in which we explored a number of technical ways to integrate visualizations and metrics. The list below thinks about the responsive sizes of visualizations for different screen and panel sizes, downloading both metrics and visualizations, and adding histogram plots to the metrics panel that can adjust and reflect what is shown in the visualization panel.

display(Image("media/nn\_october\_meeting.png"))



Our desire to create a more robust set of visualizations for *Network Navigator* coincided with a number of interface principles and developments from our discussion of minimal computing in the previous section. Since the development of tools like *Gephi*, browser capabilities for network visualization have advanced to the point where a browser-based tool can do as much as a desktop application. This is primarily thanks to the D3.js library and, more recently, WebGL, which have greatly expanded both the possibilities for manipulating data and handling larger quantities of data within web browsers. The use of D3 was a major part of several humanities network analysis projects, including *Palladio* and the redesign of *Six Degrees of Francis Bacon*, and the visualization features of *Network Navigator* take advantage of D3's interactive capabilities. D3, which stands for data-driven documents, is a tool for attaching data to visual objects in the browser using the SVG or Canvas web standards (<https://d3js.org>). At the time it was one of the few tools that could handle the level of visual flexibility that we wanted for *Network Navigator*, but it is limited by the amount of nodes and edges it can display at one time. Often humanities networks are larger than what the browser can display using SVG, and this is why humanities tools have come up with novel ways of dividing large networks, such as SDFB’s method of displaying only 2-degree ego networks and never the full graph. A newer web standard, WebGL (Web Graphics Library), has made it possible to display hundreds or thousands more points in the browser while maintaining the interactivity that D3 provides (<https://www.khronos.org/api/webgl>). We discussed the trade-offs between WebGL and D3 early in the design process, and because WebGL tools for creating networks were still brand new, D3 was still the best option. However, as the creators of D3 themselves point out, “D3 makes things possible, not necessarily easy; even simple things that should be easy are often not” (<https://d3js.org/what-is-d3#d3-is-for-bespoke-visualization>). We wound up writing much more JavaScript code than we would need to today to produce the same visualizations. Only just being refined at the time of our redesign, higher-level libraries Graphology and Sigma.js, built on top of WebGL by the SciencesPo médialab, allow very large networks to be displayed in the browser with less code (<https://graphology.github.io/>, <https://www.sigmajs.org/>). These libraries power the new *Gephi Lite* web tool, released by *Gephi* in 2022, which allows large force-directed network visualizations to be made in the browser. A similar approach could be taken by *Network Navigator* in the future to expand the size of networks that it can accept. But unlike *Network Navigator*, while *Gephi Lite* allows the user to calculate network metrics in order to customize the visualization, it doesn’t display or allow export of those metrics. In the three years since *Network Navigator 2.0* was developed, browser capabilities for network visualization have only increased, and by using D3 to display networks in a range of layouts, *Network Navigator* was at the forefront of these changes.

We set about to take advantage of all the customization and interactivity that D3 would offer: *Network Navigator* could offer easy switching between a variety of network layouts, it could allow the user to filter based on metadata of their choice, and it could let researchers customize the color, size, and shape of their visualizations. But we chose to do this in such a way that visualization would be integrated with and equal to (but never superseding) what the site was already featuring in its metrics. This resulted in an interface which allows the user to easily move back and forth between the two views or see both at once. Other network analysis tools offer two views, but often on completely separate panels or pages---by offering them side by side we emphasized that their equal importance and kept the site to a minimal number of pages, with fewer interfaces for a new user to learn. That is to say, the *Network Navigator* interface is the result of careful consideration of the rhetorical force of its organizational choices. What does it mean that so many network tools have metrics on a separate page or in a pop-up window? What does it mean to have those metrics appear on the page by default? The overall goal is not just to create something easy to use for people who already understand network analysis, but to provide guidance to researchers who are just starting out about what network analysis can accomplish visually and computationally. What follows is an in-depth description of the visualization features of the site and how they are integrated into *Network Navigator*.

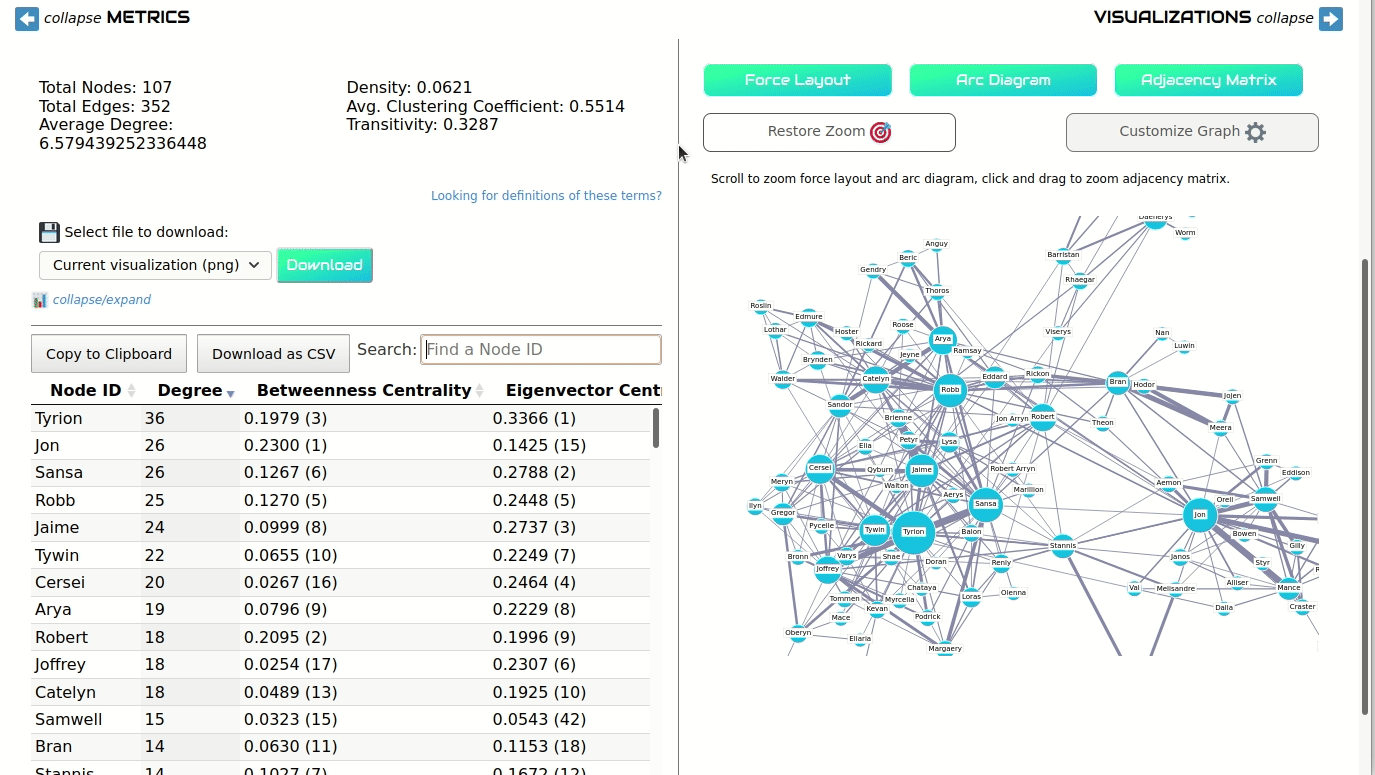
START HERMENEUTICS

It’s notable that all of these libraries were made by academic researchers or with those researchers in mind: Mike Bostock worked on D3 with Jeffrey Heer and Vadim Ogievetsky of the Stanford Vis Group, which has since become the University of Washington’s Interactive Data Lab (<http://vis.stanford.edu/papers/d3>), and Graphology and Sigma.js were developed by network researchers at SciencesPo médialab (<https://medialab.sciencespo.fr/en/tools/>).

The page layout in *Network Navigator* was further complicated by the need for a mobile interface with collapsing columns. This was another point of detailed discussion during the development process. When a user visits the page on a smartphone or table, there may not be enough room to display visualizations and metrics side-by-side, but in this case the columns “collapse” to appear one on top of the other. Even in the mobile interface, both metrics and visualizations are available on the first page of results.

END HERMENEUTICS

display(Image("media/select\_viz.gif"))



The image above shows the current *Network Navigator* interface with the *Game of Thrones* network loaded. This dataset is part of Melanie Walsh's curated social network datasets and was first collected by A. Beveridge and J. Shan in their "Network of Thrones" article for *Math Horizons Magazine* (available for download in this GitHub repository <https://github.com/melaniewalsh/sample-social-network-datasets>). Because of both its size and its similarity to other co-occurrence networks derived from fiction, which are common in humanities network analysis, it was frequently used for internally testing the *Network Navigator* visualization tools.

Three prominent buttons at the top of the visualization panel let the user switch between "Force Layout," "Arc Diagram," and "Adjacency Matrix" visualization layouts. Network analysis tools of the 2010s privileged force-directed layouts that attempt to organize nodes into communities. This led to a boom in humanities network visualization of so-called “hairball” layouts—attempts at force-directed display of a network that wind up showing an unreadable jumble of overlapping nodes and edges. *Network Navigator* provides a Force Layout option that creates a force-directed node-link diagram, and the implementation for this layout comes from D3 and is similar to Gephi's ForceAtlas and ForceAtlas2 layouts. For many, this visualization type is synonymous with network, but it was important to us that *Network Navigator* be as clear as possible that this was one of many visualization options. A newcomer to network analysis may not even realize that other visualization styles are possible or, in many cases, preferable. The interface choice of large layout buttons reinforced the pedagogical point of the site’s new Visualization panel: if one layout of your network isn’t showing anything useful, there’s no reason to continue with it. Instead, try a new layout that emphasizes something different. No other network analysis tool was offering multiple visualization types in this way, though projects like SDFB had piloted this approach. *Palladio* displays only a force-directed node-link diagram. *Gephi* presents a number of layout algorithm options, but most of these are for different variations of node-link diagrams, and none of these tools provides a matrix visualization. To provide the user with more options, we settled on three layouts—rather than overwhelming users with too many layout choices, we wanted to give a few key options that would underscore the main point and prepare the user for more detailed visualization tools.

This prominent placement is meant to reinforce the idea that there is no single layout that is well suited to all networks and all research questions. The Force Layout is good at showing communities and structural holes, but it obscures rankings and can sometimes become so dense that the graph is difficult to read. It works well for the *Game of Thrones* network, but with larger networks this layout often produces the dreaded "hairball" visualization. It also prompts particular kinds of questions about community and proximity in networks that may not be a researcher's main interest or reason for creating the network. The other two types of visualizations we provide are far less common in network visualization, and by putting them on equal footing with the force layout our hope is to encourage researchers to embrace non-force-directed visualization approaches.

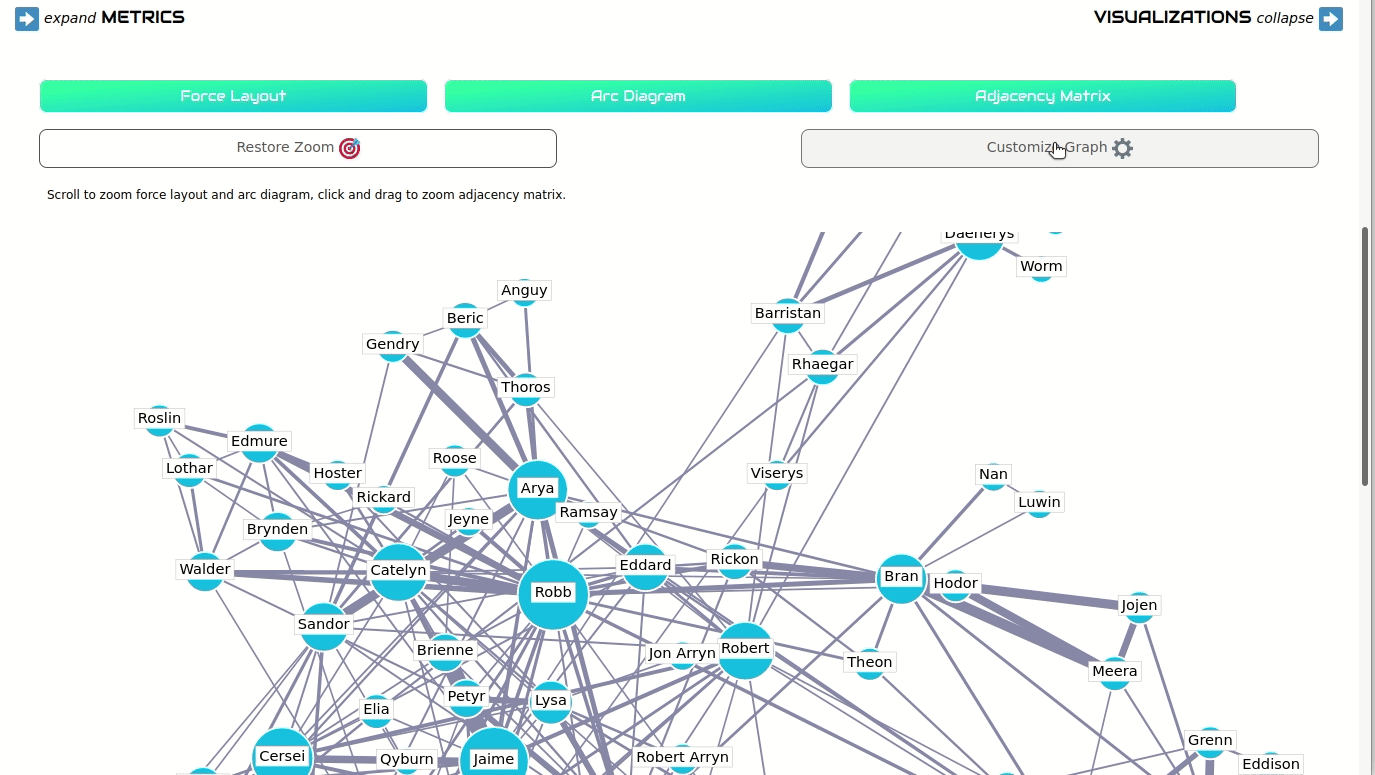
The Arc Diagram is a rectilinear layout—it arranges nodes in a line, and it displays edges as arcs so that they are visible above or next to the line. Our version of the Arc Diagram is available in both horizontal and vertical arrangements, depending on the preference of the user. Rectilinear layouts are most useful when the order of nodes is important. The ordering could help to show change over time, to sort nodes by network metrics, or to make particular nodes and subgroups easier to find. The Arc Diagram was meant to emphasize that in a node-link diagram the nodes can be positioned in any way that is helpful for the researcher or viewer and need not follow a force-directed pattern.

By contrast, the final visualization layout, the Adjacency Matrix, shows that networks can be visualized without node-link diagrams at all. An adjacency matrix visualizes nodes in rows and columns according to the mathematical representation of a graph that represents the presence or absence of an edge as a table full of 1s and 0s. In an adjacency matrix visualization of a unipartite network (one with just one type of nodes) the rows and columns both represent all the nodes in the network. In the cells where those nodes intersect, a color is displayed when a connection is present. Instead of representing nodes as circles and edges as lines between them, the adjacency matrix represents nodes as rows and columns and edges as color-filled cells. When the weight of the connection is greater, the shading in the cell is darker. Node-link diagrams like the Force Layout and Arc Diagram become very hard to read when a graph is very dense: there are too many edges to be visualized cleanly and the graph winds up a jumble of lines in all directions. Adjacency matrices automatically display a dedicated cell for every *possible* edge in the network, and therefore the visualization doesn't get any more visually complex as the graph gets denser. In a very dense network, there would simply be more colorful cells than blank ones in the adjacency matrix representation. Furthermore, we also hope that the Adjacency Matrix will encourage users to understand that networks can be conceived of and operationalized as matrices. Too often network analysis is treated as completely distinct from other methods of data analysis, whether clustering or dimensonality reduction algorithms, so we intended this visualization to also help users see these connections, especially as they continue to learn computational methods for representing and analyzing their data.

In providing three visually distinct forms of visualization and making them very easy to switch between, *Network Navigator 2.0* aims to instruct its users, by showing rather than telling, that network visualization can take many different possible forms and that even within this interface's relatively small set of visual possibilities there's still a huge amount of control the researcher has over customization and the look and ultimately the visual rhetoric of their graph. And we made sure that all of this could happen within the browser, taking advantage of then-recent advances in Javascript and browser capabilities to provide functions that would previously have required installing a desktop application with a much heavier footprint. This makes a big difference for the classroom as well: rather than needing to devote time to installation and troubleshooting, an instructor can more immediately begin working with students on the network itself. The minimal computing approach is in this case not only the most equitable and matches most closely with our theoretical aims for the project, it is also the most practical approach. And the same browser technologies that make it fast to create the network make it equally quick to edit and alter the network’s structure.

After getting the visualization itself working the way we wanted through different layouts, we were able to spend time and attention on allowing users to customize that visualization. We did this knowing that our features for visualizing would never be as detailed as other tools in the ecosystem, but we felt there should be a tool that provided a first step, a glimpse of what was possible, with sensible guardrails. A more limited set of editing tools would allow a researcher to get a feel for adjusting their network without having to learn a complex interface or getting in so deep that it was impossible to start again or go back to a previous version. That is, by staying simple, *Network Navigator* incorporates both “reversibility and reproducibility” (*van Geenen 2020*).

display(Image("media/attribute\_filter.gif"))



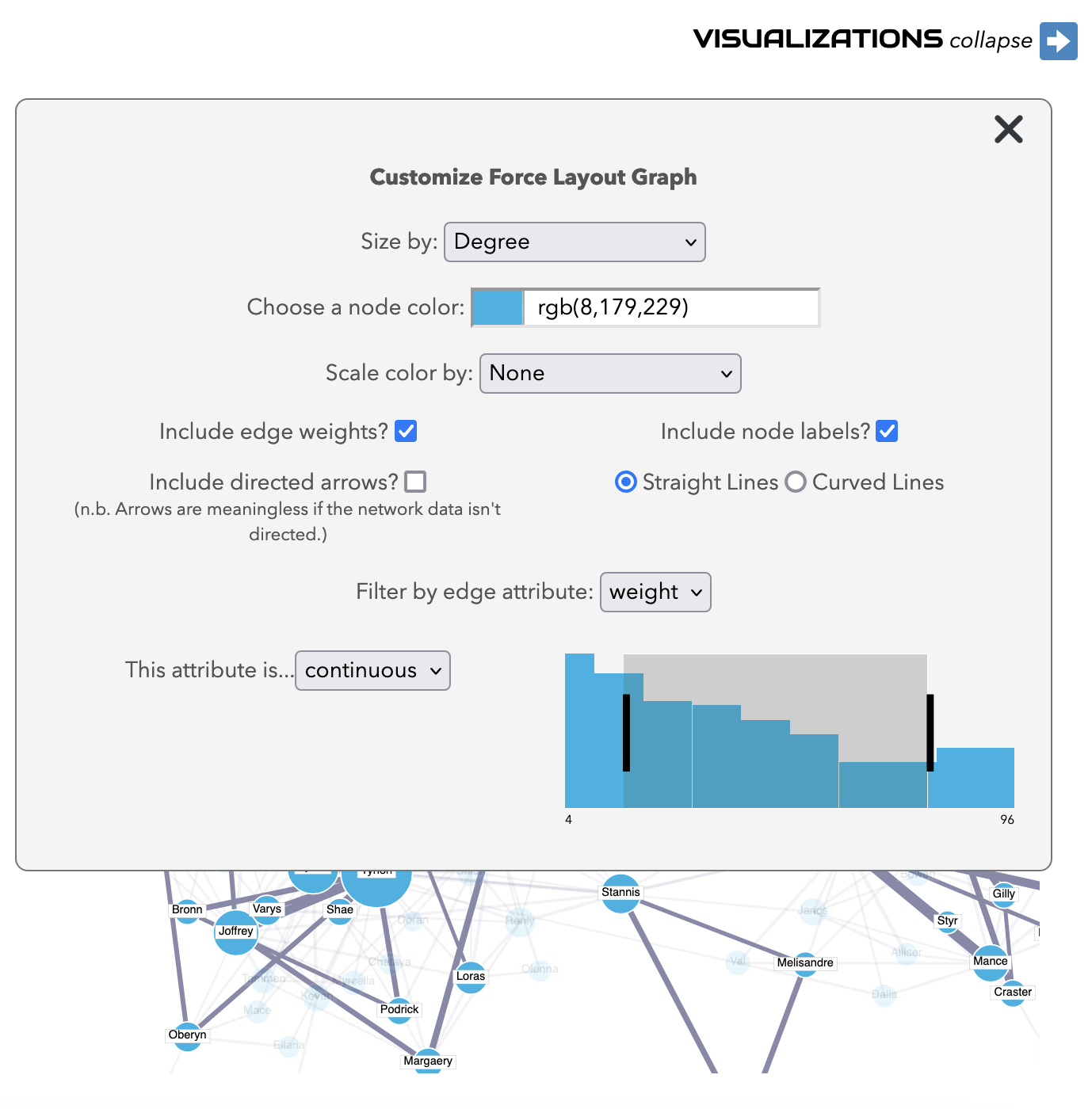
The image above shows one of the main ways that researchers can control their visualization and adapt it to particular research questions. Many humanities network applications involve change over time, a property with which traditional network visualizations struggle. One of the advantages of working with dynamic JavaScript visualizations is that the networks can be adjusted and updated based on user input. Given attributes on an edge, it's relatively simple in D3 to let the user control the graph based on those attributes.

This is how *Six Degrees of Francis Bacon* approached the problem of change over time as well. The site lets users visualize a subset of the network based on a beginning and ending year, and they can do the same thing to filter the network based on its algorithmic "confidence" measure. *Network Navigator* does something very similar, but instead of using just a couple pre-selected edge attributes, the site will let you filter based on *any* edge attribute. Users can easily create a time filter, but also a filter for edge weights, type of relationship, or any additional data that they’ve recorded about their edges.

In the "Customize Graph" panel, you can select an attribute to use as a filter from a dropdown menu. After specifying whether this attribute is a continuous (numerical) variable or a categorical one, you can select a subset of that attribute to filter. If the attribute is categorical, for example a variable that shows a type of relationship (familial, professional, friendship, etc.), the filter will be displayed as a series of radio buttons, allowing the user to select the category they'd like to display. If the attribute is numerical, the filter will be displayed as a histogram showing that attribute's distribution through the graph. The user can select a shaded bounding box around parts of the histogram to filter a specific range of that variable (as you see above).

In network visualization there is often an impulse to show the entire network at all times, but this impulse frequently runs counter to the research questions being asked, which typically are interested in specific parts or segments of the overall network. The site's easy-to-use filtering tools to emphasize that visual simplicity is often the best way to make an argument in and through network visualization.

display(Image("media/customize.png"))



The customize window does much more than just filter the graph. Visualization is full of critical choices about how to display aspects of the network. Our goal with the tool is not only to be critical makers ourselves, but to train the user to be a responsible critic of their own work and the work of others, to consider and test multiple possibilities for metrics and visualization. In response to this, the site tries to make a limited number of choices available to users of the web interface. While there aren't as many customization options available as in a tool like *Gephi*, the goal was to allow a reasonable amount of the most common visualization adjustments in order to highlight for users how those visual choices alter the arguments they are able to make about their networks.

A key example of this is in the size and color of nodes. Node size and sequential color scaling can both be based on the different degree measurements that *Network Navigator* calculates. This allows a user to make node size reflect degree and node color reflect betweenness, effectively highlighting a network's pivotal nodes. And node attribute customizations are meant to reflect the relationship between metrics and visualizations. User interactions in the metrics panel, for instance changing the histogram from degree to eigenvector centrality, will also update the network visualization. In this way, a user can learn how metrics and visualizations are intertwined and the ways in which visual choices rely on a strong understanding of the mathematics of network analysis. The initial purpose of the tool, to foreground metrics and fill a niche within the larger infrastructure of tools for networks in DH, stayed with *Network Navigator* all the way through this later stage of development. By making adjustments on the metrics side of the site, the visualization is changed commensurably.

Other visual customizations reflect similar relationships. For instance, when a researcher labels their network as Directed when entering data, the visualization automatically includes arrows showing the direction of an edge. Similarly, edge thickness will be mapped to weight when a Weighted network is selected. This reinforces the idea that high level choices about the *kind* of network a researcher is working with will lead to certain conventional visualization choices. But all of these choices can also be directly altered by the researcher: arrows and edge weights are as easy to turn off as they are to make visible, albeit with caveats about *why* a user may want to add these visual elements. The goal is not to force conventional visualizations on the researcher but instead to give the researcher a set of options and equip them with the information to make good choices. For this same reason, curved edges are automatically turned on for the arc diagram to increase readability but never displayed by default for the force layout. Curved edges are sometimes used to indicate direction, to add visual style, or both. To reduce any confusion between these two uses, this customization is left entirely up to the researcher and never automatically triggered by other choices.

Ultimately the goal is to give the researcher tools for creating an effective and accurate visualization that reflects the properties of their network. By starting with sensible choices but allowing the researcher a great deal of control over the final look of the graph, *Network Navigator* is able to provide guidance to researchers without leaving them feeling overwhelmed by options. A critical approach to infrastructure requires us to take a middle course: to find a place for *Network Navigator* that neither oversimplifies for the beginner nor caters to an imagined “power user” with a foreknowledge of all of the functions they’ll need. By never trying to be a “Killer App,” the site can find a niche within a larger ecosystem and serve a purpose without having to serve *every* purpose.

## Communities: Navigating the Future of DH Tools and Next Steps for *Network Navigator*

This essay arose out of our conviction that the individual choices made about *Network Navigator* detailed above are deeply entwined with the tool’s intervention into existing scholarship and our own experience as DH practitioners. As Julia Flanders puts it in her recent essay “Jobs, Roles, and Tools in the Digital Humanities”: “We can start to see here that there is a reason why digital humanists are so preoccupied with tools, and it is not because tools are important in themselves: it is because they are an irritant. They catalyse something complex and difficult concerning professional identity, scholarly methods and practices, and specific types of expertise” (*Flanders 2023*, 156). Completing the redesign of *Network Navigator* was only half of the work. The other half is this piece, which articulates that particular “something complex and difficult” in which our scholarly practices and technical expertise shape both our professional identities and the uses of the tool. This is why there was equal focus on the components of the tool we included—global and local network metrics, responsive visualizations, flexible interface design—as well as those components we specifically and critically chose to omit—bipartite visualization, undo features, user sign-on, and, overall, our decision to avoid making our tool a "Killer App" for network analysis. In software development, these types of considerations are often described as "edge cases", but in DH tools, we believe that these concerns need to be brought from the periphery to the center of the design process.

The history and critical discourse around DH tools, as well as the specific examples of network analysis tools for both humanities and non-humanities researchers, provides a solid foundation for understanding the ways that DH tools intersect with research and teaching. But we still need more examples that delve into the specifics of how a DH tool was made, the motivations of who made it, and the ways that a tool fits into its larger ecosystem and scholarly context. We hope that this essay can be an example for how to articulate the intersection of these aims. Flanders continues, “The questions of ‘who is digital humanities?’ and ‘how is digital humanities?’ can evidently be reframed as questions like ‘what are the design goals of our tools?’, ‘when should a tool “just work”?’, and ‘are we tool users, tool builders, or tool theorists?’” (*Flanders 2023*, 156). We believe our close examination of *Network Navigator* is exemplar for how to answer these reframed questions. Our tool was designed for first-time network researchers and students to get acquainted with both the visual and measurable aspects of network analysis. To accomplish this, it had to “just work” in the browser for a wide variety of users and data, but with enough friction to require users to think carefully about the ways that networks would serve their individual research and teaching aims. Finally, through this experience we have come to identify ourselves as occupying all three of Flanders’s categories: we are tool users, tool builders, and tool theorists when we engage in the complex knot of decision making that results in a tool like *Network Navigator*.

And this triple scholarly activity is felt as we participate in the larger community of researchers that use our and others’ network analysis tools. *Network Navigator* isn’t finished and hopefully will never be fully complete. Our work as tool users inspired us to become tool builders, and we hope that *Network Navigator* can inspire others in the same way. The tool’s current users have already helped us to think about what’s next for the site: perhaps with today’s web technologies an easy form of bipartite network analysis will be possible someday, and maybe the software libraries used by tools like *Gephi Lite* might soon enhance *Network Navigator*’s technical capabilities. Any tool that exists within such a robust community will not remain static. Our aim here has been to describe that community and ecosystem as much as the tool itself, to lay the groundwork for future building, use, and theorizing of this and other network analysis and DH tools.