Journal of the Midwest Association for Information Systems

Volume 2024 | Issue 2 Article 3

Date: 07-02-2024

R Code Authorship Attribution using the ASAP Tool

Austin Coursey
Vanderbilt University, austin.c.coursey@vanderbilt.edu

Matthew F. Tennyson
Murray State University, mtennyson@murraystate.edu

Vlad Krotov
Murray State University, vkrotov@murraystate.edu

Abstract

Source code authorship attribution is the task of determining the author of a program. Code authorship attribution has many useful applications, such as plagiarism detection and settling copyright infringement disputes. With the rise in popularity of the R programming language in the Data Science community, the need for source code authorship attribution of R programs has also risen. In this research note, we propose and evaluate the use of a tool called "ASAP: A Source-code Authorship Program" for attributing authorship of R code. We run experiments on two different datasets of R code: a "clean" one (where we are sure of each program's author), and an "unclean" one (with more realistic data, were authorship of some code files is not certain). We find that in both datasets running an experiment using the ASAP tool with the Source Code Author Profile (SCAP) algorithm on R programs attributes authorship successfully. A number of implications for both academics and practitioners are formulated based on these experiments, together with directions for future research in this area.

Keywords: Authorship attribution, SCAP, ASAP, R programming, plagiarism detection, copyright infringement

DOI: 10.17705/3jmwa.000090

1. Introduction

R has become one of the most widely used languages for data analysis in many industries and academic fields (Lander 2014; Schutt and O'Neil, 2014). Academic researchers use R and related tools for collecting, organizing, and analyzing both quantitative and qualitative data (Krotov and Tennyson, 2018). Instructors use R as a tool for equipping students with hands-on knowledge of the research process, statistics, and data wrangling (Fawcett, 2018). Many educational institutions offering Data Science or Business Analytics programs have at least a portion of their curriculum devoted to R. Industry researchers use R for addressing important operational and strategic questions for their organizations in such industries as banking, marketing, pharmaceuticals, public services, etc. (Lander, 2014). In addition, there is a growing practice of building data products and publishing them on the Web with the help of the Shiny add-on to the RStudio Integrated Development Environment (RStudio, 2021a).

Today, there is a vibrant and growing community of R developers coming from all kinds of backgrounds. With R being an "open source" language, there is wide-spread practice of sharing R code freely among the members of the community through such online platforms as GitHub, Stack Overflow, and many others. This code sharing practice creates a problem of proper authorship attribution in the R community. This problem, as explained further in this paper, is driven by two overlapping issues: intellectual property and intellectual honesty. While an open source tool, R can be used by private, for-profit companies to develop proprietary software products. These products constitute intellectual property and are protected by copyright law. However, given the open source nature of R and many R-based projects, the issue of code authorship contribution is usually a problem of intellectual honesty and proper recognition of those who develop the original code. This paper is an attempt towards addressing the problem of proper code authorship attribution in the R community.

This paper explains the problem of authorship attribution of R code given the popularity of R in the Data Science community. The study utilizes a tool (ASAP: A Source-code Authorship Program) for attributing authorship of R code using the Source Code Author Profile (SCAP) algorithm. The primary research question being addressed is this: Can the authorship of R code be properly attributed using methods established for other programming languages? The primary goal is to assist researchers and practitioners in these endeavors by proposing and testing a method and a tool for R code authorship attribution. The tool is tested using large sets of real-world R language project files. These experiments confirm usefulness of the ASAP tool and SCAP algorithm for attributing authorship of R code in real-world scenarios involving students, researchers, and practitioners. Specific ways in which these tools can be used for enforcing intellectual honesty and resolving copyright infringement disputes are proposed. The study also provides directions for future research that can potentially improve accuracy of authorship attribution in the R community.

2. Factors Behind the Growing Popularity of R

Several factors can explain the growing popularity of R among researchers and practitioners. These factors include: the free, open source distribution of R and related tools; built-in functionalities for data analysis; simple and intuitive syntax that facilitates broader use of the language; a comprehensive collection of user-contributed R language extensions that facilitate virtually all known forms of data manipulation, analysis, visualization, and research communication; and presence of vibrant online communities of R developers who share their expertise in R and help others find solutions to R-related problems (R Project, 2021). Each of these factors behind R popularity is discussed in more detail below.

First, unlike some of the well-known proprietary tools for data analysis, such SPSS or SAS, the R language is free and distributed in open-source fashion, in accordance with the terms of the Free Software Foundation's GNU General Public Licence (R Project, 2021). Even some very powerful add-ons for R, such as the RStudio Integrated Development Environment, can be used in an open-source fashion (RStudio, 2021b).

Second, unlike other popular programming languages used in data analysis (e.g., Python, C++, or Java), R was built specifically for data analysis. Just like any programming language, R syntax includes variables, data types, functions for input-output, loops, conditionals, user-defined functions, etc. But on the top of these typical features of a programming

language, R contains built-in facilities for retrieving, cleaning, and storing data; a collection of operators for performing mathematical operations using arrays and matrices; built-in tools for data visualization and statistical analysis; and many other features useful for data analysis (R Project, 2021).

Third, the R language has a simple and intuitive syntax in comparison to other popular and well-established programming languages such as C++ (R Project, 2021). This makes R appealing to users interested in data analysis who, nevertheless, are not professional programmers and may lack a deep understanding of some fundamental aspects of programming, such as the Object-Oriented Paradigm (OOP). While the OOP is implemented in R (Wickham, 2014), most R applications are developed in a script-like, procedural fashion to automate some or all aspects of data retrieval, organization, or analysis (Krotov and Tennyson, 2018). This appeals to statisticians and academic researchers, many of whom can write code but are not professional programmers. Still, R can be used in conjunction with other popular languages, such as Python or C++, to expand its functionality or speed up computations (R Project, 2021).

Fourth, R has a very large collection or R "packages" – user contributed extensions of the R language containing various functionalities for data analysis. For example, the Comprehensive R Archive Network (CRAN) online collection contains almost 17,000 R packages developed by the R user community. These packages contain a wide array of functionalities for processing and analyzing both quantitative and qualitative data – anything from simple data cleaning and manipulation tasks (e.g. "dplyr" package) to advanced machine learning algorithms (e.g. "superml" package). These packages also cover a wide variety of industries and academic fields: from financial analysis (e.g. "XBRL" package) to analysis of literary works performed by academics in the English literature field (e.g. "syuzhet" package). Of special importance are graphical packages (e.g., "ggplot2" package) that contain customizable graphics elements that allow users to create useful and aesthetical data visualizations of virtually all known forms (Lander, 2014). With all these tools or packages available, R can be used to automate and unambiguously describe all steps of a research project: from the time a dataset is loaded from the Web or desktop computer to the time documents and slides are produced to communicate the results. In fact, there is a package called "knitr" that assists in documenting and communicating all the steps and outcomes of a research project (Lander, 2014). These tools for communicating research in R can potentially increase its impact and enhance its reproducibility (Peng, 2011).

Another factor that is probably both a cause and an effect of the growing popularity of R is the presence and constant growth of vibrant online communities of R developers. For example, Stack Overflow, an online community where developers get their programming questions answers and share their solutions and experiences, contains close to half a million answered questions or discussion threads related to R. With these and countless other online communities and resources for R developers, one only has to type "how to do […] in R" to get R source code that does what the user wants in a matter of seconds. Oftentimes, researchers "cut and paste" R code into their own projects without putting much thought into proper attribution of authorship of the code they are using.

While R has all of the advantages outlined above, Python is another popular programming language used in data science. Both R and Python are widely-used and have their advantages and disadvantages. Both are free, relatively easy to learn, and have large reusable libraries. The most fundamental difference between the languages is that Python was designed as a general-purpose programming language, while R was specifically designed for data analytics. R has more packages and specifically more packages that help with data mining and statistical analysis. R can be used to quickly perform specific data analysis tasks and create visualizations, while Python is more suitable to building complete applications (Griffiths, 2022). Both languages are growing in popularity and have their place in the field of data science. In our study, we have chosen to focus on the R langu age.

3. The Problem of R Code Authorship Attribution

This increasing breadth and depth of R usage exacerbates the problem of R code authorship attribution. In this paper we define authorship attribution as the task of deciding who wrote a particular R source file (Zhao and Zobel, 2005). The problem of code authorship attribution, as defined here, is somewhat different and subtler than the problem of detecting plagiarism in R code – a situation where somebody does a "copy and paste" of R code written by somebody else and presents it as his or her own solution. The problem of code plagiarism has been addressed in the field of Computer Science with the help of such tools as JPlag (Prechelt et al., 2002) and MOSS. Moreover, the traditional plagiarism detection tools, such as Turnitin or SNITCH, although not designed specifically for code plagiarism detection, can detect "copy paste" code

as well (Niezgoda and Way, 2006). The problem of R code authorship contribution arises when someone reuses R code by modifying it for his or her own research project or proprietary data product. In this case, the code is not plagiarized (at least, not in a blatant fashion). But the problem of determining who is the author of this modified code still remains.

This problem of deciding on the true author of an R code is important for both academics and practitioners. While R development is often guided by the open-source practices (RStudio, 2021b), the problems of fairness to the original author of code and intellectual honesty still remain. Just because the code is open source, this does not mean that someone can take it and present it as his or her own. For example, most of the licenses under Creative Commons, an organization responsible for setting licensing level in the open-source community, require proper authorship attribution when open-source code is used (Creative Commons, 2021).

It is interesting that nowadays many academic journals encourage or even require the submission of supplementary resources (e.g., data sets, software code, etc.) together with the text of an article itself. But it is usually only the article that is checked for plagiarism and not the source code that was used to conduct a study that led to the article. With the availability of such R tools as "knitr" (a package that allows to "mesh" together R code, analysis output, and author text explaining the analysis and interpreting the findings), the boundary between an article and the code used to generate the article may become quite blurry.

Another aspect of the problem facing academics in relation to R code authorship involves students submitting a modified version of somebody else's R code for course credit. Opinions regarding the appropriateness of code reuse for classroom projects may differ among educators. Still, most academics, being exposed to the topics of plagiarism and intellectual honesty in their doctoral programs and via policies of the professional organizations that they belong to, would still prefer students to acknowledge that they are relying on somebody else's work for their classroom projects.

When it comes to proprietary R products and solutions, then the problem of R code authorship transcends the realm of ethics and becomes a legal issue. As most developers know, software constitutes intellectual property and can be copyrighted (Bainbridge, 1999). Intellectual property in the form of software code constitutes a very strategic intangible asset for modern companies (Lev, 2000). Improper use of copyrighted intellectual property, be it text, R code, or even data that that the text or R code is related to, constitutes copyright infringement – something that can lead to a lengthy and costly litigation in a court of law (Dreyer and Stockton, 2013).

Given the growing popularity of R in industry and academic research projects, both academics and practitioners need to become well-equipped for enforcing the highest standards of intellectual honesty and protecting their intellectual assets. The main goal of this paper is to assist researchers and practitioners in these endeavors by proposing and testing a method and a tool for R code authorship contribution.

4. Additional Literature Review

Determining authorship of natural language text is a problem that has been studied for years across many different applications. Stamatatos (2008) provided brief history of authorship attribution of natural language documents with a focus on modern computational methods. In addition to a review of the history of authorship attribution, an overview is given of stylometric features commonly used in natural language processing and an overview of approaches used, such as profile-based, probabilistic, compression, similarity-based, and hybrid models.

In the past year alone, several studies of authorship attribution of natural language documents have been published. Makhmutova et al. (2023) evaluated the importance of attention scores using character n-grams specifically within news articles written in English and Russian. Uchendu et al. (2023) provided a review of not only authorship attribution methods, but also authorship obfuscation methods, where "authorship obfuscation" refers to the task of modifying a document to hide its true authorship. Alqahtani and Dohler (2023) looked at the problem of authorship attribution specifically of Arabic documents, which is uniquely challenging due to the complexity of the Arabic-language morphology. Sebastián et al. (2023) presented a method for identifying the owner of a domain or website. Their method was shown to be much more accurate than the traditional approach of using the "WHOIS" protocol for looking up a website's registered owner. These recent publications provide just a sampling of the ongoing research and innovation within the field of natural language authorship

attribution.

Determining the authorship of programs and the related problem of detecting plagiarism of source code is a much newer area of study. Kalgutkar et al. (2019) provided a summary of code authorship methods and challenges. They specifically frame the problem around the challenge of identifying the source of malware as their main motivation. A plethora of studies have been published in regards to various aspects of source code authorship attribution. Methods have been proposed to attribute authors of programs written in traditional programming languages like Java and C++ and studies have measured the effectiveness of those methods (Frantzeskou et al., 2007; Burrows and Tahaghoghi, 2007; Ding and Samadzadeh, 2004; Tennyson, 2013). Tools have been created to detect plagiarism of source code (Prechelt et al., 2002), especially within the context of programming assignments in the classroom. Studies have looked at source code authorship attribution from a practical software engineering standpoint (Bogomolov et al., 2021). Petrik and Chuda (2021) studied how a programmer's style changes over time and how that affects source code authorship attribution. Dauber et al. (2018) looked at how to attribute authorship of small, incomplete code fragments. Gonzalez et al. (2018) proposed an approach to perform authorship attribution specifically on Android apps. Hendrikse (2017) investigated the possibility of attributing authorship of executable machine code, when the source code isn't available, with surprising levels of accuracy.

So, authorship attribution is a problem that has been studied in many different ways and in many different applications for many different languages. However, to our knowledge, it has not been studied specifically in R code. That being said, studies have been performed in regard to intellectual property of open source software, which is a related area of study. Santos et al. (2011), for example, executed a longitudinal study to evaluate the impact of changes in intellectual property policy of open source projects. Additionally, studies requiring the analysis of R code from the CRAN archive have been published, but not specifically related to authorship attribution. For example, Atchison et al. (2018) utilized R code from the CRAN archive to analyze the topic space of scientific computing.

5. The ASAP Tool

The tool proposed for R source code authorship attribution is the ASAP tool (Tennyson, 2019). It was chosen because of its ease of use, variety of experimental settings, and proven performance. To install the ASAP tool, all one must do is download the ASAP folder to any location on the target computer. The ASAP folder can be obtained and downloaded from the following GitHub repository: https://github.com/ASAP-Project/ASAP.

The ASAP tool requires a Java runtime engine and Perl. If the user does not have Java or Perl installed, those can be obtained for free online. Additional instructions are provided in the tool's documentation. Once the user has Java and Perl, all that must be done to run the tool is execute the ASAP.jar file.

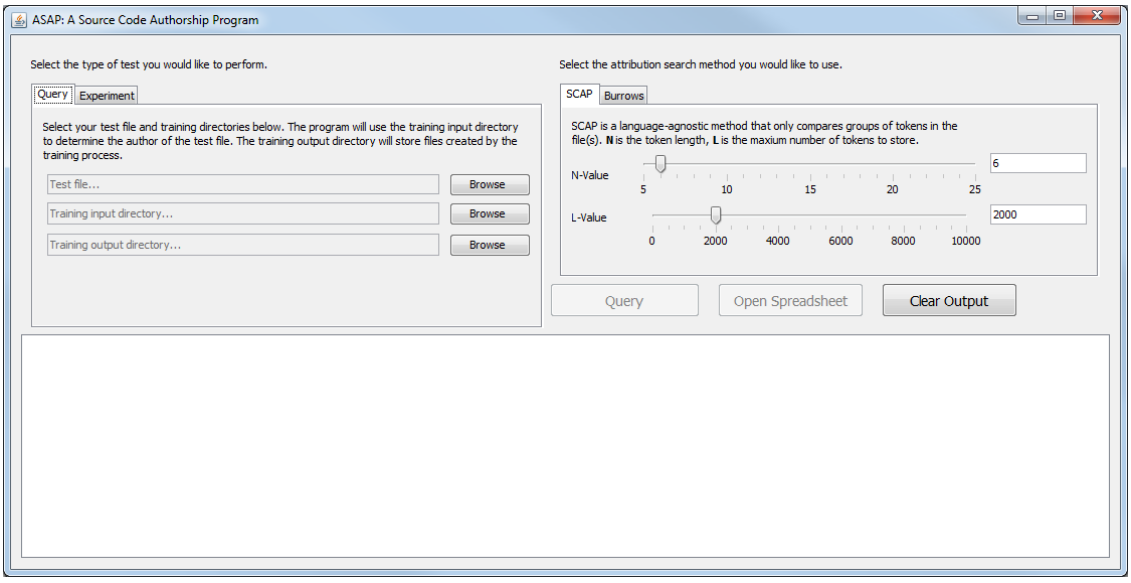

Figure 1. ASAP GUI

Once the ASAP.jar file is executed, a GUI will appear (see Figure 1). This GUI contains the options for the query or experiment the user can run. First, the user can choose between running a query or an experiment. A query will test for the author of a single unknown file. If a query is selected, the user must select their test file, their input directory, a directory containing folders from the possible authors with their known source code inside, and the output directory. An experiment will attribute the author to many documents at once. If an experiment is selected, one of three different types of experiments can be chosen. The first is a default-split experiment. This "queries all the files in the test directory, using the files in the training directory to attribute authorship." (Tennyson, 2019) The next is a k-fold experiment. This splits the dataset into k number of folds. One of these folds is used at a time to query while the others are used to attribute authorship. The final type of experiment supported by the ASAP tool is a leave-one-out experiment. This tests every file, one at a time, against every other file in the test directory. For a more detailed description of the types of test the ASAP tool offers, the reader is encouraged to read Tennyson (2019).

After a test type is determined, the user must then determine the method of authorship attribution they would like to use. The ASAP tool provides two options for this: the SCAP method (Frantzeskou et al., 2007) and the Burrows method (Burrows and Tahaghoghi, 2007). If the SCAP method is chosen, the user has the option to change the token length and maximum number of tokens to store. A more in-depth description of the SCAP method is provided further in the paper.

The Burrows method is not used in this study. The main reason for excluding the Burrows method is that it depends on the programming languages being analyzed: its search utilizes features of the specific programming language, such as keywords, identifiers, operators, etc. In order to apply the Burrows method to the R programming language, features would have to be selected for the R language, which is outside the scope of this study. Due to its current lack of support for the R programming language, an explanation of the Burrows method and its settings supported by the ASAP tool are omitted.

Author	NumFiles	NumCorrect	Percentage
AaronRobotham	24	20	83.3%
AbdulMajedRajaRS	5	4	80.0%
AdamLund	12	11	91.7%
AdamRothman	20	20	100.0%
AdelinoFerreiradaSilva	25	23	92.0%
AlbertoKroneMartins	17	16	94.1%
AlboukadelKassambara	9	9	100.0%
AlexCannon	53	51	96.2%
AlexZvoleff	14	14	100.0%
AlexisDinno	9	9	100.0%

File	AaronRobotham	JulMajedRaj	AdamLund	damRothma	noFerreirad	rtoKroneMa
addhead.R	47	5	12	15	15	18
car2sph.R	101	12	23	33	35	29
deg2dms.R	97	18	32	36	42	31
deg2hms.R	92	12	21	32	31	27
dms2deg.R	117	18	50	68	53	43
genparam.R	47	10	40	26	36	57
hms2deg.R	107	17	30	54	49	32
IAUID.R	48	9	12	16	12	13

Figure 2. ASAP Excel spreadsheet output samples. Attribution totals (top). Each file's attribution for a specific author (bottom).

Once the test type and method of authorship attribution are determined, the user can simply click the "Query" or "Experiment" button and the Java GUI will run the necessary Perl commands to run the test. Once the experiment is finished, an Excel spreadsheet will be generated with the results (see Figure 2), and the user can click the "Open Spreadsheet" button to view it. Additionally, if the user wishes to bypass the GUI, such as in the case of running an experiment with an L-value larger than 10,000, then the user can run the Perl commands themselves from the command line.

6. The SCAP Method

The method of source code authorship attribution chosen for this project was the SCAP method. It was chosen over the other method incorporated by the ASAP tool, the Burrows method for reasons mentioned above.

The SCAP method, unlike the Burrows method, is not dependent on any set programming language features. Thus, it can be used with any programming language. It also has proven high performance in the C++ and Java languages (Frantzeskou et al., 2007). All of these factors contributed to our choosing of this method for testing authorship attribution in R. A brief overview of how the SCAP method works is provided below.

The SCAP method of source code authorship attribution creates a unique profile for each author. The program whose author is unknown is compared against each author's Source Code Author Profile (SCAP) using a similarity measure known as the Simplified Profile Intersection (SPI). The SCAP that is the most similar to the program is attributed authorship of that program.

An author's profile is created by concatenating together all of the available programs known to be written by that author. Instead of representing this profile as plain text, it is represented as a table of byte-level n-grams. Since these n-grams are extracted at the byte level, the SCAP method is not dependent on any programming language. All the characters in all the files by an author are converted into n-grams. Only the L most frequent n-grams are kept in the table, and this table becomes the author's profile.

The SPI is "the number of n-grams an author profile and a program have in common" (Tennyson, 2019). For a program being tested, the author that has the highest SPI with it is attributed authorship.

7. Methodology

To test the effectiveness of the SCAP method of authorship attribution with the R programming language using the ASAP tool, two experiments were run. First, an experiment on clean data was run. This featured a dataset that we knew contained R programs written by specific individual authors. Second, an experiment was run on unclean data. This was done to replicate what real-world data might look like. It contained any R programs we could find, even ones with multiple or unclear authors. All of the experiments were executed on a machine with an Intel Core i7 CPU running at 1.8 GHz with 4 cores and 16 GB of RAM. A more detailed description of how the data for these experiments were gathered and the ASAP settings chosen is provided below.

7.1. Gathering and Cleaning the Data

To gather the data for the clean experiment, the CRAN R Project Archive (CRAN Archive, 2021), one of the most popular repositories of R code, was chosen as the source of the R programs. It was chosen primarily because of its very large repository of R programs and its documentation of the authorship of its packages.

From this repository, one unique author with more than 2 packages published was chosen at a time. Only authors with more than 2 packages published were chosen to limit the number of authors and prevent authors with too few files from being included in the dataset. If an author had a similar name to another author, the author with fewer packages was excluded. If a package had multiple authors, that package was not included in the dataset. These steps were done to ensure that each author profile would represent a distinctive author.

Once an author was selected, its package names were manually reviewed to see if any had the same base name. If any did, they were excluded from the dataset. For example, "package1" and "package2" both have the base name, "package", so they would both be excluded. If more than two usable packages still existed by that author, then those packages were downloaded from the CRAN Archive into a folder named after that author. In the case that an author had more than 2 usable packages and the download failed to acquire 1 or more of them, the author was kept in the dataset as there is another step further on that handles authors with too few files. A total of 855 packages from 272 authors were downloaded.

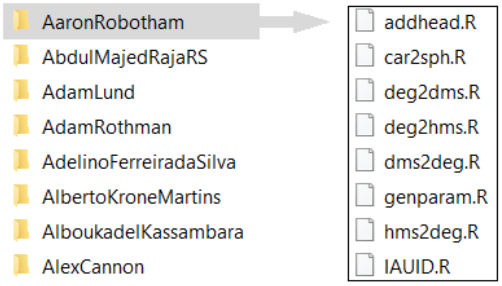

Figure 3. File structure. Author folders all in the same folder. Inside each author folder are the files that belong to that author.

After all the available authors and their usable packages were downloaded, the author folders contained compressed packages in a tar.gz format. Those were programmatically decompressed and unpackaged. The packages by an author, now in a normal folder format, contained many files pertaining to testing and documentation of those packages. The actual R files for that package were in a folder called "R". Anything that was not in that folder was programmatically deleted. The files inside that folder were brought out of that folder and that folder was deleted. The R files were then brought out of their respective packages into the author folder, and the now-empty packages were deleted. This led to the author folders containing only the R programs written by that author, the file structure needed for the ASAP tool (see Figure 3).

To ensure that none of the programs were duplicates of each other in the entire dataset, we programmatically checked the similarity of the text in the files. If any of the files were more than 95% similar to another, they were considered duplicate and both removed from the dataset. In the case that one file was more than 95% similar to another but the second file was less than 95% similar, they were still both excluded. In total, 33 files were found to be duplicates and were excluded. It should be noted that this is much different than the process of attributing authorship. This step simply compared the text of the programs while source code authorship attribution focuses on more specific details such as stylistic features of an author.

With the duplicate files gone, any author with fewer than four R programs was programmatically deleted. This was to ensure that each author had enough data to build a profile that represented them as accurately as possible. In total, there were 6,197 R files representing 225 authors for the clean dataset.

To gather the data for the unclean experiment, we simply programmatically downloaded as many authors and their respective packages from the CRAN Archive as possible. If a package was authored by more than one person, those people together were treated as a unique author. If an author appeared more than once with a slightly different name, we treated both of those appearances as different authors.

Not much cleaning was done on the data in comparison with the data used for the clean experiment. The packages were decompressed and put into the same file structure as the clean data using the same programmatic approaches. Any author with fewer than four files was removed from the dataset to ensure they would have an accurate profile. This led to a total of 5,674 authors and 96,074 R files for the unclean dataset.

7.2. Running the Clean Experiment

Three different experiments were run on the clean dataset. The only variable changed throughout these was the L-value, the length of the author profile, to test how that length would impact the results.

The first experiment was run on the dataset of 225 authors and 6,197 R files using a leave-one-out experiment with the SCAP method incorporated in the ASAP tool. A leave-one-out experiment was chosen to ensure that every program got individually compared against every author profile to produce the most accurate results; it maximizes sampling while avoiding overfitting.

An n-gram length of 6 bytes and an L-value of 690,000 were chosen. That L-value was selected because previous research showed that the higher the L-value, the better the attribution accuracy (Tennyson and Mitropoulos 2014). The L-value was almost 1,000 over the number of bytes the largest author folder had, 689,003. This ensured that every n-gram from every author would be represented. The ASAP GUI has a max L-value of 10,000, so to use the L-value of 690,000, we ran the command generated by the ASAP tool from the command line with the L-value we chose instead of the default 2,000.

The second and third experiments were run using the same leave-one-out experiment using the SCAP method. The ASAP GUI was used in these experiments with an n-gram length of 6 bytes and an L-value of 10,000 and 5,000 respectively. These were done to see how a smaller L-value might impact the accuracy of the authorship attribution.

7.3. Running the Unclean Experiment

Only one experiment was run on the unclean data. After attempting to run a leave-one-out experiment with an L-value of 10,000, we realized that including all 5,674 authors and 96,074 files was not computationally feasible for us. The reason it was not feasible is explained in the next paragraph.

A leave-one-out experiment removes one file for testing and performs training using all of the other files. This is done for every file in the data set, thus maximizing the number of tests as well as the size of the training set for each test. So, in this case, the top 10,000 n-grams of 96,074 files would be compared to the top 10,000 n-grams of 96,073 other files. This leads to a total of 9,230,117,402 file comparisons. If 100 comparisons could be done a second, this would take over 1,068 days to complete. (9,230,117,402 / 100 comparisons / 60 seconds / 60 minutes / 24 hours = 1,068.3). As a note, 100 comparisons per second is not necessarily representative of the speed of an experiment. The speed will vary based on the processing power of the computer running it; we observed less than 100 comparisons per second. To make the data possible to experiment on, we excluded files and authors from the dataset.

The steps for fairly shrinking the clean dataset are as follows and were done programmatically: remove any author with fewer than ten files, randomly remove authors until only 300 are left, randomly remove files from each of those authors until each author is left with only 10 files. This left the unclean dataset with 300 authors, 3,000 R programs, and 8,997,000 file comparisons - something much more realistic to run an experiment on.

After this, a leave-one-out experiment with an L-value of 10,000 and n-gram length of 6 bytes was run. These are the same parameters as the second experiment run on the clean dataset.

8. Results

The results of running a leave-one-out experiment using the SCAP method with different L-values on the clean dataset, where we knew each file belonged to its respective author, can be seen in Table 1. As is shown, the L-value 690,000 resulted in 83.59% of files being correctly attributed to their author. When the profile length was 10,000 n-grams, 92.29% of files were correctly attributed. Finally, with an author profile length of 5,000 n-grams, 91.22% of the files were attributed correctly.

Table 1. Results of using the SCAP method with different L-values on clean data.

N-Gram Length (bytes)	L-Value (n-grams)	Number of Files	Number Correctly Attributed	Percentage Correct
6	**690,000**	**6,197**	**5,180**	**83.59%**
6	**10,000**	**6,197**	**5,719**	**92.29%**
6	**5,000**	**6,197**	**5,653**	**91.22%**

For the experiment with an L-value of 10,000, a 5-number-summary of the percentage of each author's files correctly attributed is shown in Table 2. An accompanying box plot can be seen in Figure 4. The impact of the number of files each

author has in the dataset on the percentage of their programs correctly attributed can be seen in Figure 5. The number of files has been natural-log-transformed to better show this relationship.

Table 2. Summary of author files correctly attributed using L=10,000 on clean data.

5-Number Summary	
Min	**0%**
Q1	**82.35%**
Median	**93.33%**
Q3	**100.00%**
Max	**100.00%**

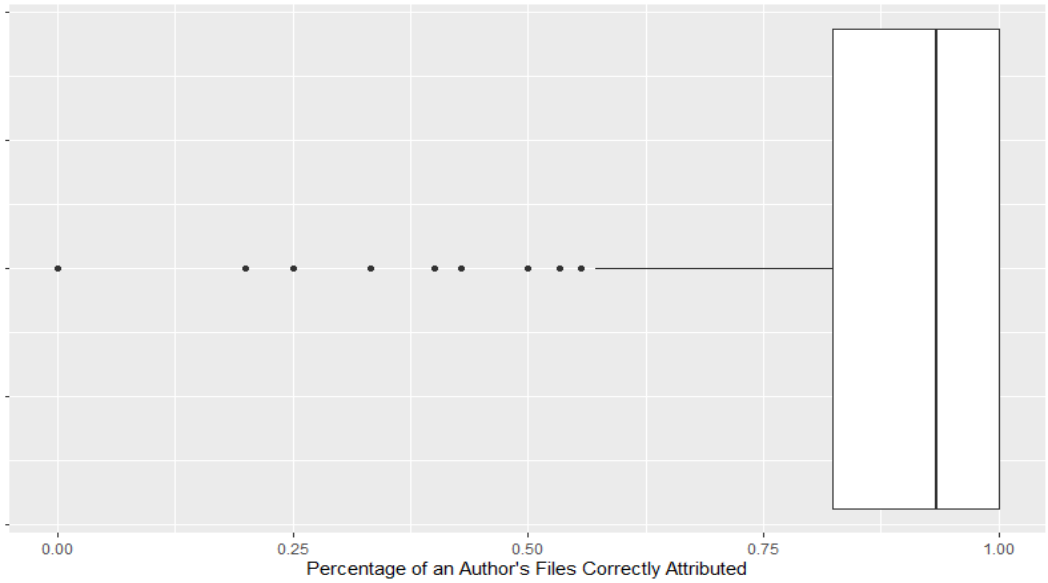

Figure 4. Box plot displaying the 5-number summary from Table 2.

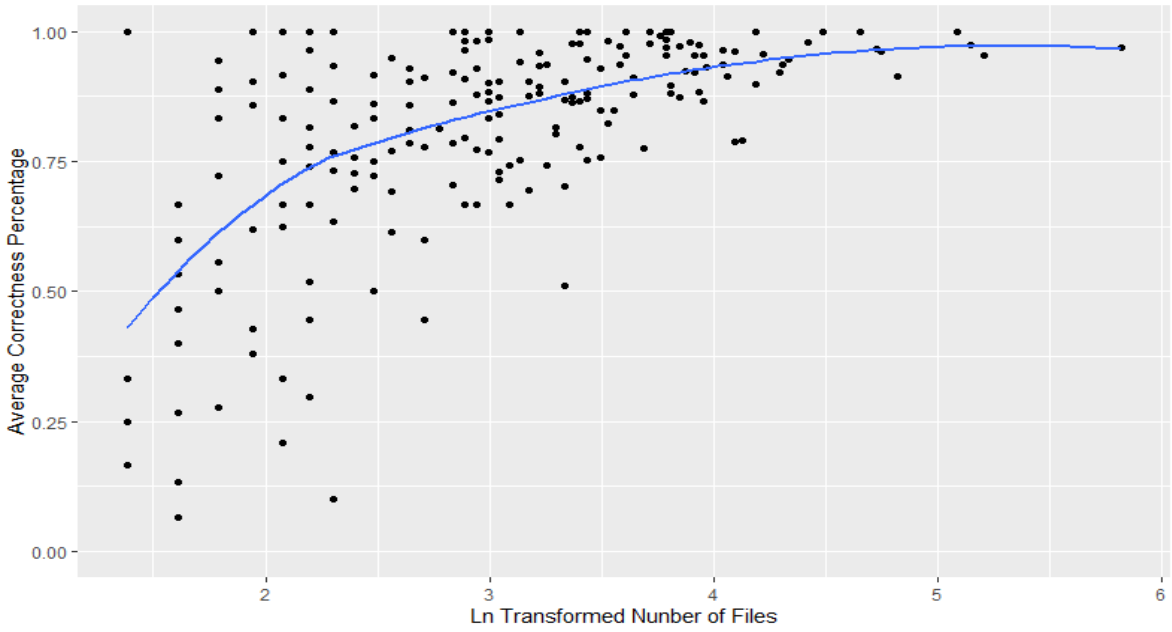

Figure 5. The percentage of each of the 10,000 n-gram length authors' files correctly attributed vs the ln-

transformed number of files that author has.

The results of running a leave-one-out experiment using the SCAP method with an L-value of 10,000 on the unclean dataset, where we could not be sure each file belonged to its respective author, can be seen in Table 3. As is shown, 2,682 out of 3,000 R programs were attributed to their correct author. A 5-number-summary and accompanying boxplot are also shown for this experiment in Table 4 and Figure 6, respectively.

Table 3. Results of using the SCAP method with an L-value of 10,000 on unclean data.

N-Gram Length (bytes)	L-Value (n-grams)	Number of Files	Number Correctly Attributed	Percentage Correct
6	10,000	3,000	2,682	89.4%

Table 4. Summary of author files correctly attributed using L=10,000 on unclean data.

5-Number Summary	
Min	40%
Q1	80.00%
Median	90.00%
Q3	100.00%
Max	100.00%

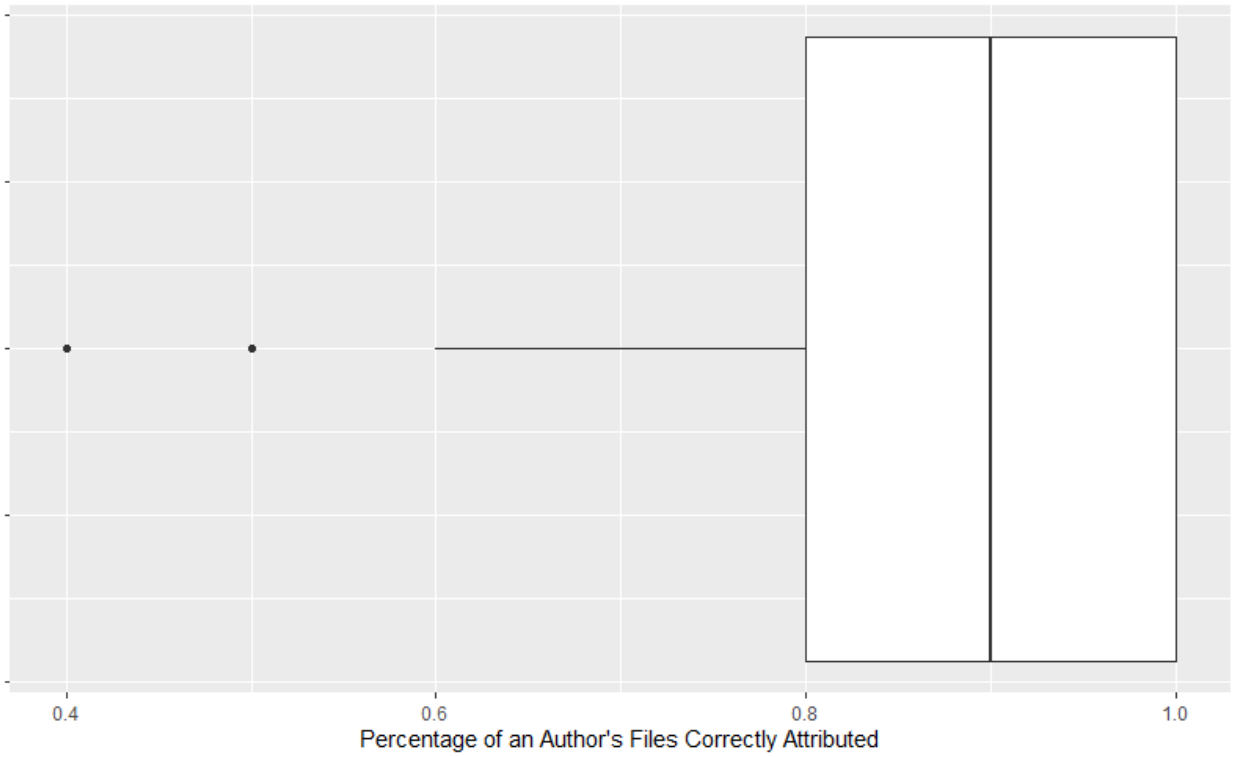

Figure 6. Box plot displaying the 5-number summary from Table 4.

9. Analysis of Results

The leave-one-out SCAP method experiments run on the "clean" dataset correctly attributed different percentages of the files based on how large the L-value was. As can be seen, the experiment with an L-value of 10,000 performed better than a virtually infinite L-value and a smaller L-value. This was unforeseen by us because previous research (Tennyson and Mitropoulos, 2014) implied that the larger the L-value, the more accurate the results.

In that experiment with an L-value of 10,000, 92.29% of the R programs were attributed to their correct author. This is comparable to results from previous research using the SCAP method on Java and C++ programs (Frantzeskou et al., 2007; Tennyson and Mitropoulos, 2014).

Despite the increase in profile length seeming to have a negative impact on the results of the experiment, increasing the number of files per author seems to improve the accuracy of that author's attribution. This is quite intuitive: the more samples of code you have for someone, the more accurate of a profile you can build for them.

The results of the leave-one-out SCAP experiment with an L-value of 10,000 on the "unclean" dataset showed 89.4% accuracy. As was expected, this is a lower accuracy than the one achieved with the "clean" dataset. In the "unclean" dataset, which more closely follows real-world data, one cannot be sure that any given R program was written by the same sole author as the rest of the programs in the author profile. This leads to less accurate profiles. As an example, if an author profile consists of nine R programs written by one author and one R program written by another, the author profile will consist mostly of n-grams from the author with nine R programs. If the programming style of the author with only 1 R program in that package is different than the other author, then that R program will likely not be correctly attributed to that package.

Although the experiment with the "unclean" data did not perform as well as the "clean" dataset, it still performed relatively well. Nearly 90% of the files were attributed to their correct author.

10. Implications

In an academic setting, the need for determining the author of an R program is clear. First, it is needed to ensure that students did not plagiarize or collaborate with one another on an assignment. Second, it could be needed by academic journals to maintain integrity and ensure that everyone is getting the credit they deserve. The need for attributing authorship of R code also exists outside of an academic setting. It is easy to imagine many situations where companies producing products that use R would want to ensure their products are wholly written by their employees for both legal and ethical reasons.

The way to determine if a program was written by a student in one's class, a researcher submitting to one's journal, or an employee at one's place of work is not as simple as just checking for blatant plagiarism. Comparing files to see what percentage of one was copied from the other is straightforward, and there are many tools available to check for this. The real problem arises when someone copies another's code but modifies it. The SCAP method of authorship attribution integrated in the ASAP tool can be used to address such situations in academic and industry contexts.

For example, one can imagine a scenario where an instructor keeps all of the assignments submitted by students so far, both individual and group assignments. The instructor could then run each file submitted for the current assignment through the ASAP tool, using the previous assignments to build an author profile for each student. If the students have worked together before, or one contributed much more to their current assignment they were not supposed to collaborate on, one of their R programs will likely not be attributed to the student that submitted it. The instructor, seeing this, could then delve deeper into the code to make a judgement.

In a similar fashion, a journal editor can build a collection of profiles of R code authors using one or more popular R code repositories such as CRAN (used in this study), GitHub, or Stack Overflow. Given that the journal requires submitting R

code used for data retrieval, processing, and analysis – the submitted code can be compared to existing profiles of known R code contributors to see whether it has a high match with one or more authors (other than the ones who authored the paper). If it does closely match the code of other R code authors, the editor can make sure that the authorship of the code used in the study is properly attributed to avoid potential issues with intellectual honesty or even copyright infringement.

In the workplace, an employer could utilize the ASAP tool with the SCAP method to build profiles for its employees or company as a whole. This could include a routine check on the codebase against a collection of open-source projects to ensure that the company is not, knowingly or unknowingly, using the code of someone else. It could also potentially be used as evidence in cases of copyright infringement. Since the SCAP method builds profiles based on the programming styles of the authors, a profile could be built for the company as a whole, which would hypothetically reflect their coding standards and the style of the employees as a collective. If a suspect R program is more similar to Company A's profile than Company B's profile, then it would be more likely written by Company A.

11. Future Work

Further work could be done to improve upon the knowledge of the effectiveness of authorship attribution with the R programming language and authorship attribution as a whole. One such example is determining what the optimal L-value for the SCAP method is. As is shown in the results of the running the experiment on the "clean" data, the percentage of correct attribution is dependent on the number of n-grams included in the author programs, the L-value. While the L-value of 10,000 yielded good results for our experiment, there is likely a value that is better, and discovering that could improve the accuracy of authorship attribution in R. Another possible task for future research is determining a feature set to be used to apply the Burrows method of authorship attribution to the R programming language. Generative AI is increasingly being used to develop software, including R code. It would be interesting to study how generative AI affects the problem of authorship attribution. A final example could be running more experiments on real-world data with differing files per author. To shrink our "unclean" dataset without bias, we kept the number of files per author consistent, but it is possible that the results of the experiment would have been slightly different if the number of files per author had remained variable like in the "clean" dataset. In a real-world application, there are scenarios where the number of files per author will be both variable and consistent, so determining the effect of the variability in file number could improve the accuracy of R authorship attribution.

12. Conclusion

With the continual rising popularity of R for students, academic researchers, and industry Data Scientists - the need for determining the author of an R program is also increasing. More and more people are needing to write code in R for their personal, work, or academic purposes. Along with this, the amount of R resources available with a simple online search are increasing by the day. The practice of "copying and pasting" someone else's code into one's own research project and modifying it for the project's unique purposes is commonplace. This gives rise to the ethical and legal problem of deciding who is the actual author of the code.

We recommend the use of the ASAP tool based on the SCAP for one's R authorship attribution needs. It is a program that is simple to install, comes with an intuitive GUI, and implements the SCAP method that has shown successful performance attributing authorship to R code. As is shown in previous sections of the paper, the SCAP method using a leave-one-out experiment and an L-value of 10,000 was able to correctly attribute authorship of R programs 92.29% of the time. Even when files were just thrown into the experiment with no confidence that each author profile consisted of programs written solely by that author, it was able to correctly attribute authorship 89.4% of the time. Thus, we deem this method of R code authorship contribution to be useful for many real-world applications.

13. References

Alqahtani, F. and Dohler, M. (2023). Survey of Authorship Identification Tasks on Arabic Texts, *ACM Transactions on Asian and Low-Resource Language Information Processing*, *22*(4), 1-24.

Atchison, A., Anderson, H., Berardi, C., Best, N., Firmani, C., German, R., and Linstead, E. (2018). A topic analysis of the R programming language, *Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings*, 183-184.

Bainbridge, D. I. (1999). *Software Copyright Law*. West Sussex, UK: Bloomsbury Professional.

Bogomolov, E., Kovalenko, V., Rebryk, Y., Bacchelli, A., and Brykson, T. (2021). Authorship attribution of source code: a language-agnostic approach and applicability in software engineering, *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 932-944.

Burrows, S. and Tahaghoghi, S.M.M. (2007). Source code authorship attribution using n-grams, *Proceedings of the 12th Australasian Document Computing Symposium*, 32-39.

CRAN (2020). Retrieved from https://cran.r-project.org/web/packages/

CRAN Archive (2021). Retrieved from https://cran.r-project.org/src/contrib/Archive/

Creative Commons (2021). Three "Layers" of Licenses. Retrieved from https://creativecommons.org/licenses/

Dauber, E., Caliskan, A., Harang, R., and Greenstadt, R. (2018). Git blame who? Stylistic authorship attribution of small, incomplete source code fragments, *Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings*, 356-357.

Ding, H. and Samadzadeh, M. (2004). Extraction of java program fingerprints for software authorship identification, *The Journal of Systems and Software, 72*, 49-57.

Dryer, A.J. and Stockton, J. (2013). Internet "Data Scraping": A Primer for Counseling Clients. New York Law Journal. Retrieved from https://www.law.com/newyorklawjournal/almID/1202610687621

Fawcett, L. (2018). Using Interactive Shiny Applications to Facilitate Research-Informed Learning and Teaching. *Journal of Statistics Education, 26*(1), 2-16.

Frantzeskou, G., Stamatatos, E., Gritzalis, S., Chaski, C., and Howald, B. (2007). Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method. *International Journal of Digital Evidence, 6*(1).

Gonzalez, H., Stakhanova, N., and Ghorbani, A.A. (2018). Authorship Attribution of Android Apps, *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, 277-286.

Griffiths, T. (2022). Python vs. R: What's the Difference? Retrieved from https://www.indeed.com/career-advice/career-development/r-vs-python

Hendrikse, S. (2017). *The Effect of Code Obfuscation on Authorship Attribution of Binary Computer Files*, Nova Southeastern University, Fort Lauderdale-Davie, FL.

Kalgutkar, V., Kaur, R., Gonzalez, H., Stakhanova, N., and Matyukhina, A. (2019). Code Authorship Attribution: Methods and Challenges, *ACM Computing Surveys, 52*(1), 1-36.

Krotov, V. and Tennyson, M. (2018). Research Note: Scraping Financial Data from the Web Using the R Language. *Journal of Emerging Technologies in Accounting, 15*(1), 169-181.

Lander, J. P. (2014). *R for Everyone: Advanced Analytics and Graphics*. Boston, MA: Addison-Wesley.

Lev, B. (2000). *Intangibles: Management, measurement, and reporting*. Brookings institution press.

Makhmutova, L., Ross, R., and Salton, G. (2023). Impact of Character n-grams Attention Scores for English and Russian News Articles Authorship Attribution. *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*, 939-941.

Niezgoda, S. and Way, T. P. (2006). SNITCH: a software tool for detecting cut and paste plagiarism, *ACM SIGCSE Bulletin, 38*(1), 51-55.

Peng, R. D. (2011). Reproducible research in computational science, *Science, 334*(6060), 1226-1227.

Petrik, J. and Chuda, D. (2021). The effect of time drift in source code authorship attribution: Time drifting in source code – stylochronometry, *Proceedings of the 22nd International Conference on Computer Systems and Technologies*, 87-92.

Prechelt, L., Malpohl, G., and Philippsen, M. (2002). Finding plagiarisms among a set of programs with JPlag. J. *UCS, 8*(11), 1016.

R Project (2021). What is R? Retrieved from https://www.r-project.org/about.html

RStudio (2021a). Shiny from RStudio. Retrieved from: https://shiny.rstudio.com

RStudio (2021b). Take control of your R code. Retrieved from https://www.rstudio.com/products/RStudio/

Santos, C.D., Cavalca, M.B., Kon, F., Singer, J., Ritter, V., Regina, D., and Tsujimoto, T. (2011). Intellectual property policy and attractiveness: a longitudinal study of free and open-source software projects, *Proceedings of the ACM 2011 conference on computer supported cooperative work*, 705-708.

Schutt, R. and O'Neil, C. (2013). *Doing data science: Straight talk from the frontline*. Sebastopol, CA: O'Reilly Media, Inc.

Sebastián, S., Diugan, R.G., Caballero, J., Sanchez-Rola, I., and Bilge, L. (2023). Domain and Website Attribution beyond WHOIS, *Proceedings of the 39th Annual Computer Security Applications Conference*, 124-137.

Stack Overflow (2021). Questions tagged [r]. Retrieved from https://stackoverflow.com/questions/tagged/r

Stamatatos, E. (2008). A survey of modern authorship attribution methods. *Journal of the American Society for Information Science and Technology, 60*(3), 538-556.

Tennyson, M. (2013). *Authorship Attribution of Source Code*, Nova Southeastern University, Fort Lauderdale-Davie, FL.

Tennyson, M. (2019). ASAP: A Source Code Authorship Program. *International Journal on Software Tools for Technology Transfer*.

Tennyson, M. and Mitropoulos, F. (2014). Choosing a profile length in the SCAP method of source code authorship attribution. *IEEE SOUTHEASTCON 2014*. Lexington, KY, USA, 13-16 March 2014. IEEE.

Uchendu, A., Le, T., and Lee, D. (2023). Attribution and Obfuscation of Neural Text Authorship: A Data Mining Perspective, *ACM SIGKDD Explorations Newsletter*, *25*(1), 1–18.

Wickham, H. (2014). *Advanced R*. Boca Raton, FL: CRC Press

Zhao, Y. and Zobel, J. (2005). Effective and scalable authorship attribution using function words. *Asia Information Retrieval Symposium*, 174-189.

Author Biographies

Austin Coursey is a Computer Science Ph.D. student in the Institute for Software Integrated Systems at Vanderbilt University. He earned his B.S. in Computer Science and Mathematics from Murray State University in 2022. His research applies and develops deep learning techniques to solve challenging problems in complex systems. His recent work has addressed robust unmanned aerial vehicle control using reinforcement learning, continual reinforcement learning, anomaly detection for building energy consumption and freeway traffic incidents, and data-driven prognostics for aircraft engines and hard disk drives. He authored this work during his time as an undergraduate student at Murray State University.

Dr. Matthew Tennyson is an Associate Professor of Computer Science at the Department of Computer Science and Information Systems, Arthur J. Bauernfeind College of Business, Murray State University. Matthew earned his B.S. in Computer Engineering from Rose-Hulman in 1999. He worked at Caterpillar as an engineer, developing embedded systems for various types of earth-moving machinery. He earned his M.S. in Computer Science from Bradley University. He earned his Ph.D. at Nova Southeastern University in 2013. Matthew's teaching and research interests include software engineering, programming practice and theory, and computer science education.

Dr. Vlad Krotov is a Professor of Information Systems and Cybersecurity Management at the Department of Computer Science and Information Systems, Arthur J. Bauernfeind College of Business, Murray State University. Dr. Vlad Krotov received his PhD in Management Information Systems from the Department of Decision and Information Sciences, University of Houston (USA). His teaching, research and consulting work is devoted to helping managers and organizations to use Information and Communication Technologies for analyzing organizational data in a way that enhances organizational performance. His quantitative and qualitative research has appeared in a number of academic and practitioner-oriented journals and conferences, such as: CIO Magazine, Journal of Theoretical and Applied E-Commerce, Communications of the Association of Information Systems, Business Horizons, Blackwell Encyclopedia of Management, America's Conference on Information Systems (AMCIS), Hawaii International Conference on System Sciences (HICSS), International Conference on Mobile Business (ICMB). His research was recognized by the 2016 Outstanding Researcher award and 2017 Emerging Scholar award at Murray State University.