Information and Software Technology 148 (2022) 106924

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

How ReadMe files are structured in open source Java projects

Yuyang Liu[1], Ehsan Noei [*,1], Kelly Lyons[1]

University of Toronto, Canada

ARTICLE INFO

Keywords:
Empirical study
ReadMe files
Software popularity
Clustering

ABSTRACT

Context: Recent studies on open source platforms, such as GitHub, provide insights into how developers engage with software artifacts such as ReadMe files. Since ReadMe files are usually the first item users interact with in a repository, it is important that ReadMe files provide users with the information needed to engage with the corresponding repository.

Objective: We investigate and compare ReadMe files of open source Java projects on GitHub in order to (i) determine the degree to which ReadMe files are aligned with the official guidelines, (ii) identify the common patterns in the structure of ReadMe files, and (iii) characterize the relationship between ReadMe file structure and popularity of associated repositories.

Method: We apply statistical analyzes and clustering methods on 14,901 Java repositories to identify structural patterns of ReadMe files and the relationship of ReadMe file structure to repository stars.

Results: While the majority of ReadMe files do not align with the GitHub guidelines, repositories whose ReadMe files follow the GitHub guidelines tend to receive more stars. We identify 32 clusters of common ReadMe file structures and the features associated with each structure. We show that projects with ReadMe files that contain project name, usage information, installation instructions, license information, code snippets, or links to images tend to get more stars.

Conclusion: ReadMe file structure shares a statistically significant relationship with popularity as measured by number of stars; however, the most frequent ReadMe file structures are associated with less popular repositories on GitHub. Our findings can be used to understand the importance of ReadMe file structures and their relationship with popularity.

1. Introduction

GitHub[2] is a popular open-source version control platform based on Git[3] that serves both as a repository and a community, hosting activities, such as pull requests and issue tracking, work processes, and work outputs [1]. At the time of this research, GitHub hosted more than 100 million repositories (28 million of which are public repositories) and served over 37 million users.[4]

One of the advantages of open-source software development is that collaboration is common and contributions to software projects from members of the community are encouraged [2]. In fact, having new people continually join open-source projects and participate in the development process is critical for project success [3]. As such, successful open source projects must find ways to encourage developers, especially those new to the project, to contribute to the development activities of the project [4,5]. One of the ways that projects encourage participation is through documentation [6] and ReadMe files [7]. In fact, ReadMe files, one of the first things that a potential contributor reviews when accessing a GitHub repository [8,9], contribute to the initial impression a developer has of the software project [7] and may influence a decision to participate in its development activities.

Because of the importance of ReadMe files, GitHub provides guidelines for developers and recommends that there be at least one ReadMe file in each repository. Although there are no restrictions on the formatting and content of ReadMe files, GitHub also suggests ReadMe files follow a specific format in order to help users quickly locate important information [8]. For example, it is recommended that ReadMe files include the following sections: description, table of contents, installation information, usage instructions, contribution instructions, author credits, and license information [8].

* Corresponding author.
E-mail addresses: yuyang@cs.toronto.edu (Y. Liu), e.noei@utoronto.ca (E. Noei), kelly.lyons@utoronto.ca (K. Lyons).

[1] Yuyang Liu, Ehsan Noei, and Kelly Lyons equally contributed to the research.

[2] http://www.github.com.

[3] https://git-scm.com/.

[4] https://web.archive.org/web/20191010081343/https://en.wikipedia.org/wiki/GitHub.

https://doi.org/10.1016/j.infsof.2022.106924
Received 15 April 2021; Received in revised form 22 March 2022; Accepted 7 April 2022
Available online 12 April 2022

Previous research has reported that developers and engineers do not always follow best practices or recommendations [10,11]. For example, Businge et al. [10] found that 44% of Eclipse Plugin developers use the discouraged practice of non-API interfaces in their implementations even though non-API interfaces are unstable and unsupported. Elazhary et al. [11] found that the actual activity observed in projects studied on GitHub differ from the documented contribution guidelines for those projects.

In this paper, we investigate how closely ReadMe files follow the GitHub guidelines and the relationship between the popularity of software projects and the ways in which their ReadMe files are structured. GitHub offers users the ability to *star* a repository as a way of expressing an appreciation for a repository and bookmarking it to keep track of its activities [12]. Past studies have used stars as a proxy for the popularity of repositories [13–16]. Repository popularity is important in open source communities because more popular repositories attract new contributors [13].

We study 14,901 open-source Java repositories and their ReadMe files that are hosted on GitHub. We chose Java projects because Java is a popular and well-established programming language [17]. We investigate the structure of ReadMe files by identifying 16 commonly included sections in Java repository ReadMe files (including the 7 sections recommended by the GitHub guidelines) and 6 additional structural elements. We find that more than half of the ReadMe files do not include sections recommended in the GitHub guidelines. Although developers might have strong motivations for not following the recommendations [18], in the case of formatting GitHub ReadMe files, we observe that the alignment of ReadMe files with GitHub guidelines shares a statistically significant relationship with popularity. Specifically, repositories whose ReadMe files contain GitHub-recommended sections receive more stars.

If most ReadMe files do not follow the GitHub guidelines, we are interested in understanding more about how ReadMe files are structured and the relationship of structure to popularity. Therefore, we identify the format of the ReadMe files using 22 structural variables: 16 variables that measure whether each of the 16 sections we identified are included in the ReadMe file or not and 6 variables that measure additional structural statistics for each ReadMe file. We cluster the ReadMe files into similar groups using the 22 variables and analyze the relationship of each cluster with popularity as measured by number of stars [13]. Using Kruskal–Wallis and Dunn's tests we show that 56.25% of clusters have ReadMe files whose repositories have statistically significant different distributions of stars from those in the other clusters.

Our findings provide insights into the structures of ReadMe files. Furthermore, our study is not limited to ReadMe files written only in English, and, therefore, our findings are relevant to repository owners all around the world. We address the following research questions:

(RQ1) In what ways do English ReadMe files of Java repositories align with GitHub guidelines?
Although GitHub provides official guidelines to help developers document their repositories, we observe that the majority of ReadMe files do **not** include the GitHub recommended sections, i.e., installation information, table of contents, usage instructions, contribution instructions, credits, and license information. We find that repositories whose ReadMe files are more aligned with GitHub guidelines receive more stars.

(RQ2) How are ReadMe files of Java repositories formatted?
In the first research question, we find that most of the ReadMe files are not formatted following the official GitHub guidelines; therefore, we identify the most common formats. We apply DBSCAN [19], which is an unsupervised learning algorithm to group the ReadMe files according to their formatting, resulting in the identification of 32 different formats of ReadMe files.

(RQ3) What is the relationship between ReadMe file format and the popularity of the associated Java repositories?
We compare the popularity of the repositories in the 32 clusters (based on their ReadMe file structures) and we observe that 56% of the clusters are statistically significantly different from each other when comparing their repository star counts. For example, the repositories for which their ReadMe files include installation instructions and usage information are more popular than the clusters with only a project name and some screenshots.

Paper Organization. Section 2 describes our research methodology. Section 3 presents our approach and findings for each of the three research questions. We discuss the threats to validity in Section 4 and present related work in Section 5. Finally, Section 6 concludes with a summary of findings, contributions, and suggestions for future research.

2. Research methodology

Fig. 1 provides an overview of the study setup. First, we gather repositories with language set to Java using GHTorrent [20] and GitHub API [21]. Then, as detailed in Section 2.1, we perform a number of preprocessing steps, such as removing repositories with only one committer. After that, we measure the number of repository stars and parse the ReadMe files to collect information about 22 structural variables in order to cluster and study ReadMe file structures and the relationship with repository popularity.

2.1. Collecting data

We use the GitHub historical data provided by GHTorrent [20] as of 2019-06-01 to collect the repositories that we study. We use three main tables from GHTorrent: (i) the *projects* table to capture general information about the projects and the hyperlink to its GitHub page, (ii) the *commits* table to assess project activity level, and (iii) *watchers* table to capture star information. Users can add a repository to their favorites by clicking the *star* button for that repository. These users are called *stargazers* and receive notifications of updates for the repositories they have starred. The *watchers* table captures the star actions of stargazers [22]. We use the *repo_id*, the unique identifier for repositories, and *user_id* to aggregate the number of stars for each repository. We invoke the GitHub API V3 [21] to retrieve the ReadMe files of the repositories in our study.

We gathered all the repositories in GHTorrent that have not been deleted from GitHub and have their identified language field as Java (8,288,310 in total). We removed personal repositories (those with only one committer) because personal repositories are usually used either for experimentation or storing data [1] and are not intended for use by other members of the open source community. In this case, the associated ReadMe files are not likely being used to convey information to members of the open source community and, hence, are not relevant to our study. Removal of personal repositories resulted in 1,305,832 non-personal repositories. We also removed repositories that do not have an associated ReadMe file resulting in 699,236 repositories.

We wanted to make sure that all ReadMe files in our study were being used to convey information about the associated repositories. We considered what to do with repositories with no stars at all. On the one hand, zero stars may indicate information about the popularity (or lack thereof) of a repository; however, a repository with zero stars may also indicate that there is no active engagement by others. In this case, it is not clear if the associated ReadMe file is being used to, or expected to be used to, convey information about the repository. We manually inspected 385 out of 699,236 repositories (a representative sample with a 95% confidence level) that had no stars. We observed that such repositories are mainly: (i) small demos, (ii) course projects, and (iii) test projects for personal exercises. Therefore, we decided to remove repositories with no stars, resulting in 161,333 Java repositories, all of which have been engaged with actively by others as indicated by at least one star.

A forked repository on GitHub initially inherits the characteristics of the main repository including its ReadMe file; however, forked repositories do not inherit the stars from their parent repositories. We manually

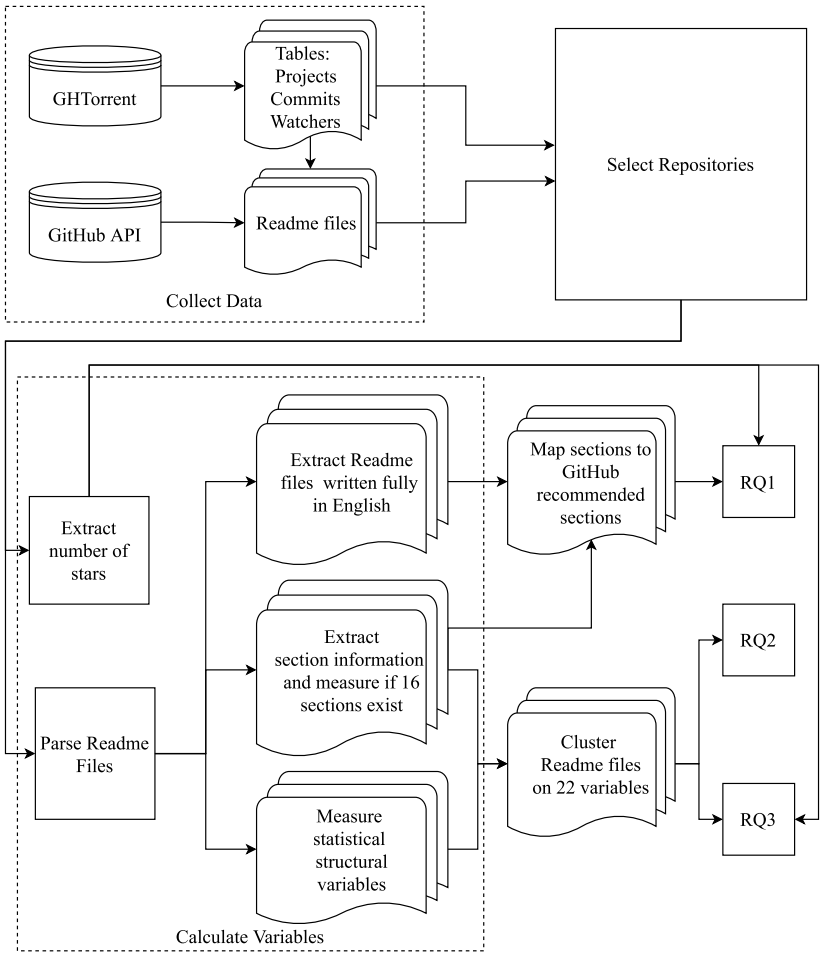


Fig. 1. Overview of the study setup.

inspected a randomly sampled set of 385 out of 161,333 ReadMe files from the forked repositories that have more than zero stars. While the majority (76%) of the sampled ReadMe files are identical or only slightly different from the ReadMe files of the repositories from which they where forked (e.g., added references, updated author information), over 97.1% of the forked repositories have a different number of stars than the repositories from which they were forked (with the forked repositories having many fewer stars). Our clustering algorithm will group repositories with similar ReadMe files and, in the case of forked repositories which may have similar ReadMe files, the difference in stars may be solely because stars are not inherited. To mitigate the noise introduced by the forked repositories, we excluded them from our study, resulting in 142,471 Java repositories that are not forked from another repository.

Finally, to reduce the number of Java ReadMe files that need to be manually inspected in our study while maintaining the statistical significance of our findings, we randomly sampled 14,901 out of the 142,471 repositories with a 99% confidence level and confidence interval of 1. Our sampled repositories have a median of 24 commits and average of 131 commits. The medium time between two adjacent commits in our selected repositories is 3.5 days and the average is 19.3 days.

2.2. Identifying structural elements of the readme files

GitHub suggests that ReadMe files use *GitHub flavored Markdown* [23], a GitHub-extended dialect of the original *Markdown* style[5] which is a lightweight markup language with readability and is easy parsed into HTML. We parse the ReadMe files using Mistune[6] which is a popular Python Markdown parser.

First, we identified sections in the ReadMe files. GitHub supports six levels of headings [23]: H1 (the largest) to H6 (the smallest). We extracted the six levels of headers from the parsed ReadMe files and one of the authors inspected a random sample of 383, out of 142,471 repositories (with a 95% confidence level). As expected [24], we observed that developers use different terms (keywords) to refer to the same concept, such as *Setup*, *Install*, or *Installation* as a section heading used to provide information about installing a repository. A second author then inspected a different random sample of 383 out of the 142,471 repositories. The two inspections resulted the same 16 commonly included sections (represented by the assigned variable names in the first column of Table 1). Furthermore, for 14 of the identified sections, there was 100% agreement on the keywords used to identify the sections in the ReadMe files. In the case of two sections (the *description* and *installation* sections) there were only minor differences between the identified keywords of each inspection (with Cohen's Kappa [25] of 0.83 (strong match)). The keyword *what is* was identified as a keyword for the *description* section in the first inspection (but not in the second) and *guide* was identified as a keyword for the *installation* section in the second manual inspection (but not in the first). After discussion, it was agreed that the keyword *what is* would be used to identify the *description* section but the keyword *guide* would not be used to identify the *installation* section. In column two of Table 1,

[5] https://daringfireball.net/projects/markdown/.

[6] https://github.com/lepture/mistune.

Table 1
The list of 16 commonly identified sections in the ReadMe files (column 1), keywords searched for in the headings to identify each section (column 2), the corresponding GitHub-recommended sections (column 3), and a brief description of each GitHub-recommended section (column 4).

Assigned variable names	Heading keyword(s)	Corresponding GitHub-recommended sections	Description
description_header_kw, project_name_header_kw	Describe, description, overview, about, summary, introduction, what is.	Description	Description is used to provide an overview of a repository. It should be brief, but thorough, and convey the goals for the repository.
content_header_kw	Content	Table of contents	It is optional, but can be useful for users to quickly find the parts they are interested in.
install_header_kw	Install, build, setup, download, compile	Installation	The installation section describes how to install a project, and GIF images useful for illustration purposes.
usage_header_kw document_header_kw example_header_kw troubleshoot_header_kw	Use, usage, quickstart, run, start document, docs example, demo, sample troubleshoot	Usage	The usage section explains how to use a screenshots can be included here.
contribut_header_kw	Contribute	Contributing	Contribution instructions are presented here, especially applicable to large projects. It can also be in a separate file.
credit_header_kw author_header_kw	Credit, acknowledge author	Credits	Credits list the authors of the repository.
license_header_kw	License, licence, copyright	License	The license that is used if applicable.
test_header_kw archive_header_kw screenshot_header_kw deprecate_header_kw	Test archive screenshot deprecate, retire		Additional keywords that show up frequently in the ReadMe file headers but do not map to GitHub recommended sections.

we list the final set of terms (keywords) that we searched for in the headings to determine whether the 16 commonly included sections were present in the ReadMe file or not. We mapped these sections to the sections recommended by the GitHub guidelines [8]: description, table of contents, installation, usage, contributing, credits, and license (the third column of Table 1). Note that the bottom row contains commonly-included sections that do not map to any of the GitHub recommended sections.

We also gathered information about additional structural elements of the ReadMe files using various statistics shown in Table 2. The first column indicates the variable we assigned to each of these descriptive statistics about the ReadMe files.

We assessed whether ReadMe files were written entirely in English. Most of the ReadMe files in our set of 14,901 are written entirely in English (i.e., they contain only English words and characters). There are 3307 ReadMe files that contain some non-English words or characters and only 133 out of 3307 are completely non-English (i.e., they have no English words/characters at all). Of the ReadMe files that have a mix of English and non-English words, on average, the proportion of non-English words is just over 34%.

We counted the number of headings (from all six header levels) in the parsed ReadMe files. The ReadMe files in our dataset have an average of 4.32 headers, with 1090 ReadMe files having no headers at all, and 75% of ReadMe files having at most 6 headers.

GitHub allows a section for highlighting programming code snippets in specialized Markdown syntax. We extracted code snippets in the parsed ReadMe files and counted the number of snippets to measure how many coding snippets are included in the ReadMe files. On average, the ReadMe files contain 4.37 code snippets presented in Markdown syntax but there are 9165 ReadMe files out of 14,901 that do not include any code blocks.

GitHub Markdown supports displaying images. We extracted the HTML tags that are used for displaying images from the parsed ReadMe files and counted the number of images. The average number of images in the ReadMe files is less than one (0.80), with 4109 ReadMe files having at least one image in them. We also counted the number of hyperlinks in the ReadMe files using HTML link tags. We found that most ReadMe files (8159 out of 14,901) include at least one hyperlink in them, and there are 3.43 hyperlinks on average in our ReadMe files.

To give an indication of ReadMe file size, we counted the number of tokens in each ReadMe file. First, we removed stop words, punctuation, digits, white space, numbers written in plain English (e.g., six), hyperlinks, and non-ASCII tokens using spaCy [26] which is an open-source library for natural language processing in Python. There are 223.2 tokens in our ReadMe files on average, with approximately 67.3% of ReadMe files having no more than 20 tokens.

For each repository, we also measured the number of stars as an indicator of popularity [13,27].

3. Approach and findings

(RQ1) In what ways do English ReadMe files of Java repositories align with GitHub guidelines?

Motivation

As stated by the official GitHub guidelines [8]: *"Making documentation accessible enables people to learn about a project; making it easy to update ensures that documentation stays relevant. It is a good idea to at least have a* **ReadMe** *on your project, because it is the* ***first thing*** *many people will* ***read*** *when they* ***first find*** *your work"* (***emphasis*** added) [8]. GitHub also recommends including specific sections, such as an installation section, in the ReadMe files. For this research question, we investigate to what degree ReadMe files follow the GitHub guidelines. We also look into the relationship between the inclusion of GitHub guideline sections and repository popularity (as measured by number of stars).

Approach

As explained in Section 2.2, we identify the GitHub sections in the ReadMe files by finding headers that include the keywords shown in Table 1 column 2. For example, we look for the keywords *credit*, *acknowledge*, or *author* to identify the GitHub *credits* section. Note that for

Table 2
The statistical information gathered for each ReadMe file and the motivation for including it as a structural element of the ReadMe files.

Variable	Description	Motivation
isEnglish	Whether or not a ReadMe file is written in English.	ReadMe files are written in both English and non-English.
readme_header_count	The number of sections in a ReadMe file.	More headers indicate more sections which may improve readability.
readme_code_block_count	The number of code blocks in a ReadMe.	Code blocks are straightforward to understand and may lower natural language barriers for international developers.
readme_image_count	The number of images in a ReadMe file.	Images may be beneficial for providing information, training, and demonstrating the functionality and features of a repository.
readme_url_count	The number of hyperlinks in a ReadMe file.	Hyperlinks are used to link to other resources and web pages. Hyperlinks can be used to provide more details and references.
readme_tokens_count	The number of tokens, after the preprocessing steps in Section 2.2, in ReadMe files.	A larger ReadMe file provides more material and content for users.

Table 3
The number of English java ReadMe files that contain each recommended section versus the ones that do not. The asterisks in the last column indicates if the popularity of the repositories associated with each group is statistically significantly different.

Section	Not included	Included	*p-value*		Cohen's d
Table of contents	11,417 (98%)	177 (2%)	5.69e−007	***	0.219
Credits	11,188 (96%)	406 (4%)	7.62e−023	***	0.287
Contributing	10,974 (95%)	620 (5%)	6.98e−068	***	0.415
License	10,123 (87%)	1471 (13%)	5.56e−147	***	0.416
Installation	9196 (79%)	2398 (21%)	2.43e−122	***	0.311
Usage	7979 (69%)	3615 (31%)	1.50e−161	***	0.314
Description	6864 (59%)	4730 (41%)	3.07e−004	***	0.039

this research question, we focus on those ReadMe files written entirely in English, i.e., 11,594 out of 14,901 repositories, as our identifying keywords are in English.

We measured the popularity (number of stars) of each repository, and, using the Mann–Whitney U test [28], we analyze how the number of stars relates to the inclusion of the guideline sections in ReadMe files. The *null* hypothesis in the Mann–Whitney U test states that the distributions of stars between the ReadMe files containing or not containing each recommended section are the same. For each recommended section, such as the *installation* section, with a $p\text{-}value \leq 0.05$, we reject the *null* hypothesis and, thus, conclude that the distributions of stars of the repositories associated with the ReadMe files containing the recommended section are statistically significantly different from distribution of stars for repositories whose ReadMe files do not contain the section.

We also measured the effect size by calculating Cliff's Delta of the difference in the number of stars between the repositories that contain each recommended section in their ReadMe files and the repositories that do not include that section in their ReadMe files [29]. Cliff's Delta measures how much overlap exists in the number of stars of each pair of repositories. If the number of stars for the repositories whose ReadMe files include certain sections is greater than the number of stars for the repositories whose ReadMe files do not, the Cliff's Delta will be greater than 0. Cliff's Delta gets closer to 1 as the difference in the number of stars increases. Similarly the Cliff's Delta is less than 0 if the number of stars for the repositories whose ReadMe files include certain sections is less than the number of stars for the repositories whose ReadMe files do not and gets closer to −1 as the difference increases. We then interpret the Cliff's Delta by the standards of Cohen's d. Specifically, we map a Cliff's Delta of 0.474, 0.330, and 0.147 to large, medium, and small effect size, respectively [30,31].

Results

Table 3 (column 2) shows the number (and percentage) of repositories with ReadMe files that do not include each of the recommended sections, ranging from 59% for the *description* section to 98% for *table of contents* section. Fig. 3 shows the number of stars for the repositories with English ReadMe files that contain each section (lighter color, right hand side) versus those that do not (darker color, left hand side).

The *description* section provides users with an introduction to the repository. As shown in Table 3, this section is the most common section included in the ReadMe files, having been included in 41% of the ReadMe files in our study. As illustrated in Fig. 3, the repositories whose ReadMe files do not contain this section are statistically significantly less popular than ones that do contain a *description*. With Cohen's d of 0.039, the difference in the number of stars between including the description section and not including description section is small.

Table of contents is an optional section according to GitHub guidelines and it is only included in 177 of the ReadMe files we analyzed (i.e., 2%). However, despite being optional, we observe a statistically significant difference in the number of stars for the repositories with ReadMe files having a *table of contents* with a $p\text{-}value \leq 0.05$ and Cohen's d of 0.219.

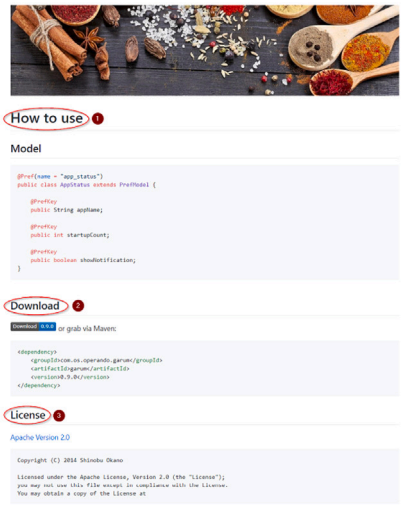
The *installation* section gives instructions for setting up a repository. This section appears in only 21% of the repositories in our dataset, while the inclusion of this section shares a statistically significant relationship with more popular repositories with a $p\text{-}value \leq 0.05$. With a Cohen's d of 0.311, the inclusion of an *installation* section has a small effect size on the number of stars for the repositories.

The *usage* section appears in the ReadMe files of 3615 repositories (i.e., 31%). This section should instruct users how to use the repository after installation. With a $p\text{-}value = 1.50e^{-161}$ and Cohen's d of 0.314, the ReadMe files including this section are associated with more popular repositories with a small effect size.

Out of the 11,594 English ReadMe files, only 620 (5%) have a dedicated section for *contributing* information; however, we observe a statistically significant relationship between this section and popularity. The median number of stars associated with ReadMe files including the *contributing* section is 9 while the median number of stars for the repositories without this section is only 2 (see Fig. 3). Including a *contributing* section in ReadMe files has a medium effect (Cohen's $d = 0.415$) on the number of stars. The *contributing* section encourages users to make contributions to the repository, and, therefore, could be a reason for the higher recorded popularity.

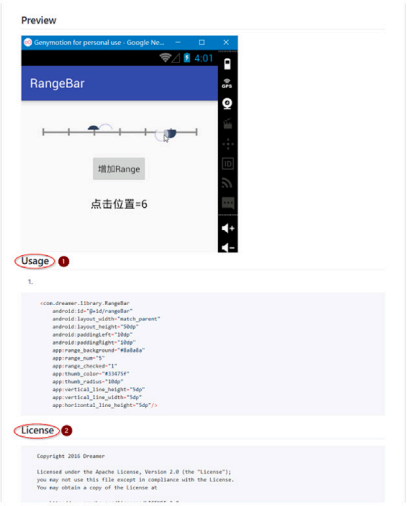
Out of the 11,594 English ReadMe files, only 406 (4%) of them have a section for *credit* information; however, the ones that have a *credit* section, as shown in Fig. 3, are associated with more popular repositories. With a Cohen's d of 0.287, having a *credit* section in the ReadMe file has small effect size on the number of stars.

The *license* section is provided in 1471 (13%) of English ReadMe files in our dataset. Fig. 3 shows that the repositories that include a *license* section are more popular compared to the ones without a *license*

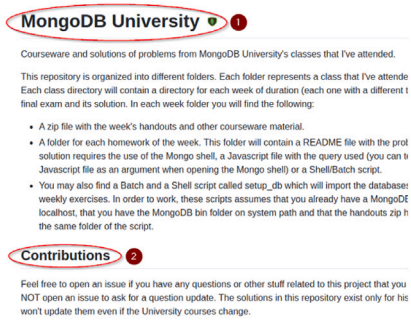


(a) README with ① usage, ② installation, and ③ licence sections (cluster 1). https://github.com/operando/Garum

(b) Non-English README with ① project name, ② license, images, and code blocks (cluster 13). https://github.com/Ryanst/ FloatingBall



(c) Non-English README with ① usage and ② license sections (cluster 27). https://github.com/YangShaoXiong/ RangeBar



(d) README with ① project name and ② contributing section (cluster 23). https://github.com/aurasphere/ mongodb-university-classes

Fig. 2. Example README files from clusters 1, 13, 23, and 27.

section, which is also reflected in terms of Cohen's $d = 0.416$ with a medium effect size.

In summary, the majority of the English README files do not include the recommended sections as shown in Table 3. Developers tend to include the *description* and *usage* sections more often than the other sections (41% and 31% for *description* and *usage* sections, respectively). On the other hand, the other recommended sections are rarely included in the README files (between 2% and 21%). During the initialization of a repository on GitHub, a trivial README file with repository name is automatically generated, which is possibly motivating developers to enter a description for their repository. There are statistically significant differences in the popularly of the repositories that include each section versus the ones that do not, as observed by the Mann–Whitney U test, with $p\text{-}values \leq 0.05$ for each recommended section (see last column of Table 3).

Furthermore, we observe that including the sections of *contributing* and *license* share a larger difference in terms of the distribution of stars. Also, the Cohen's d values of including these two sections (0.415 and 0.416, respectively) shows that they both have the highest effect size among all the recommended sections. We considered the average number of stars in README files that contain (or do not contain) each GitHub-recommended section. To further interrogate these results, we carried out a small qualitative analysis of 13 popular Java repositories (high number of stars) whose README files do not include the recommended sections. We found that popular repositories whose README files do not contain GitHub recommended sections have a small number of contributors (between 2 and 15), have a large number of forks (between 363 and 1200), and most (all but 2) have open issues, an indication of repository activity. It is important to note that the number of forks has been found to correlate with the number of stars [32] so a high number of forks for these repositories is not surprising; however, it is possible that having a small number of contributors means that communication through README files is not as critical to project success, especially in projects with regular activity (as indicated by issues).

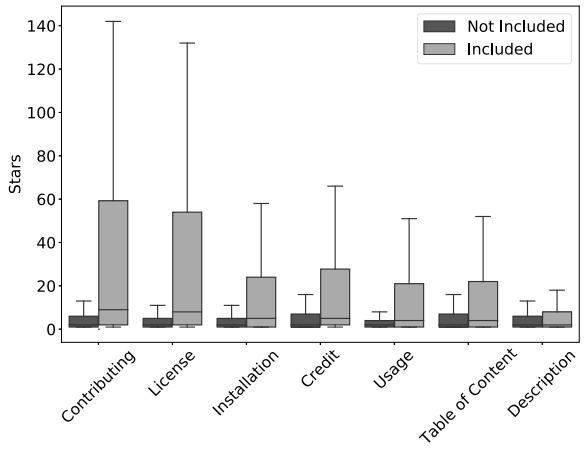


Fig. 3. The distribution of stars of Java repositories with English ReadMe files that include (lighter bar) and do not include (darker bar) each GitHub-recommended section, sorted by decreasing median number of stars in the lighter bars. Repositories whose ReadMe files include these sections receive more stars.

> *The majority of English* ReadMe *files (83.47% on average) are not aligned with the GitHub guidelines. However, the repositories whose* ReadMe *files are aligned with GitHub guidelines receive statistically significantly more stars.*

(RQ2) How are ReadMe files of Java repositories formatted?

Motivation

As observed in **RQ1**, the majority of ReadMe files are not formatted according to GitHub guidelines. Hence, we identify common patterns of formatting for ReadMe files. Understanding how ReadMe files are typically formatted informs the open-source community about the most and the least frequently used patterns. Moreover, GitHub and other similar open-source repository platforms can update their guidelines according to our findings, e.g., introducing new content categories to help detail the most frequently included sections and providing developers with more detailed instructions on documenting projects.

Approach

We are interested in the structure of ReadMe files beyond just whether there is alignment with GitHub guidelines; thus, for this research question, we consider all of the 14,901 ReadMe files in our set: those written entirely in English, and those with a mix of English and non-English words or with no English words. It is interesting that some ReadMe files with a mix of English and non-English words have headers written in English (for example, see Fig. 2(b) and (c)). In this case, we measure the existence of the English headers in ReadMe files that had both English and non-English words. We then treat our set of 22 structural elements (Tables 1 and 2) as a feature vector and use the Vector Space Model [33] to represent the ReadMe files in the feature space. We use DBSCAN [19,34], a density based non-parametric clustering algorithm, to categorize the ReadMe files from the 14,901 Java repositories. Similar ReadMe files, with similar feature vectors, are a shorter distance from one another in the vector space.

DBSCAN considers each ReadMe file as a node in the feature space and the nodes that appear together within a high density area are clustered together. We normalize statistical elements in order to give each metric a fair contribution to the clustering algorithm. We use *cosine* distance similarity in DBSCAN [33] to calculate the distance between the nodes. There are two hyper-parameters in DBSCAN that need to be set prior to running the algorithm:

- ϵ: The maximum distance between two nodes to be considered in one cluster.
- μ: The minimum number of nodes in one cluster forming the core nodes.

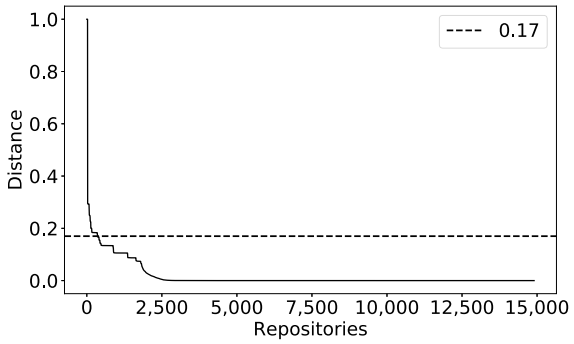


Fig. 4. The *k-dist* plot: distance graph for our Java repositories. The x-axis is the number of repositories whose kth nearest neighbor is at the distance given on the y-axis. The horizontal line intersecting at distance 0.17 shows the first knee point.

The nodes that have at least μ neighbor nodes within the ϵ distance are called the core nodes. Starting from the core nodes, if traveling along the paths formed by the core nodes, DBSCAN groups all the nodes that are reachable from each of the others in one cluster. The points that are not reachable by any of the nodes are considered as *noise*.

We use the heuristics proposed by Ester et al. [19] and Birant et al. [35] to determine μ, and, as a result, set $\mu = 10$. For finding the best ϵ, we followed the same heuristics provided by Ester et al. [19] and Birant et al. [35] where *k-dist*, i.e., the distance between a node to its kth nearest neighbor, is more sensitive to noisy nodes compared to the nodes in a high density area. By plotting the *k-dist* graph, mapping each node to the distance from its kth nearest neighbor, we identify the best ϵ (i.e., the maximum distance).

Fig. 4 shows the *k-dist* plot, illustrating different values of distance against the number of repositories. The optimum value of ϵ should be around the first knee point (highlighted by a horizontal line) which is between 0.17 and 0.18. After manually inspecting different values of ϵ, we identified 0.17 as the best estimation of ϵ for clustering our ReadMe files.

Results

Table 4 describes the structure of the ReadMe files in each cluster. There are 32 clusters of ReadMe files that resulted by applying DBSCAN on the 22 ReadMe file structural elements. Out of 14,901 ReadMe files, only 185 were marked as noisy (i.e., the ReadMe files that could not be fit in any of the clusters).

Table 4 lists which of the 16 sections are included in each cluster of ReadMe files and, for the structural variables, it reports the average value for each variable across ReadMe files. In Table 4, the clusters of ReadMe files are sorted by cluster size (the number of ReadMe files in each cluster).

Cluster 1 contains the largest number of repositories. The ReadMe files in this cluster have 9 sections on average and more than half of them provide *installation* instructions and *usage* information (i.e. 54% and 65%, respectively). However, we could not find a common pattern of sections in this cluster. Also, in the fourth column of Table 4, 1 indicates if all the ReadMe files in a cluster are entirely in English, and 0 indicates otherwise (i.e., ReadMe files are either non-English or a mixed usage of English and non-English). Unlike other clusters (which are either entirely in English or not), in Cluster 1, 92% of the ReadMe files are written all in English and the remaining ones have a mixed usage of English and non-English (indicated by a dash in Table 4). Fig. 2(a) shows an example ReadMe file in this cluster, showcasing the *usage*, *installation*, and *license* sections.

The second largest cluster of ReadMe files is Cluster 0. The ReadMe files in this cluster are all written in English but they either use plain

Table 4
Overview of the ReadMe file structural elements in all of the identified clusters, sorted by the number of ReadMe files in each cluster. The right five columns show the average value in the cluster.

ID	Size	Sections included	All in English	Avg value of structural variables				
				header_count	code_block_count	image_count	url_count	tokens_count
1	4350	Mixed usage of some sections	–	9	10	1	7	476
0	3515	None of the 16 included	1	1	1	0	1	87
5	2872	Project name	1	1	1	0	1	55
4	1234	None of the 16 included	0	4	4	1	3	138
11	1171	Project name	0	3	2	1	2	81
2	375	Usage	1	5	6	1	3	286
6	251	Install	1	4	5	1	3	234
8	145	Example	1	3	4	0	2	175
3	119	Description	1	4	3	1	2	232
10	110	License	1	3	2	1	4	186
19	61	License	0	6	5	3	5	189
26	45	Usage	0	8	5	3	4	320
7	38	Install	0	6	4	1	5	344
12	36	Test	1	3	2	0	2	111
13	35	Project name, license	0	6	5	2	3	151
16	34	Project name, usage	0	10	13	1	7	453
15	33	Document	1	4	1	1	6	177
23	31	Contribute	1	4	1	2	7	189
31	31	Example	0	8	4	1	7	264
28	30	Screenshot	1	3	0	2	2	86
22	26	Project name, install	0	10	4	1	2	165
24	23	Project name, example	0	7	6	3	2	184
17	22	Install, usage	0	10	12	1	5	720
20	21	Description	0	8	15	1	6	543
25	18	Content	1	4	1	1	3	234
9	17	Install, license	0	9	18	3	10	657
21	16	Project name, test	0	7	1	4	2	125
27	13	Usage, license	0	11	16	3	8	579
14	12	Credit	1	5	3	0	4	506
29	11	Test	0	8	6	1	23	290
30	11	Project name, description	0	9	8	2	3	220
18	10	Author	1	3	0	1	2	93

text to present their ReadMe files or are not formatted using GitHub Markdown to create sections. Cluster 0 is similar in terms of structure to Cluster 4 (the fourth largest cluster). However, ReadMe files in Cluster 4 are not entirely in English. Moreover, ReadMe files in Cluster 4 contain a larger number of sections (4 on average) in comparison to Cluster 0 (1 on average). Also, ReadMe files in Cluster 4 are longer than those in Cluster 0 on average (see the last column in Table 4).

The third and fifth largest clusters are Cluster 5 and 11, respectively. All the ReadMe files in these clusters include the repository name in the ReadMe files. However, the ReadMe files in these clusters do not include any of the other GitHub recommended sections.

As shown in Table 4, the ReadMe files in Clusters 0, 5, 4, and 11 have a relatively small number of *section headers*, *code blocks*, and *images*. As also shown in Table 4, the clusters with ReadMe files with non-English words are more likely to have a higher number of headers, code blocks, images, and hyperlinks. This may be a tactic used by non-English speaking developers to attract a larger international audience. However, *installation*, *usage* and *license* sections are the most frequently included sections regardless of the languages used to write the ReadMe files.

Cluster 2 is the sixth largest cluster, and the third largest cluster entirely in English. The ReadMe files in this cluster include the recommended *usage* section and contain 286 tokens on average. On average, they contain one image and six blocks of code. Cluster 6, the next largest cluster, is similar to Cluster 2 in terms of the structural variables, but instead of *usage*, the ReadMe files in Cluster 6 include an *installation* section.

Overall, as in Table 4, among all the 32 clusters of ReadMe files, *project name* has been used commonly in 8 clusters. The next popular sections are *usage*, *install*, and *license* appearing in 5 of the identified clusters each. Example ReadMe files are shown in Fig. 2. The next commonly included sections are *example* and *description*, each of which have been included in three of our clusters.

We identify 32 clusters of ReadMe *file formats.* ReadMe *files tend to include project name, installation instructions, usage information, and license. Those* ReadMe *files that are not written fully in English provide more content in comparison to English ones; for example,* ReadMe *files with non-English words include 7.65 sections and 5.71 hyperlinks while their English counterparts include 3.73 sections and 3.27 hyperlinks on average.*

(RQ3) What is the relationship between ReadMe file format and the popularity of the associated Java repositories?

Motivation

Having identified 32 patterns (clusters) of ReadMe file formats in **RQ2**, we explore the relationship between the identified patterns of ReadMe files and repository popularity reflected by their number of GitHub stars. Our findings provide insights into ReadMe file formats of popular repositories.

Approach

We compare the number of stars associated with the ReadMe files of repositories in each cluster using the Kruskal–Wallis test [36]. As the *null* hypothesis, we assume that all the clusters share the same distribution of number of stars. Kruskal–Wallis test [36] is non-parametric statistical test to check whether the distribution of the number of stars are the same among all the clusters. We reject the *null* hypothesis with a $p\text{-}value \leq 0.05$. After that analysis, we apply the Dunn's test [37] to conduct a pairwise comparison between all of the clusters to identify the ones that have a statistically significant different distribution of stars than the others.

Results

The Kruskal–Wallis test reports a $p\text{-}value = 1.451e^{-306}$; therefore, it rejects the null hypothesis, i.e., there exists at least one cluster with

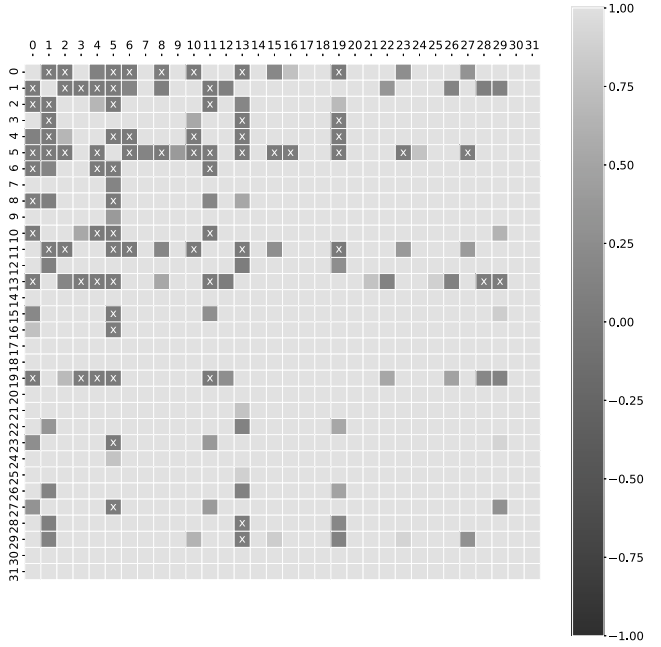


Fig. 5. Pairwise Dunn's test on all the identified clusters. The darker squares represent smaller *p-value*, and lighter represents larger. The cross (X) marks on the heatmap denote *p-values* ≤ 0.05.

a statistically significant different distribution of stars from the rest of the clusters. By applying the Dunn's test, as shown in Fig. 5, we observe that 56.25% of clusters have statistically significant different distributions of stars from each other. The cross (X) marks in Fig. 5 denote if the difference between that pair of clusters (in terms of number of stars) is statistically significant. Fig. 6 shows the distribution of project popularity (i.e., stars) for each cluster, sorted by the median number of stars.

As shown in Fig. 6, Cluster 13 has the highest median number of stars. As presented in Table 4, ReadMe files in this cluster contain non-English words but they include the *project name* (in English) and a section for *license* information. The ReadMe files in this cluster also include several images and code snippets. Fig. 2(b) provides an example ReadMe file in this cluster. In Fig. 2(b), the *project name* and *license* sections are circled.

Cluster 27 is associated with repositories that have the second highest median number of stars. Interestingly, in both Cluster 27 and Cluster 13, the ReadMe files are not written all in English. The ReadMe files in Cluster 27 include *usage* and *license* sections (circled in an example ReadMe file from Cluster 27 in Fig. 2(c)). They are among the largest ReadMe files on average and include a relatively large number of header blocks, code snippets, and images.

Cluster 23 contains ReadMe files of the third most popular group of repositories. Different from the previous two clusters, the ReadMe files in Cluster 23 are written all in English. They also have a *contributing* section with the *project name* in the header. Fig. 2(d) shows an example ReadMe file in this cluster.

Cluster 19, the fourth most popular cluster, has similar characteristics as Cluster 13, except the repositories in this cluster do not contain the project name in their ReadMe file headers.

Cluster 9, the fifth most popular cluster, has the largest average number of tokens and largest average number of code snippets. The ReadMe files in this cluster use code blocks to demonstrate their detailed usage e.g. json format patterns, error code definitions.

As shown in Table 4, and discussed in **RQ2**, clusters 0, 5, 4, and 11 include the largest number of repositories; however, they are among the least popular clusters in terms of stars (see Fig. 6). The repositories in these four clusters do not include any of the recommended sections from GitHub guidelines in their ReadMe files. Furthermore, their ReadMe files have on average fewer than 4 headers, fewer than 4 code blocks, fewer than 1 image, and fewer than 3 URLs, which makes the length of the ReadMe files smaller as measured by number of tokens. As previously investigated in **RQ2**, cluster 0 is comprised of ReadMe files that are not formatted using GitHub Markdown, or are in a different format, or are in plain text. This may suggest that developers should avoid these ReadMe patterns and try to include more sections recommended by the GitHub guidelines and also include more code examples, images that show how to use or setup the repositories, and URL links to other available resources.

> *The differences in the popularity of the repositories associated with each cluster of* ReadMe *files is statistically significant. Repositories with* ReadMe *files that include usage and license sections with several images and code snippets are among the most popular repositories. In fact, the most common patterns of* ReadMe *files are associated with the least popular repositories.*

4. Threats to validity

4.1. Internal validity

We use various variables and keywords to identify each section in the ReadMe files. In order to identify keywords for matching headers, two people independently manually identified keywords. Their findings match strongly with Cohen's Kappa [25] of 0.83. Then, the keywords and variables were engineered using a representative sample of ReadMe files in order to capture semantically-related keywords. We include ReadMe files written in all languages in our study. The inclusion of non-English language ReadMe files might affect the heuristics used to identify the sections in the ReadMe files. To mitigate this threat, we not only engineered the section variables but also introduced structural variables, such as the number of images or code snippets, since they are invariant to languages being used to write ReadMe files. Moreover, since our clustering algorithm identified different clusters for non-English versus English ReadMe files, we were able to compare and contrast the structure of ReadMe files written in English vs. those written in other languages.

ReadMe files may not be the only documentation that developers provide on GitHub. Wiki pages are officially provided by GitHub as well. We selected a representative sample of Java repositories, and found that only 2.93% of them have wikis with non-empty content. Another way of conveying information to prospective contributors is by using repository badges [38]. Analysis of the repositories in our dataset found that more than 85% of the repositories did not have any badges associated with them and 92% have only zero or one badge.

The latest update of GitHub's official guidelines was on July 15, 2016. Therefore, the ReadMe files that were introduced before this date may not have had the opportunity to align with the official guidelines. However, the focus of this study is the alignment of the existing ReadMe files with the latest GitHub guidelines and the relationship of that alignment with repository popularity.

4.2. External validity

Our findings may not generalize well to ReadMe files of repositories written in other programming languages, such as Python or R. However, there exist several languages similar to Java, such as C# and JavaScript, for which our findings can potentially apply. Future research should analyze ReadMe file patterns in other programming languages.

When considering the findings for RQ3, there might be two possible scenarios related to the observed relationship between ReadMe file format and number of stars: (i) stars increase after the changes in the

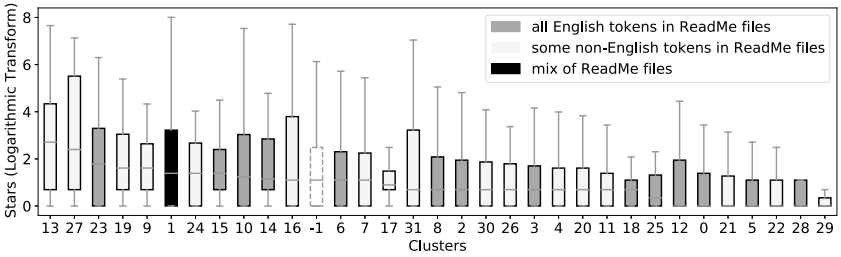


Fig. 6. Distribution of repository popularity (logarithmic transform of the number of stars), sorted by median. Gray bars are clusters with ReadMe files that only have English words, very light gray bars are clusters with ReadMe files that have some non-English words, the black bar is the cluster with a mixture of ReadMe file languages, cluster −1 (dashed outline) contains noisy data points.

structure of ReadMe files, or (ii) ReadMe files are changed after an increase in the number of stars (after becoming more popular). We applied the Granger causality test to shed more light on this question. We investigated 182 repositories that had more than 1 change in the number of identified ReadMe sections, a change in the number of stars by at least one and for which there were at least 4 observed changes in the time series used in the Granger causality test. We formed two time series: (i) ReadMe file alignments (i.e., count of the number of sections recommended by GitHub over time), and (ii) number of stars. The Granger causality test shows that, for 58.2% of the repositories (106 out of 182), the increase in stars occurs after changes are made to the number of recommended sections in the ReadMe file. However, for only 29.1% of the repositories (52 out of 182), the increase in the number of stars happens before changes in the number of recommended sections in the ReadMe files.

4.3. Reliability

This study is based on open source repositories that are available online and can be used in future research. Moreover, the datasets and scripts of this study are available online[7] for an easier replication process.

5. Related work

Stars are commonly used by researchers to represent the popularity of repositories [13–16]. The number of stars has also been found to be positively correlated with number of forks [13]. We decided to use number of stars to represent repository popularity because others have used it and it is relatively easy to retrieve.

Hata et al. [39] applied game theory to project documentation, e.g., the wiki page, to analyze the crucial points for attracting new developers to participate in project development. They found that having documentation for projects can reduce the effort to set up environments, and can encourage developers to participate. However, they did not look into the details of the documentation. Our research focused on the structure of repository documentation and its relationship to popularity.

Aggarwal et al. [6] studied the co-evolution between documentation and popularity of a project and found that consistently popular projects have consistent documentation efforts, which in turn, attracts more collaborators contributing to the documentation. Their research focuses on the changes in the documentation through time and how these changes correlate with popularity. Our work reveals the usage patterns of specific sections in ReadMe file documentation, the extent to which the ReadMe files align with GitHub recommended guidelines and how that alignment is related to popularity.

ReadMe files have also been studied by software engineering researchers. Hassan and Wang [40] used ReadMe files to automatically build software projects. They extracted code snippets from ReadMe files and identified build systems based on keyword matching. Zhang et al. [41] searched for similar projects on GitHub by extracting features from ReadMe files. ReadMe file size was one of several features of repositories used to predict GitHub project popularity in [14]. However, earlier research does not leverage the format information in the ReadMe files, which we incorporate in our structural variables. Prana et al. [7] manually engineered features, such as linguistic patterns and number of elements, from a set of 393 ReadMe files. They trained classifiers based on the features to identify eight categories of ReadMe file content: what, why, who, how, when, references, contributions, and other, that can be used to help users with information discovery. Some of their categories are similar to what we consider as ReadMe sections, e.g. their *how* category is similar to the GitHub *usage* section, and their *contribution* category is similar to the GitHub *contributing* section. However, they did not relate the identified ReadMe file features or content structure to repository popularity.

6. Conclusion

In this paper, we study different patterns that are used to format ReadMe files by investigating 14,901 open-source Java repositories that are hosted on GitHub. First, we compare English ReadMe files with the GitHub guidelines. We observe that the majority of English ReadMe files (83.47% on average) do not include the sections recommended by GitHub. However, the English ReadMe files that are more aligned with GitHub guidelines belong to the repositories that are statistically significantly more popular than other repositories. Second, we calculate 22 structural variables of the ReadMe files and perform a clustering algorithm on the ReadMe files. We identify 32 common patterns for formatting ReadMe files. Most of the ReadMe files tend to include *project name*, *installation instructions*, *usage information*, and a *license* section. Also, we find that ReadMe files that are not written all in English provide more text and media content compared to those with only English words. Third, by comparing the popularity of the repositories that are associated with each cluster of ReadMe files, we observe a statistically significant difference in number of stars associated with each cluster of ReadMe files. We find that the most frequent patterns of ReadMe files are associated with less popular repositories on GitHub. Developers and the open-source community benefit from our findings by understanding the patterns of ReadMe files associated with more popular repositories.

In the future, we plan to extend the research to other programming languages in GitHub and investigate if ReadMe files for repositories written in other languages have similar or different patterns. We will also apply our methodology to other open-source platforms and investigate if ReadMe files in different open-source version control platforms have similar patterns as the ones on GitHub. Finally, we will consider adding other documentation-related signals to understand their relationship to popularity such as wikis [39] and documentation [6].

[7] https://doi.org/10.5281/zenodo.6369440.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was funded in part by an NSERC Strategic Partnership Grant.

References

[1] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D.M. German, D. Damian, An in-depth study of the promises and perils of mining github, Empir. Softw. Eng. 21 (2016) 2035–2071, http://dx.doi.org/10.1007/s10664-015-9393-5.

[2] V. Cosentino, J.L. Cánovas Izquierdo, J. Cabot, A systematic mapping study of software development with GitHub, IEEE Access 5 (2017) 7173–7192, http://dx.doi.org/10.1109/ACCESS.2017.2682323.

[3] Y. Park, C. Jensen, Beyond pretty pictures: Examining the benefits of code visualization for open source newcomers, in: 2009 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis, IEEE, 2009, pp. 3–10.

[4] I. Steinmacher, M.A. Gerosa, D. Redmiles, Attracting, onboarding, and retaining newcomer developers in open source software projects, in: Workshop on Global Software Development in a CSCW Perspective, 2014.

[5] I. Steinmacher, I.S. Wiese, T. Conte, M.A. Gerosa, D. Redmiles, The hard life of open source software project newcomers, in: Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering, 2014, pp. 72–78.

[6] K. Aggarwal, A. Hindle, E. Stroulia, Co-evolution of project documentation and popularity within GitHub, in: Proceedings of the 11th Working Conference on Mining Software Repositories, ACM, New York, NY, USA, 2014, pp. 360–363, http://dx.doi.org/10.1145/2597073.2597120, URL: http://doi.acm.org/10.1145/2597073.2597120.

[7] G.A.A. Prana, C. Treude, F. Thung, T. Atapattu, D. Lo, Categorizing the content of GitHub readme files, Empir. Softw. Eng. 24 (2019) 1296–1327, http://dx.doi.org/10.1007/s10664-018-9660-3.

[8] GitHub, Documenting your projects on GitHub, 2016, URL: https://guides.github.com/features/wikis/.

[9] M. Koskela, I. Simola, K. Stefanidis, Open source software recommendations using GitHub, in: International Conference on Theory and Practice of Digital Libraries, Springer, 2018, pp. 279–285.

[10] J. Businge, A. Serebrenik, M. Brand, Analyzing the eclipse API usage: Putting the developer in the loop, 2013, p. 1, http://dx.doi.org/10.1109/CSMR.2013.14.

[11] O. Elazhary, M.A. Storey, N. Ernst, A. Zaidman, Do as I do, not as I say: Do contribution guidelines match the GitHub contribution process? in: 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, 2019, pp. 286–290.

[12] GitHub, Saving repositories with stars, 2021, URL: https://help.github.com/en/github/getting-started-with-github/saving-repositories-with-stars.

[13] H. Borges, A. Hora, M.T. Valente, Understanding the factors that impact the popularity of GitHub repositories, in: 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2016, pp. 334–344, http://dx.doi.org/10.1109/ICSME.2016.31.

[14] J. Han, S. Deng, X. Xia, D. Wang, J. Yin, Characterization and prediction of popular projects on GitHub, in: 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC), 2019, pp. 21–26, http://dx.doi.org/10.1109/COMPSAC.2019.00013.

[15] A. Zerouali, T. Mens, G. Robles, J.M. Gonzalez-Barahona, On the diversity of software package popularity metrics: An empirical study of npm, in: 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2019, pp. 589–593, http://dx.doi.org/10.1109/SANER.2019.8667997.

[16] W. Mao, B. Sun, G. Xu, C. Liu, C. Si, W. Wang, Understanding effects of collaborations in developing mobile computing systems: Popularity, efficiency, and quality, IEEE Access 7 (2019) 33380–33392, http://dx.doi.org/10.1109/ACCESS.2019.2904333.

[17] M. Storey, A. Zagalsky, F.F. Filho, L. Singer, D.M. German, How social and communication channels shape and challenge a participatory culture in software development, IEEE Trans. Softw. Eng. 43 (2017) 185–204, http://dx.doi.org/10.1109/TSE.2016.2584053.

[18] N. Yang, P. Cuijpers, R. Schiffelers, J. Lukkien, A. Serebrenik, Painting flowers: reasons for using single-state state machines in model-driven engineering, in: Proceedings of the 17th International Conference on Mining Software Repositories, 2020, pp. 362–373. 17th International Conference on Mining Software Repositories, MSR ; Conference date: 25-05-2020 Through 26-05-2020.

[19] M. Ester, H.P. Kriegel, J. Sander, X. Xu, A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise, in: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI Press, 1996, pp. 226–231, URL: http://dl.acm.org/citation.cfm?id=3001460.3001507.

[20] G. Gousios, The GHTorrent dataset and tool suite, in: Proceedings of the 10th Working Conference on Mining Software Repositories, IEEE Press, Piscataway, NJ, USA, 2013, pp. 233–236, URL: http://dl.acm.org/citation.cfm?id=2487085.2487132.

[21] GitHub, GitHub API, 2020, https://developer.github.com/v3/. URL: https://developer.github.com/v3/.

[22] J. Sheoran, K. Blincoe, E. Kalliamvakou, D. Damian, J. Ell, Understanding watchers on GitHub, in: Proceedings of the 11th Working Conference on Mining Software Repositories, 2014, pp. 336–339.

[23] GitHub, GitHub flavored markdown spec, 2020, https://github.github.com/gfm/. URL: https://github.github.com/gfm/.

[24] S. Ikeda, A. Ihara, R.G. Kula, K. Matsumoto, An empirical study of readme contents for javascript packages, IEICE Trans. Inform. Syst. 102 (2019) 280–288.

[25] A.B. Cantor, Sample-size calculations for Cohen's kappa, Psychol. Methods 1 (150) (1996).

[26] M. Honnibal, I. Montani, spaCy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing, 2017, (Unpublished software application, https://spacy.io).

[27] K. Blincoe, J. Sheoran, S. Goggins, E. Petakovic, D. Damian, Understanding the popular users: Following, affiliation influence and leadership on GitHub, Inf. Softw. Technol. 70 (2016) 30–39.

[28] H.B. Mann, D.R. Whitney, On a test of whether one of two random variables is stochastically larger than the other, Ann. Math. Statist. 18 (1947) 50–60, http://dx.doi.org/10.1214/aoms/1177730491.

[29] N. Cliff, Dominance statistics: Ordinal analyses to answer ordinal questions, Psychol. Bull. 114 (3) (1993) 494–509, http://dx.doi.org/10.1037/0033-2909.114.3.494.

[30] M. Hess, J. Kromrey, Robust confidence intervals for effect sizes: A comparative study of cohen's d and cliff's delta under non-normality and heterogeneous variances, in: Paper Presented At the Annual Meeting of the American Educational Research Association, 2004.

[31] E. Noei, F. Zhang, S. Wang, Y. Zou, Towards prioritizing user-related issue reports of mobile applications, Empir. Softw. Eng. 24 (2019) 1964–1996.

[32] H. Borges, M.T. Valente, What's in a GitHub star? Understanding repository starring practices in a social coding platform, J. Syst. Softw. 146 (2018) 112–129.

[33] G. Salton, A. Wong, C.S. Yang, A vector space model for automatic indexing, Commun. ACM 18 (1975) 613–620, http://dx.doi.org/10.1145/361219.361220, URL: http://doi.acm.org/10.1145/361219.361220.

[34] E. Noei, D.A. Da Costa, Y. Zou, Winning the app production rally, in: Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ACM, 2018, pp. 283–294.

[35] D. Birant, A. Kut, ST-DBSCAN: An algorithm for clustering spatial–temporal data, Data Knowl. Eng. 60 (2007) 208–221, http://dx.doi.org/10.1016/j.datak.2006.01.013.

[36] W.H. Kruskal, W.A. Wallis, Use of ranks in one-criterion variance analysis, J. Amer. Statist. Assoc. 47 (1952) 583–621, http://dx.doi.org/10.1080/01621459.1952.10483441, URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1952.10483441.

[37] O.J. Dunn, Multiple comparisons using rank sums, Technometrics 6 (1964) 241–252, http://dx.doi.org/10.1080/00401706.1964.10490181, URL: https://www.tandfonline.com/doi/abs/10.1080/00401706.1964.10490181.

[38] A. Trockman, S. Zhou, C. Kästner, B. Vasilescu, Adding sparkle to social coding: an empirical study of repository badges in the npm ecosystem, in: Proceedings of the 40th International Conference on Software Engineering, 2018, pp. 511–522.

[39] H. Hata, T. Todo, S. Onoue, K. Matsumoto, Characteristics of sustainable OSS projects: A theoretical and empirical study, in: 2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering, 2015, pp. 15–21, http://dx.doi.org/10.1109/CHASE.2015.9.

[40] F. Hassan, Xiaoyin Wang, Mining readme files to support automatic building of java projects in software repositories, in: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), 2017, pp. 277–279, http://dx.doi.org/10.1109/ICSE-C.2017.114.

[41] Y. Zhang, D. Lo, P.S. Kochhar, X. Xia, Q. Li, J. Sun, Detecting similar repositories on GitHub, in: 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2017, pp. 13–23, http://dx.doi.org/10.1109/SANER.2017.7884605.