Master's Thesis

ON GENDER DIVERSITY IN OPEN-SOURCE SOFTWARE PROJECTS

MIRABDULLA YUSIFLI

October 13, 2021

Advisor:
Christian Hechtl   Chair of Software Engineering

Examiners:

Prof. Dr. Sven Apel	Chair of Software Engineering
Prof. Dr. Martina Maggio	Chair of Software and Hardware Systems

Chair of Software Engineering
Saarland Informatics Campus
Saarland University

Erklärung

Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken,______________________ ______________________________
(Datum/Date) (Unterschrift/Signature)

ABSTRACT

With the advent of the information age, software engineering has become one of the most prominent occupations. However, marginalization of women still remains one of the biggest issues in this field. Biases against female developers in software development have been investigated in a variety of studies. In this study, we further explore this issue by studying differences between the outcomes of participation of male and female developers in open-source software (OSS) projects. After collecting data and analyzing nine open-source projects, we first classify developers according to gender using a name-to-gender inference tool. We then proceed to compare the two groups based on the number of created pull requests, created issues, merge operations, merged pull requests, pull request comments, issue comments, commits, changed files, changed lines, and the developer importance.

Our results suggest that there are small differences between male and female developers in the overall contribution process. Therefore, certain particularities can be pointed out. We observe that in certain projects male developers appear to be much more active than their female counterparts in issue-related contributions. However, we do not deem it to be true in general, since this difference can only be seen in 6 out of 9 projects. It was also the case that in 3 of the projects, all merge operations for pull requests were exclusively performed by male developers, which can indicate marginalization of female developers. When it comes to developer importance, i.e. the measure of "coreness" of a developer within a project, both genders follow a similar distribution over the coreness scale in the issue-related contributions, although in total there are much more male developers than female ones. Unlike the issue-related contributions, analysis of code contributions also reveals a partly similar distribution between the two groups. However, in both code and issue-related contributions, male developers constitute a large proportion of the top contributors.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

In the last decade, open-source software (OSS) projects have become increasingly popular thanks to fast development cycles and decentralized community structures. The popularity of OSS projects has been increased by individuals as well as companies. These projects are a large part of today's IT infrastructure and commercial companies are willing to be part of the OSS development [6] [2] [28]. Extensive usage of OSS projects makes these projects great samples to draw a general picture of the software development industry. Moreover, these projects bring people of varied cultural and educational backgrounds, gender, age, country, and expertise together. For example, students, professional programmers, people whose hobby is programming, and others can work together in an OSS project. According to a study by Alexander Hars [1], self-determination and human capital are the main motivations of these people to participate in OSS projects. Additionally, the research shows that students and hobbyists are self-motivated internally, while professional programmers are willing to gain profit by selling products and services.

Vasilescu, Filkov, and Serebrenik [33] conducted a research that shows diversity in a team brings new opinions, skills, and alternative solutions. For example, diversity in experience level has an advantage and a disadvantage. Communication with more junior developers increases understanding of own thoughts, while communication with more senior developers helps to learn new ropes. On the other hand, senior developers should spend additional time to prevent junior developers from making mistakes and help them to follow best practices. The paper also notes that nationality and language diversity may lead to communication problems in a team. Moreover, gender diversity has a negative impact on women. Some of them use fake GitHub names to be assumed as male. In addition to this research, also Terrell et al. [32] find that hiding one's gender in the GitHub profile affects the code acceptance rate positively, which means that a female profile whose gender information is invisible gets better acceptance rates than a female profile that shows gender information explicitly. Besides, a majority of code contributions are made by men, while women contribute in other ways such as leading and coordinating [27].

Ashcraft, McLain, and Eger [4] state that the percentage of computing-related occupations held by women declined from 36% to 25% between 1991 and 2015, although several researches [13] [34] note that gender-balanced teams are more productive and show better performance in comparison with teams dominated by men or women.

The information described above explains the importance of gender diversity in the IT industry. Big tech companies such as Google, Facebook, and Apple prepare annual reports to draw attention to diversity. On a global scale, the percentage of tech positions held by women in Google increased from 16.6% in 2014 to 23.6% in 2020 [1]. Similarly, the Facebook diversity report[2] demonstrates that 15% of tech positions were held by women in 2014 and this percentage reached 24.1% in 2020. Additionally, according to the Apple diversity report[3], the percentage of women who work in a tech position increased slightly from 20% to 24% between 2014 and 2020. As a result, we can say that the number of positions held by women have been increasing in recent years.

Our motivation to study this topic is understanding differences between genders in terms of contributions and developer importance. The research may help to answer how females are represented in OSS projects. Moreover, understanding differences between genders may help to reduce female marginalization and create gender-balanced teams.

For these reasons, the goal of this thesis is to find out the main differences between male and female developers in OSS projects. In this study, we do not assume anyone's gender to assign them to a specific gender group. We just work with binary genders assumed based on names. Simply, we take people's names and predict the probable genders of these names.

The projects we use for this research are Vscode, Tensorflow, Atom, Keras, Nextcloud, Vue, Three.js, Deno, and Reveal.js, for which we mine data from GitHub and their git repositories. The main data we use, are the list of commits, issues, and developers. Once we have mined the data, we use GenderAPI, a name-to-gender inference tool, to group developers into the male and female groups.

In the first part of the analysis, we compare genders using statistical tests to understand whether one gender predominates the other one in terms of the general contribution statistics which are created pull requests, created issues, merge operations, merged pull requests, pull request comments, issue comments, commits, changed files, and diff size.

In the second part of the analysis, we compare genders in terms of developer importance. For the comparison, we build three developer networks: co-change networks, issue networks, and co-change-issue networks. Co-change networks use the commit data as a base, while issue networks are based on the issue data. Co-change-issue networks use both commit and issue data to describe relationships between developers. After building networks, we calculate coreness values for each developer in each network. For the calculation, we use two different network metrics: eigenvector centrality and hierarchy. Then, we use the coreness values as a proxy for developer importance. In the last step, we compare male and female developers in terms of the coreness values with the help of statistical tests and plots.

1 `https://diversity.google/annual-report`, last accessed on 05/05/2021
2 `https://diversity.fb.com/read-report`, last accessed on 05/05/2021
3 `https://www.apple.com/diversity`, last accessed on 05/05/2021

Our results show that the number of male developers is more than that of female developers in all the projects. Although some projects show a statistically significant difference between genders in terms of the general contribution statistics, we can not accept or reject our hypotheses completely, since the results differ from OSS project to OSS project. Our results depict that all merge operations are performed by only male developers in some projects. In other words, there is no female developer that performs a merge operation. The next outcome of the study is that there is no statistically significant difference between genders in the code contribution process. On the other hand, the results of most projects evidence that male developers participate actively in the issue-related contributions. These results are similar to the results of the second part of the analysis in which we look at the differences in terms of developer importance which is explained by coreness values. The results of the second part lead us to conclude that the difference between genders in terms of coreness values is not statistically significant in code contributions, although male developers dominate the top of the organizational hierarchy. Differently from the code contributions, also female developers take some of the most important roles in issue-related contributions of all the projects, despite that male developers generally have a higher importance level in some projects. Moreover, both genders follow a similar coreness distribution, except in the upper parts of the scale, which means that gender matters only in the upper parts. Furthermore, we can see there are male developers in all parts of the coreness scale in all types of networks. However, there are no female developers in some parts of the scale, usually upper parts. Until the 90th percentile of the coreness scale, upper parts have fewer male developers in comparison with lower parts. This pattern exists for only male developers in the co-change networks, for both genders in the issue and co-change-issue networks.

The rest of this thesis is structured as follows: In Chapter 2, we give an overview of the topics that are relevant to this thesis, such as OSS development, developer networks, and related work. Chapter 3 is about our hypotheses and details of the approach we use to evaluate these hypotheses. In Chapter 4, we present the outcomes of the analysis and discuss them. Moreover, we introduce threats that may have influences on our results. In the final chapter, Chapter 5, we outline the most important points of this thesis and give some ideas about future work.

2 BACKGROUND

In this chapter, we cover topics that are relevant to the thesis. It gives an insight into open-source software (OSS) development and developer networks. Moreover, we provide an overview of related work.

2.1 OPEN-SOURCE SOFTWARE DEVELOPMENT

An OSS project is a software published on the internet under licenses that gives users certain rights except for private property rights. OSS development is a production model to develop OSS projects in a way that exploits the distributed intelligence of participants in online communities [17]. It constitutes a significant part of global software development. It allows people not only to access and use feature-rich software for free, but also permits them to view the source code of the software and modify it according to their needs. In this way, some projects create large communities by engaging the attention of thousands of developers. Moreover, OSS projects give an opportunity to developers to interact with more experienced developers. Thus, developers may feel motivated to participate in OSS projects to gain experience and learn from skilled developers [36].

While OSS is publicly available and managed by the community, proprietary software is not publicly available and can be modified by only a company or team which has created it. OSS has some advantages and disadvantages over proprietary software. The main advantage is that OSS is completely free and accessible to everyone. Moreover, OSS projects are supported and maintained by large communities. Therefore, they are much faster to fix bugs and provide support to users. The main disadvantage is that competitors can develop similar products easily because of publicly available source code.

Many commercial companies tend to develop and release OSS products [28]. Commercial companies contribute to OSS projects because of three different motivational factors which are building great innovative products better and faster, gaining profit by selling complementary services such as training, technical support, consultancy and certifications, and cost reduction because of large community support [2]. Moreover, Lakhani and Wolf [18] show that about 40% of the developers get money from their employers to contribute to free and open-source software development.

2.1.1 *GitHub*

GitHub is a website that allows developers to share code, communicate, and collaborate for developing large-scale software. It uses the version control system GIT to manage projects and keep track of source code history. GitHub is one of the most popular software development platforms, which hosts over 100 million repositories and 56 million developers [1]. Furthermore, social coding platforms, such as GitHub, creates an opportunity for developers to demonstrate their skills and experiences [9]. Hence, employers consider GitHub profiles as portfolio of work in the hiring process [11].

2.1.2 *Contribution Process*

Since OSS projects are usually developed with developers from all around the world, a substantial number of projects have many contributors. For this reason, some tools are used to regulate the code and establish a communication protocol. Mainly, two main approaches are used: *pull-request-based* and *patch-based* [39]. In this thesis, we analyze the projects that use the pull-request-based approach. Hence we explain only the pull-request-based approach. However, there are several pull-request-based services, we focus on GitHub.

Once a developer aims to contribute to a repository, the repository of the project should be copied to the personal GitHub account. This process is known as *forking*. Then, the created copy is cloned to the local machine. This step is called *cloning*. Once the repository is cloned, the necessary changes are made to this copy. Afterward, all changes are added to a commit and the created commit is published on the local repository. The last two steps can be repeated until the required feature or changes are implemented. Following implementation, a pull request that contains all commits is created from the local repository against the official repository. Thereafter, other developers review the pull request to either add some comments to request some changes or reject the pull request if it is completely unnecessary. Subsequently, the creator of the pull request can publish new commits that contain required changes until the pull request gets approved and merged into the official repository by a developer who has the necessary rights. It is also possible that reviewers reject and close the pull request if it is completely wrong or unacceptable [38] [37]. The general workflow of the code contribution process is shown in Figure 2.1.

[htbp]

The contribution process is not only about addition of code to the repository. People can contribute to the project by reporting bugs, giving ideas about new features, reviewing code, providing user support, and participating in the discussions. As a consequence of developing software globally, organizing a meeting

1 https://github.com, last accessed on 06/02/2021

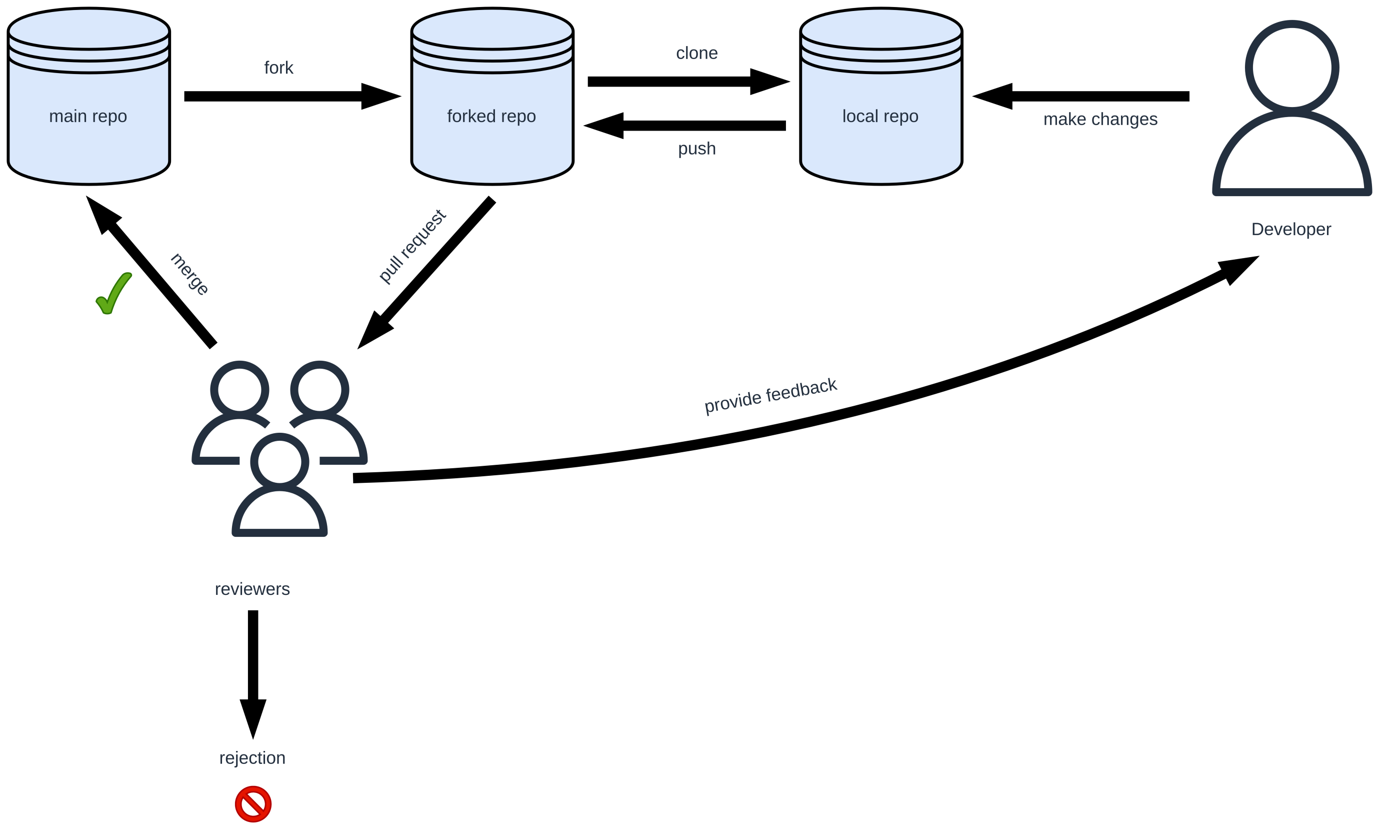


Figure 2.1: GitHub contribution process. A developer changes code and sends a pull request to reviewers. They either merge it into the official repository, request changes or reject the pull request completely.

with contributors is quite difficult. To make communication easier, GitHub has a section called *issues*. This section helps people to make discussions about bugs, new features and get help. A contributor simply creates an issue to discuss a related topic and others can add comments to the issue. Additionally, a pull request can be linked to an issue to show that the necessary code changes for the issue has been sent by this pull request. After merging the linked pull request, the issue is closed automatically. Moreover, an issue can be assigned to multiple people who are responsible for moving the issue forward and fixing it. Another important point is that, GitHub returns both issues and pull requests in search results. GitHub use *type* qualifier to distinguish pull requests and issues. Simply, a pull request is an issue whose type is *pull request*, while an issue is an issue whose type is *issue*.

2.2 DEVELOPER ROLES

In OSS projects, developers are separated into several groups depending on their roles in the development. Nakakoji et al. [24] point out eight different roles ranging from *passive user* to *project leader*. The layered structure of the roles is shown in Figure 2.2. These roles are not connected to any attributes such as age or title. Accordingly, roles are not assigned by someone else. Instead, a role is gained as a result of contributions and activities.

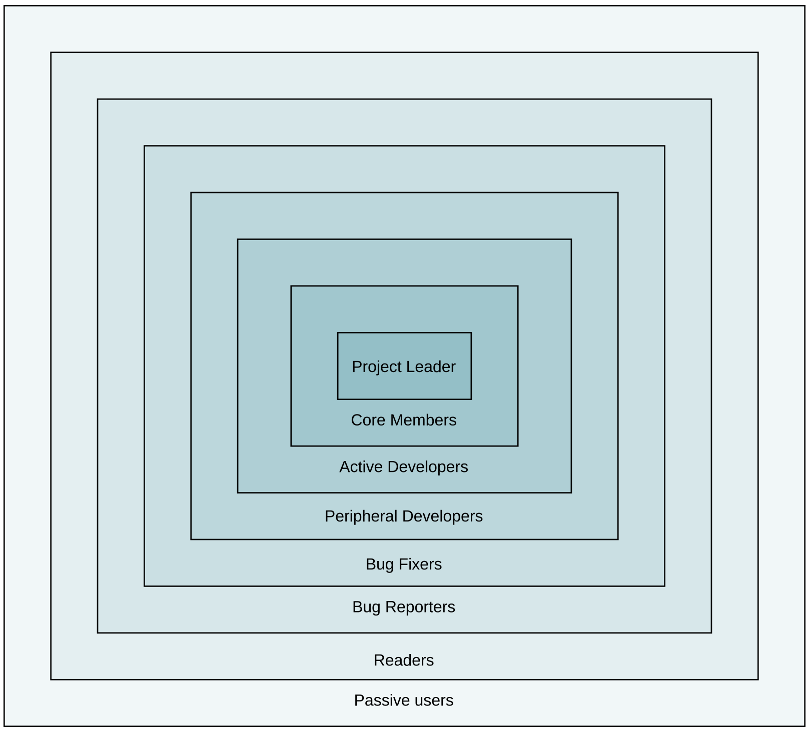


Figure 2.2: General structure of the roles in OSS projects. The darkness of blue changes in proportion to the impact of the role. The role in the darkest rectangle has the largest impact.

All groups make contributions to the development of the project, except *readers* and *passive users*. The group which contributes the least is *bug reporters*. Though they do not participate in the code contribution, they test software and report bugs. Most of the reported bugs are fixed by *bug fixers*. The next contributing group is *peripheral developers* who participate sporadically in the development of new features. Additionally, *active developers* group is one of the main groups in OSS development. They are involved actively in the development of the project and bug fixing. One of two groups that play an important role in decision-making is *core members* whose members have participated in the development process for a long time and have extensive knowledge of the software. Another group is *project leader* that usually has only one member. This member is a person who started the project and is responsible for the overall direction of the project. As shown in Figure 2.2, a core member has a larger impact than an active developer whose impact is larger than that of a peripheral developer, and so on. We group active developers, peripheral developers, bug fixers, and bug reporters together and consider all of them as peripheral developers. Therefore, the developers who contribute to the code base are coarsely represented as two groups: *core developers* and *peripheral developers* [7] [10] [15] [30].

2.3 DEVELOPER GENDERS

There are several techniques to detect people's gender: face-to-gender inference [25] [23] [20], name-to-gender inference [19] [22], and gender detection from writing style [3]. We use the name-to-gender technique, in which, gender is predicted from person's name. Santamaría and Mihaljević [29] compare five name-to-gender inference tools which are GenderAPI, NameAPI, genderize.io, gender-guesser, and NamSor, in terms of different performance metrics. Since the comparison deduces that GenderAPI shows the best performance among these services, we use GenderAPI in this research. The tool provides an API to access it with different programming languages. Simply, a GET request that contains name, should be sent to the API in order to retrieve the gender. Additionally, it is possible to upload a list of names to determine their genders using the website[2]. In both ways, before the determination of gender, GenderAPI parses the full name into first name and last name. As a response, it returns one of the gender labels which are *male*, *female*, and *unknown*, as well as the confidence parameters *samples* and *accuracy*. The *samples* shows the number of matched database records, while *accuracy* indicates the reliability of the estimated gender.

2 https://gender-api.com, last accessed on 07/06/2021

2.4 DEVELOPER NETWORKS

Another important part of this thesis is building social-network graphs for developers. Social networks are used in many different fields to describe relationships between people [8]. Vertices represent people and edges represent relationships between people. Since we analyze relationships between developers, we call these graphs *developer networks* [21].

Such networks can be directed or undirected. Directed edges are used to represent asymmetric relationships, while undirected edges define symmetric relationships [8]. For example, if a developer reviews another developer's pull request, it is an asymmetric relationship. Since this relationship has direction: from reviewer to the owner of the pull request. On the other hand, if two developers make changes to the same file, it is a symmetric relationship. Since the actions performed by developers are independent of each other.

The same asymmetric relationship may exist multiple times between the same developers. These relationships can be unified into one relationship to be represented by one edge instead of multiple edges that have the same direction. This process decreases the number of edges and increases simplicity [8].

It is possible to build different networks on the basis of relationships between developers. We build three types of developer networks for this research: *co-change*, *issue*, and *co-change-issue*. A co-change network connects developers who change the same source code artifact, while an issue network connects developers who contribute to the same issue. A co-change-issue network is the combination of both. It connects developers who contribute to the same source code artifact as well as developers who contribute to the same issue.

2.5 NETWORK-BASED METRICS

After building developer networks, the developers can be classified into the core or peripheral groups, as described in Section 2.2. To do that, we need some metrics to assign values to each developer in the network. There are different metrics for that. For this research, we use only two of them: *eigenvector centrality* and *hierarchy*.

Eigenvector centrality metric describes the centrality of a developer in the network. Since more important developers play more important roles in code contribution and discussions, we expect that they have more connections with others and take central positions in the network. Therefore, we assume that the eigenvector centrality metric correlates with the importance of developers [15]. Eigenvector centrality is determined by the number of connections. It is the sum of eigenvector centralities of a vertex's neighbors. Figure 2.3 illustrates eigenvector centrality in a small network. It is calculated with the following formula:

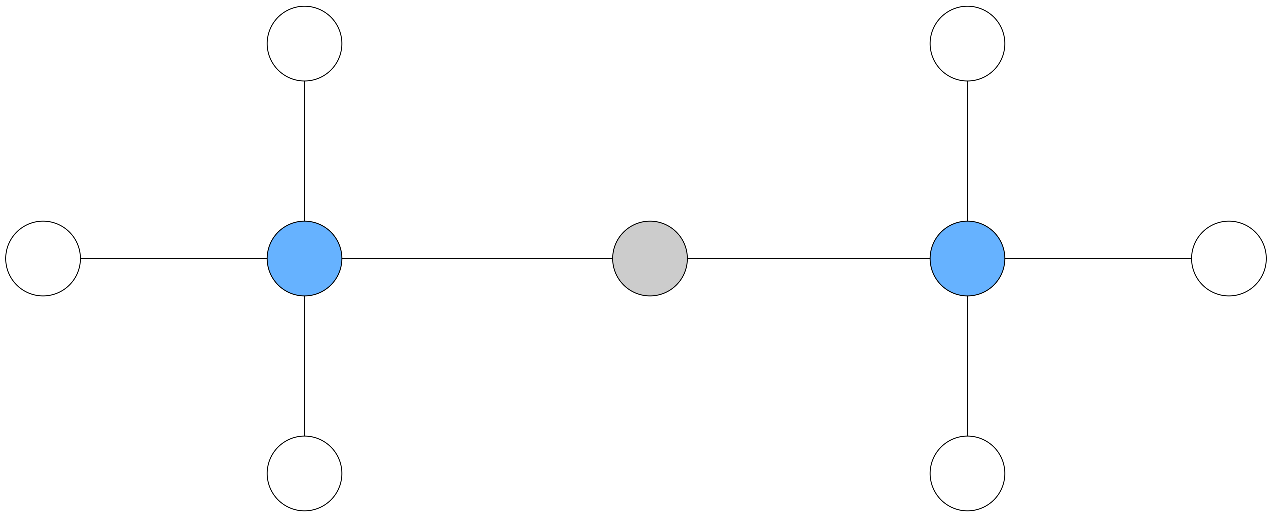


Figure 2.3: A network comprising nine vertices and eight edges. The gray vertex has the highest eigenvector centrality value, since it is connected to two vertices that are connected to three other vertices. The blue vertices have the second highest eigenvector centrality values. Since they have connections with three vertices that have one connection and one vertex that has two connections.

$$x_i = \frac{1}{\lambda} \sum_{j \in N(i)} x_j \tag{2.1}$$

x_i is eigenvector centrality for vertex i, *N(i)* is the collection of neighbors and λ is a proportionality constant [14].

Hierarchy metric allows us to split the network into hierarchical groups and explain stratification between these groups. These groups are called clusters. The stratification depends on the node clustering coefficient and node degree that represents the number of connections [16]. We show the difference between a random network and a hierarchically organized network in Figure 2.4.

The clustering coefficient is a metric that indicates modularity and is based on neighborhood connectivity. It gives us a quantitative measure that explains what to extent a vertex is embedded in a cluster. The clustering coefficient c_i for a vertex i is calculated with the following formula:

$$c_i = \frac{2n_i}{k_i(k_i - 1)} \tag{2.2}$$

k_i is the number of neighbors of the vertex i, n_i is the number of edges between neighbors. The clustering coefficient is equal to the fraction of existing edges between neighbors divided by the total number of possible edges between neighbors.

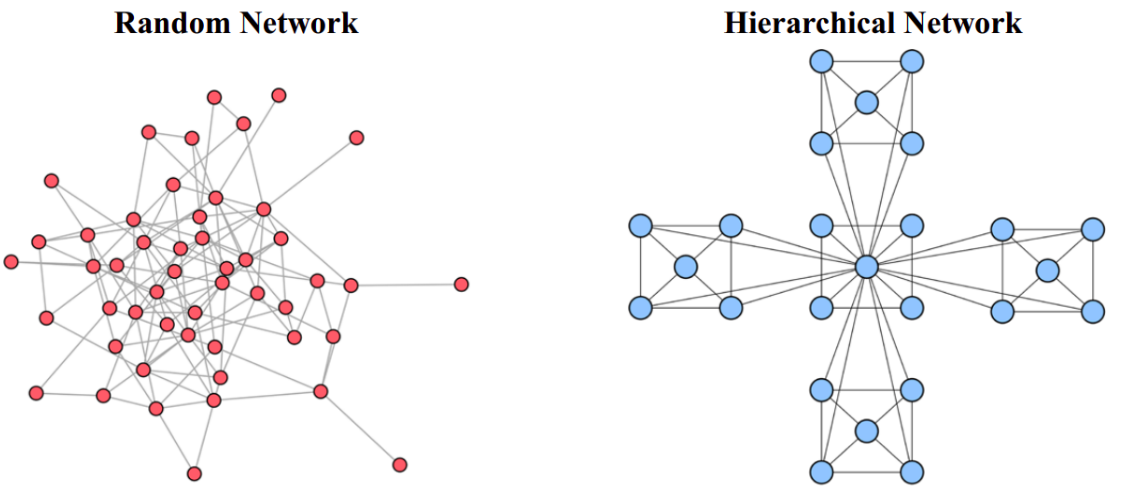


Figure 2.4: This figure is taken from [16] Fig. 4. The figure describes a random network on the left side and a hierarchical network on the right side. In the hierarchical network, clusters create a clustered network by connecting to each other. The top vertex of the hierarchy is the one in the middle because it is connected to vertices of all other clusters, not only vertices in its small cluster.

$k_i(k_i - 1)/2$ is the number of possible edges that can exist between neighbors. A high clustering coefficient means that there are many edges between the neighbors, while a low clustering coefficient indicates the existence of fewer edges between neighbors of the vertex [14] [16]. The variables k_i, n_i, and the calculated clustering coefficient c for a vertex in two different networks are shown in Figure 2.5.

The degree of nodes decreases from the top of the hierarchy to the bottom of the hierarchy, while the clustering coefficient increases. In other words, vertices that are located at the top of the hierarchy have a high node degree and a low clustering coefficient. In a hierarchical network, core developers are located at the top of the hierarchy, while peripheral developers are placed at the bottom of the hierarchy [14] [16].

2.6 CORE PERIPHERY DETECTION

The metrics described in the preceding section can be used to classify developers as core or peripheral in the social networks. In the other words, each developer gets an assigned value based on these metrics and developers are classified with the help of assigned values.

One of the most prevalent approaches is to classify developers based on their commit count. Specifically, the number of commits made by each developer is calculated and a threshold is defined at 80th percentile. Developers whose commit counts are above the threshold are included in the core group, while developers with commit counts below the threshold are included in the peripheral group.

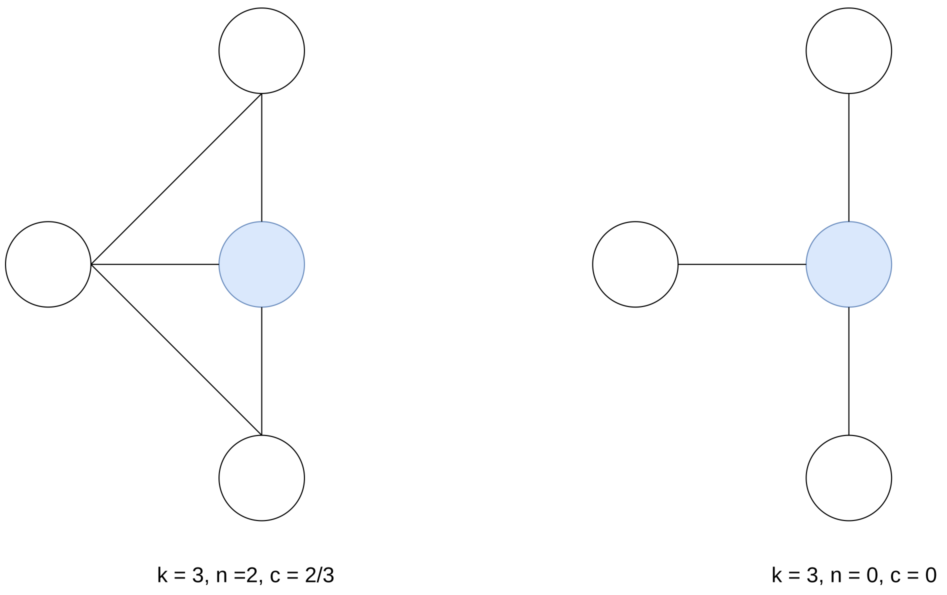


Figure 2.5: The variables k_i and n_i as well as the calculated clustering coefficient c for the blue vertex in two different networks. In both networks, k is 3 because the vertex has three neighbors. The network on the left has two edges between the neighbors of the blue vertex therefore n is . The clustering coefficient is 0 for the network on the right side since there is no edges between neighbors.

Since the number of commits has a Zipf distribution, we expect around 80% of the contributions to be made by 20% of the developers [10] [26] [31]. In our research, we use values from the network metrics above instead of commit count.

2.7 CORENESS

After the construction of networks, the method explained in Section 2.6 can be used to classify developers into core and peripheral groups. However, we do not group developers, since the research by Fedorov, Mannino, and Zhang [12] reveals that conversion of a continuous outcome to a binary outcome can lead to information loss which is a problem in terms of hypothesis testing and statistical estimation. Therefore, we compare developers based on their metric values which we get by applying the network metrics described in Section 2.5 instead of the groups they belong to. From now on, we call these values *coreness values*. Simply, coreness values are calculated using these network metrics and serve as a proxy value for the importance of each developer in the network.

2.8 RELATED WORK

There are many studies that deal with the diversity in OSS projects. In this part, we present how other researchers analyze gender inequality in OSS projects. Vedres and Vasarhelyi [35] compare the importance of gendered behavior and categorical gender for *success* and *survival*. Success is represented by the number of repository stars, while survival indicates whether the user performed any action in a year. Gendered behavior explains the relationship between gender and behavior. In this research, gendered behavior is evaluated as the probability of femaleness of behavior. The femaleness of behavior indicates the strength of the relationship between females and the behavior. If it is high, females generally tend to demonstrate the behavior, otherwise, it is seen very rarely. A Random Forest model is applied to measure femaleness of behavior based on activity, partners, and specializations in programming languages classified for gender using principal component analysis. In more detail, Vedres and Vasarhelyi look at statistics such as how many repositories each person has, how many pull requests have been opened by each person, in which field each person specializes, genders of people each person follows, genders of collaborators, and etc. GitHub is the main data source. Github API and githubarchive.org are used to acquire information about repositories, developers, and their activities. In addition, the 2016 US baby name dataset is used for name-based gender detection. As a result, the difference between men and women is found not to be statistically significant (8.76 stars for women and 13.26 stars for men on average) in terms of success if gender is considered as a category (men and women). Moreover, 88.2% of women and 92.8% of men are active after a year. According to the analysis, 84.5% of the women's disadvantages in success and 34.8% of disadvantages in survival can be explained by femaleness of behavior. In summary, the research shows that gendered behavior has a stronger relationship with success than categorical gender. In other words, women's disadvantages are because of their activities, not their actual gender.

Another related topic is gender bias in OSS projects. Terrell et al. [32] explore the difference between genders in terms of the acceptance rate of pull requests and investigate reasons for the difference. The GHTorrent dataset that contains data about pull requests, users, and projects is the main data of the research. Data about pull requests such as status, description, and comments are extracted from GitHub in order to expand the GHTorrent dataset. For gender inference, GitHub accounts are linked with Google+ social network accounts where users can provide gender information. The research reveals that 78,7% (111,011 out of 141,177) of the pull requests created by women and 74.6% (2,181,517 out of 2,923,550) of the pull requests created by men are merged. Contrary to expectations, the merge rate of women's pull requests is higher. They find that 12.4% of women's pull requests and 13.2% of men's pull requests refer to issues. A pull request referencing an issue is considered as a needed contribution. Additionally, they analyze pull requests in terms of the number of lines added and removed, changed files, and commits. The result shows that women's pull requests contain more changed lines and commits. Furthermore, the study emphasizes that women maintain a higher acceptance rate for almost all programming languages. Another remarkable point is that women's acceptance rate is lower than men's when they are neither owners nor collaborators of the project and gender is identifiable.

Since we also look at the developer roles in this thesis, another related study is a study conducted by Bosu and Carver [7]. The study highlights the influence of developer roles on the code review process. To understand the influence, they explore some parameters such as *first feedback interval*, *review interval*, *code acceptance rate*, and *the number of patch revisions*. The first feedback interval is defined as a space of time between the submission of a code review request and the first comment written by a reviewer. The review interval is the elapsed time from submission of a code review request until the end of the review process. The code acceptance rate is the ratio of merged requests to all requests sent by a developer. The number of patch revisions is the number of patchsets sent until all modifications requested by the reviewer are implemented and the code is merged successfully. To understand the relationship between developer roles and these variables, data of eight popular OSS projects is extracted from Gerrit [3]. After cleaning the data, undirected social network graphs are generated. Subsequently, developers are divided into two groups: *core* and *peripheral*. Finally, non-parametric hypothesis tests are used to understand the relationship between developer roles and the variables. The result shows that the first feedback interval of core developers is shorter than the that of peripheral developers. Moreover, the acceptance rate is higher for core developers compared to peripheral developers. Furhtermore, the difference between developer groups in terms of the number of patch revisions is not strong. The last finding is that the review process takes 2 - 19 times longer for peripheral developers in comparison with core developers.

Another paper that deals with this research is written by Bird et al. [5]. They mined mailing list archives and source code repositories to construct a social network of the participants and understand communication between individuals. The research shows that sub-communities emerge within projects as long as the projects improve. Additionally, the people within a sub-community communicate more intensively with the people in their own sub-community in comparison with the people from different sub-communities. Moreover, social networks constructed from discussions directly related to the source code, are more modular. Other topics such as high-level architecture, licensing, policy decisions affect everyone. It means that everyone should participate. Therefore, social networks constructed from discussions about these topics, are less modular. The last relevant finding is that developers within the same sub-communities have higher collaboration levels in comparison with developers outside. i.e, work in the same areas of the code. In other words, the community structure of the social networks is connected with the actual development effort.

3 https://www.gerritcodereview.com

3 APPROACH AND HYPOTHESES

This chapter is about our research approach. It introduces our hypotheses and projects. Moreover, this chapter contains the pertinent details of gender detection, network construction, and coreness detection. The final part of this chapter covers details of our approach to investigate the hypotheses.

3.1 RESEARCH QUESTIONS AND HYPOTHESES

In this thesis, we investigate the differences between male and female developers. For this reason, we try to answer the following research questions:

Research question 1: *Are there differences between male and female developers in terms of general contribution statistics in open-source software projects?*

Research question 2: *Are there differences between male and female developers in terms of developer coreness in open-source software projects?*

As shown in the research questions, our comparison consists of two parts: *general contribution statistics* and *developer coreness*. We present each part in the following sections.

3.1.1 *Comparison of General Contribution Statistics*

In this part, we compare genders from the viewpoint of the following parameters: *created pull requests*, *created issues*, *merge operations*, *merged pull requests*, *pull request comments*, *issue comments*, *commits*, *changed files*, and *diff size*. We present two different approaches for the comparison.

In the first approach, we hypothesize male developers have higher statistics than female developers. We show the hypotheses of this approach in Table 3.1. We call this approach *male approach* from now on. The hypotheses whose labels start with *H.M* belong to this approach.

In the alternative approach, we hypothesize that female developers have higher statistics than male developers. We list the hypotheses of this approach in Table 3.2. We call this approach *female approach* from now on. The hypotheses of this approach start with *H.F* label.

Table 3.1: The hypotheses of the male approach

H.M1 :	Male developers create more pull requests than female developers.
H.M2 :	Male developers create more issues than female developers.
H.M3 :	Male developers perform more merge operations than female developers.
H.M4 :	Male developers have more merged pull requests than female developers.
H.M5 :	Male developers add more comments to pull requests than female developers.
H.M6 :	Male developers add more comments to issues than female developers.
H.M7 :	Male developers create more commits than female developers.
H.M8 :	Male developers change more files than female developers.
H.M9 :	Male developers change more lines than female developers.

Table 3.2: The hypotheses of the female approach

H.F1 :	Female developers create more pull requests than male developers.
H.F2 :	Female developers create more issues than male developers.
H.F3 :	Female developers perform more merge operations than male developers.
H.F4 :	Female developers have more merged pull requests than male developers.
H.F5 :	Female developers add more comments to pull requests than male developers.
H.F6 :	Female developers add more comments to issues than male developers.
H.F7 :	Female developers create more commits than male developers.
H.F8 :	Female developers change more files than male developers.
H.F9 :	Female developers change more lines than male developers.

3.1.2 *Comparison of Developer Coreness*

In this section, we look at the differences between male and female developers in terms of developer coreness explained in Section 2.7. Firstly, we build networks explained in Section 2.4 and calculate both coreness values for each developer using the developer metrics described in Section 2.5. After assigning values to each developer, we try to find out whether there are statistically significant differences between the genders in terms of the coreness values. To answer this question, we use the same two approaches as described in Section 3.1.1. Therefore, we pose a hypothesis for each approach.

Table 3.3: The hypotheses for the comparison in terms of coreness values

H.M10 :	Male developers have higher coreness values than female developers.
H.F10 :	Female developers have higher coreness values than male developers.

In addition to the investigation of these hypotheses, we analyze the distribution of coreness values for each gender. To evaluate *H11* described in Table 3.4, we look at the distribution shape of coreness values for each gender. Moreover, we split the coreness scale into different parts to look at what percentage of each gender constitutes each part of the scale. Then we try to evaluate *H12* described in the following table:

Table 3.4: The hypotheses for the comparison in terms of distribution of the coreness values

H11 :	The coreness values of both genders follow a similar distribution.
H12 :	Both genders are shared equally within different parts of the coreness scale.

3.2 approach

To investigate hypotheses and questions listed in the previous section, we perform few steps such as data collection, gender detection, network construction, and calculation of coreness values. We present all the details of the approach in the following subsections.

3.2.1 *Case studies*

We choose nine OSS projects to evaluate our hypotheses. The first project is VsCode[1], which is an integrated development environment (IDE) made by Microsoft. The second project is Tensorflow[2] which allows developers and researchers to develop applications using machine learning techniques. Atom[3] is the third project. It is a text editor that is mainly used by developers. The next project is a deep learning API called Keras[4]. The fifth project, Nextcloud[5], is a file share and collaboration platform. The last four projects are related to the JavaScript environment. Deno[6] allows creating an environment to run JavaScript and TypeScript codes. The other three projects, Vue[7], Three.js[8], and Reveal.js[9] are JavaScript tools to create modern, and interactive user interfaces. We perform our analyses based on the data collected from these projects.

3.2.2 *Data Extraction*

To analyze the projects, we need to extract data. For this research, we need three types of data from each project: *authors*, *commits*, and *issues*. For data extraction, we use codeface [10], a tool developed by Siemens, and an extension to it called codeface-extraction[11]. This tool uses *author* keyword to represent a developer. The commit data is extracted directly from the git repository of each project, while the issue data is mined from GitHub. The developer data is extracted from both sources. After that we have some post-processing on the data to unify commit and issue data with the developer data.

Once the data is mined, it is saved into a MySQL database by codeface. After that, data is extracted into CSV files using codeface-extraction. As a result, we have three CSV files: authors.list, commits.list, and issues.list.

Before starting to analyze, we cut the commit and issue data to the same date range to be sure that both data represent the same time frame of the project lifetime.

We show an overview of the extracted data for each project through Table 3.5. One can see that different size projects are analyzed. Vscode, Tensorflow, and Atom can be considered as large projects, while Keras and Nexcloud can be categorized

1 https://code.visualstudio.com/
2 https://www.tensorflow.org/
3 https://atom.io/
4 https://keras.io/
5 https://nextcloud.com/
6 https://deno.land/
7 https://vuejs.org/
8 https://threejs.org/
9 https://revealjs.com/
10 https://github.com/siemens/codeface
11 https://github.com/se-sic/codeface-extraction

Table 3.5: The number of the extracted developers, commits, and issues for each project and the time frames the data are cut to

Project	# developers	# commits	# issues	time frame
Vscode	68,675	68,350	111,126	2015-11-13 / 2020-12-22
Tensorflow	36,848	92,432	45,664	2015-11-09 / 2020-12-22
Atom	21,402	32,402	21,163	2012-01-21 / 2020-12-10
Keras	13,604	4,626	13,512	2015-03-28 / 2019-11-06
Nextcloud	10,139	16,228	22,726	2016-03-23 / 2020-09-22
Vue	9,869	3,124	9,351	2016-04-11 / 2020-11-24
Three.js	8,623	27,201	20,856	2010-04-25 / 2020-12-22
Deno	3,198	4,805	8,762	2018-05-29 / 2020-12-22
Reveal.js	3,047	2,242	2,769	2011-06-07 / 2020-10-12

as mid-size projects. In comparison with these projects, VUE, THREE.JS, DENO, and REVEAL.JS are small projects.

3.2.3 *Gender Detection*

We upload a CSV file that contains a list of developers' full names for a project to the website of GENDERAPI. As a response, the website gives us a CSV file that contains six columns: *full name*, *first name*, *last name*, *gender*, *accuracy*, and *samples*. We remove all columns except the *full name* and *gender* columns, since we need only these two columns for the analyses. GENDERAPI leaves the *gender* column empty for names that can not be classified as male or female. Hereby, we add *unknown* label to the *gender* column for these names. We perform this process for all projects. Consequently, we have a CSV file called *gender* for each project. Finally, we merge this gender data into the developer data to classify developers according to each gender. As a result, gender column is added to the developer data. We show the number of genders in each project in Table 3.6.

3.2.4 *Network Construction*

After the data extraction, we construct three developer networks explained in Section 2.4. We use a R library called CORONET [12] to build developer networks. The library allows us to build networks based on different data sources such as commits, issues or e-mails. We use only commits and issues to build developer networks. The

12 https://github.com/se-sic/coronet

Table 3.6: The number of genders in each project

Project	# males	# females	# unknown
Vscode	41,126	2,950	19,599
Tensorflow	22,098	2,319	12,431
Atom	15,750	1,009	4,643
Keras	8,268	886	4,450
Nextcloud	5,751	356	4,032
Vue	6,249	469	3,151
Three.js	5,521	425	2,677
Deno	2,195	152	851
Reveal.js	2,268	162	617

library has two different configuration classes: *ProjectConf* and *NetworkConf*. The ProjectConf class holds configuration parameters such as data paths and project name which are necessary for the configured project. The second configuration class, NetworkConf, is used to define configuration parameters related to data retrieval and network construction. The main parameters of these classes are author relation which describes relations among developers, directedness of edges, network simplifying achieved by edge contraction, and developer elimination depending on the chosen author relation.

For this thesis, we create non-simplified directed developer networks. The network type is defined by the author relation parameter. The author relation parameter should be set to *"cochange"* for co-change networks, *"issue"* for issue networks, and *c("cochange", "issue")* for co-change-issue networks. All type of networks contain only developers remaining after elimination depending on the network type. The remaining developers are developers that have been active in the associated data source. For example, only developers that have made at least one commit appear in co-change networks, whereas issue networks contain only developers who have contributed to at least one issue. Since co-change-issue networks are considered as a combination of both networks, they contain developers that have committed a code or contributed to at least one issue.

3.2.5 *Coreness Detection*

We use eigenvector centrality and hierarchy metrics to determine the coreness values of the developers. As described in Section 2.5, each metric is represented as a numerical value. The value range of the eigenvector centrality metric is between 0 and 1, while the value of the hierarchy metric can be greater than 1. Therefore, we use a normalization technique to shift and rescale the values of the hierarchy metric to fit the values within the interval of 0 to 1. So that, the ranges of both metrics end up between 0 and 1.

To detect the coreness values, we construct networks as explained in the preceding subsection. Then, we calculate the coreness values of each developer in all networks based on each network metric. After that, we use these coreness values to analyze the hypotheses listed in Table 3.3 and Table 3.4.

3.2.6 *Approach for the Comparison of the General Contribution Statistics*

Having done all the steps explained above, we can analyze the hypotheses. In this section, we present the details of our approach for evaluation. We use R-SCRIPTS for the analysis.

To analyze genders in terms of general contribution statistics, we evaluate the hypotheses listed in Section 3.1.1. As can be seen in Table 3.1 and Table 3.2, each hypothesis is related to a different parameter such as pull request, issue, and merge operation. For example, to compare genders in terms of the created pull requests, we calculate the number of created pull requests for each developer. Then we compare male and female developers using the *Wilcoxon Mann-Whitney* test to evaluate the related hypothesis. For a successful comparison, there must be data for both groups, since it is not possible to compare groups if one of them is empty. We run the test twice with different arguments to investigate both approaches: male approach and female approach. For the analysis of all hypotheses described in Section 3.1.1, these sequential steps are performed for each parameter separately: created pull requests, issues, merge operations, merged pull requests, pull request comments, issue comments, commits, changed files, and changed lines.

3.2.7 *Approach for the Comparsion of Developer Coreness*

As described in Section 2.7, developer importance is explained by coreness values. We compare genders in terms of coreness values in three steps. The first step is the investigation of the hypotheses described in Table 3.3. To do so, we use a similar approach as the one used for the analysis of the general contribution statistics. After detection coreness values as explained in Section 3.2.5, we split the developers into the male and female groups. Then, we use the Wilcoxon Mann-Whitney test to compare these groups in terms of the coreness values. We do these steps for both metrics to analyze the hypotheses listed in Table 3.3. Like the analysis of the general contribution statistics, we run the test twice for each metric to investigate the male and female approaches.

In the second step, we compare the distribution of coreness values to evaluate *H11* described in Table 3.4. For the comparison, we create QQ plots and boxplots for both genders. QQ plots show the distribution shape of coreness values, while boxplots give us a good indication of how the values are spread out. We create four QQ plots and four boxplots for each metric to look at the different parts of the scale. The first one shows the distribution of coreness values over the full scale. The other three plots show only values that exceed specific percentiles of the coreness scale: *80th percentile*, *90th percentile*, and *95th percentile*. As a result, the QQ plots give us information about the distribution shape of the coreness values for both genders. Additionally, we compare distribution of the coreness values of each gender with the help of boxplots.

In the final step, we evaluate the hypothesis *H12* described in Table 3.4. To do so, we split the coreness scale into different ranges: *0% - 50%*, *51% - 75%*, *76% - 90%*, *91% - 95%*, and *96% - 100%*. Then, we find the number of males and females in each range. Lastly, we present the number of males and females as percentages. As a result, we can see what percentage of each gender falls within the certain range of the coreness scale.

4 EVALUATION

In this chapter, we describe the results of our analysis and discuss them in detail. We summarize the most important results of the analysis. The rest of the results for the co-change and issue networks can be found in Appendix A. We do not show the results of the co-change-issue networks, since these networks are the combination of the co-change and issue networks and the results of the co-change-issue networks are quite similar to the results of the issue networks. Lastly, we discuss all the results.

4.1 RESULTS

The following subsections outline the results of our analysis. Since we have a large amount of results, we only present results of the selected cases and representative figures.

4.1.1 *General Contribution Statistics*

We explain how we compare male and female developers in terms of the general contribution statistics in Section 3.2.6. In this subsection, we show the results of both approaches of this analysis. Since we use the Wilcoxon Mann-Whitney test, we consider the statistical significance level *p-value* as an indication of significant result.

Table 4.1 contains the p-values of the male approach. The first noticeable result is that Vue, Three.js, and Reveal.js have no results for the merge operations. The reason is that all merge operations have been performed by male developers in these projects. Therefore, we have no data on the female groups. Hence, we can not perform the Wilcoxon Mann-Whitney test. Despite the fact that we have many results, only some show statistical significance with a p-value below 0.05. We can say that the difference between male and female developers in terms of the created issues is statistically significant in the Atom, Deno, and Reveal.js projects. Moreover, the results of VsCode, Atom, Keras, Three.js, and Deno.js exhibit p-values of less than 0.05 for the issue comments, as shown in Table 4.1. Additionally, the statistically significant difference between genders according to the commits exists only in the Three.js project. Since the results of some projects are statistically significant and that of some are not significant, we cannot generalize the results.

In conclusion, the results for the hypotheses listed in Table 3.1 are overall **inconclusive**, since the results of certain projects support the hypotheses, while the results of other projects do not support the hypotheses.

The results of the female approach are shown in Table 4.2. In this approach, only TENSORFLOW shows significant results. According to the results for TENSORFLOW, there are significant differences between male and female developers in terms of the created pull requests, merged pull requests, pull request comments, commits, changed files, and diff size. In other words, females create more pull requests and more commits, write more comments on pull requests, change more files and more lines, and most of the merged pull requests belong to female developers. This indicates that females participate more actively in this project. Because of the reason explained above, we can not perform tests for merge operations of the same projects in the female approach, too.

Since only the results for TENSORFLOW support some of the hypotheses listed in Table 3.2, we have to conclude that the results for the hypotheses overall are **inconclusive**.

4.1.2 *Developer Coreness*

As explained in Section 3.2.7, the comparison of male and female developers in terms of developer coreness consists of three steps.

Table 4.3 and Table 4.4 contain the results of the first step in which we evaluate the hypotheses listed in Table 3.3. After performing the steps described in the first paragraph of Section 3.2.7, we get the p-values of the Wilcoxon Mann-Whitney tests. We show the p-values of the male approach for each metric in all three networks with Table 4.3. Only KERAS shows statistical significance in the result of the hierarchy metric in the co-change network. The VSCODE, TENSORFLOW, ATOM, KERAS, THREE.JS, and DENO projects get the p-values below 0.05 for both metrics in the issue and co-change-issue networks, while all the p-values for NEXTCLOUD and REVEAL.JS are greater than 0.05. Additionally, the results for VUE show statistical significance for only the hierarchy metric in the issue and co-change-issue networks.

Only some projects support *H.M10*, not all of them. At least 6 out of 9 projects show results that could support hypotheses *H.M10* in issue and co-change-issue networks. In the co-change networks, only the KERAS project supports *H.M10* based on the coreness values derived from the hierarchy metric. Therefore, we conclude the results for hypotheses *H.M10* to be **inconclusive**.

We show the results for the female approach of this analysis with Table 4.4. Unlike the male approach, we have only one the p-value below 0.05. This result is derived

Table 4.1: This table describes the results of the statistical tests related to general contribution statistics. The numbers mark the *p-values*. The used approach is the male approach.

Project	created pull request	created issues	merge operations	merged pull requests	pull request comments	issue comments	commits	changed files	diff size
Vscode	0.393	0.1121	0.1125	0.5757	0.3437	0.0006	0.3773	0.4375	0.4677
Tensorflow	0.9903	0.828	0.3883	0.9998	0.9917	0.0962	0.9918	0.9888	0.9965
Atom	0.4683	0.002	0.11	0.7568	0.2193	5.287E-08	0.7436	0.8641	0.8135
Keras	0.09	0.5395	0.6	0.2316	0.1307	0.035	0.3021	0.2134	0.505
Nextcloud	0.06901	0.4746	0.136	0.1526	0.5793	0.1161	0.6478	0.3272	0.4925
Vue	0.5829	0.5167	NA	0.3684	0.293	0.066	0.2354	0.6335	0.4384
Three.js	0.3308	0.09873	NA	0.1137	0.4651	0.02142	0.004	0.2265	0.1826
Deno	0.325	0.02036	0.5	0.5027	0.6582	0.004175	0.7863	0.2695	0.8054
Reveal.js	0.0986	0.01884	NA	0.4716	0.1321	0.1115	0.3287	0.5855	0.2949

Table 4.2: This table describes the results of the statistical tests related to general contribution statistics. The numbers mark the *p-values*. The used approach is the female approach.

Project	created pull request	created issues	merge operations	merged pull requests	pull request comments	issue comments	commits	changed files	diff size
Vscode	0.6071	0.8879	0.8946	0.4245	0.6564	0.9994	0.6284	0.5684	0.5377
Tensorflow	0.009	0.172	0.6187	0.0002	0.008	0.9038	0.008	0.011	0.003
Atom	0.532	0.9973	0.8929	0.2439	0.7808	1	0.2576	0.1368	0.1875
Keras	0.9095	0.4605	0.6	0.7686	0.8694	0.9645	0.6983	0.7869	0.4954
Nextcloud	0.9312	0.5255	0.8823	0.848	0.4209	0.8839	0.3533	0.6738	0.5086
Vue	0.4177	0.4833	NA	0.6335	0.7072	0.9337	0.7685	0.3703	0.5653
Three.js	0.6693	0.9013	NA	0.8864	0.535	0.9786	0.9954	0.7738	0.8177
Deno	0.6224	0.9797	0.6667	0.4984	0.3422	0.9958	0.2214	0.738	0.2009
Reveal.js	0.9016	0.9812	NA	0.5322	0.8681	0.8885	0.6767	0.4218	0.7092

from the hierarchy metric of Tensorflow in the co-change network.

We do not have sufficient results to accept hypothesis H.F10. There is only one result that supports the hypothesis based on the coreness values derived from the hiearchy metric , while we do not have any results that support the hypothesis based on the eigenvector centrality metric. Overall, we **can not accept** hypothesis *H.F10*

Now, we describe the results of the comparison in terms of the distribution of the coreness values. As explained in the second paragraph of Section 3.2.7, we look at QQ plots and boxplots to evaluate *H11* described in Table 3.4. QQ plots are used to compare distributions of our coreness values against the normal distribution. Quantiles of the normal distribution are plotted along the x-axis which is called *Theoretical quantiles*, while quantiles of our coreness values are plotted along the y-axis. First, we look at the distribution shapes of the coreness values of both genders. As an example of the co-change networks, we look at Figure 4.1 which contains the distribution of coreness values calculated using eigenvector centrality in the co-change network of the Tensorflow project. The most important result is that coreness values of male and female developers follow a similar distribution shape in the Tensorflow project, as well as other projects. Unsurprisingly, the number of male developers is more than that of female developers across the coreness scale.

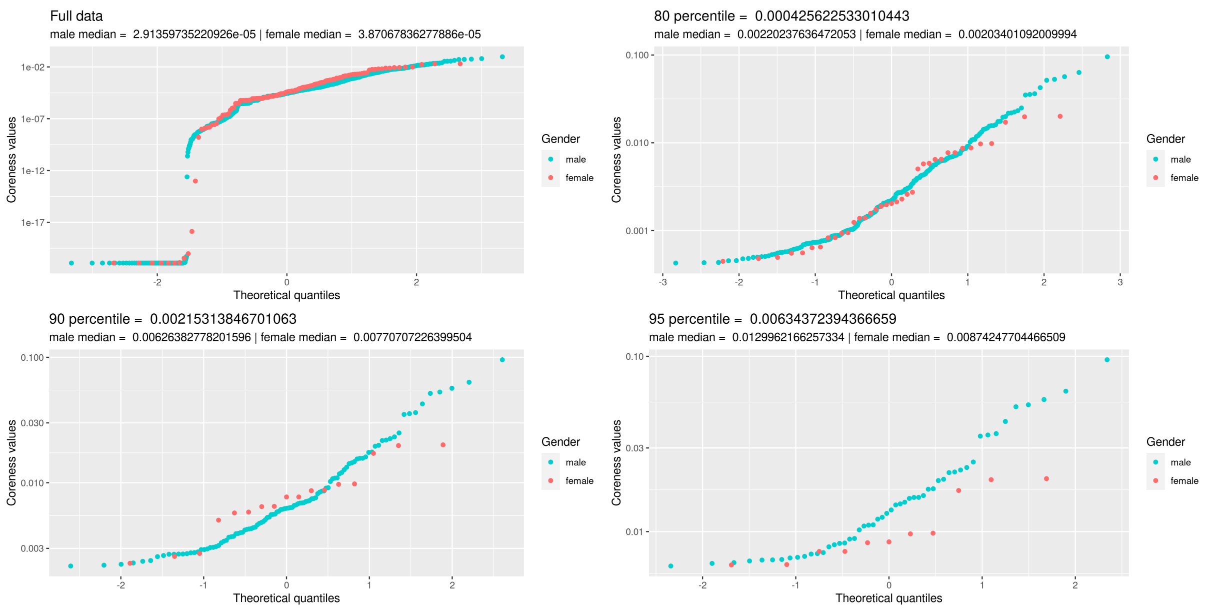


Figure 4.1: QQ-plots for the coreness values of male and female developers in Tensorflow project. The metric is eigenvector centrality. The network type is co-change. The data is shown in a logarithmic scale.

As expected, the number of male and female developers decreases going to the upper parts of the coreness scale. There are even projects such as VsCode,

Table 4.3: This table describes the results of the statistical tests related to developer coreness. The numbers mark the p-values. The used approach is the male approach.

Project	co-change eigenvector	co-change hierarhcy	issue eigenvector	issue hierarhcy	co-change-issue eigenvector	co-change-issue hierarhcy
Vscode	0.4838	0.4596	4.507E-05	0.006	5.14E-05	0.006
Tensorflow	0.8894	0.9978	5.648E-06	0.027	7.544E-06	0.046
Atom	0.8731	0.8114	9.975E-11	5.277E-06	7.072E-12	5.792E-06
Keras	0.06328	0.048	0.006	0.04	0.008	0.033
Nextcloud	0.1032	0.5376	0.5263	0.4966	0.5363	0.5293
Vue	0.1716	0.6095	0.1573	0.009	0.1441	0.009
Three.js	0.399	0.4883	0.0006	6.149E-05	0.001	9.492E-05
Deno	0.5179	0.6245	0.009	0.006	0.009	0.006
Reveal.js	0.4318	0.4369	0.1035	0.07	0.10	0.072

Table 4.4: This table describes the results of the statistical tests related to developer coreness. The numbers mark the *p-values*. The used approach is the female approach.

Project	co-change eigenvector	co-change hierarhcy	issue eigenvector	issue hierarhcy	co-change-issue eigenvector	co-change-issue hierarhcy
Vscode	0.5215	0.5457	1	0.9934	0.999	0.9933
Tensorflow	0.1106	0.002	1	0.9726	1	0.9537
Atom	0.1277	0.1897	1	1	1	1
Keras	0.9368	0.9515	0.9939	0.9571	0.9914	0.9667
Nextcloud	0.8973	0.4635	0.4737	0.5034	0.4637	0.4707
Vue	0.8308	0.3941	0.8427	0.9904	0.8559	0.9913
Three.js	0.6014	0.5121	0.9994	0.9999	0.9989	0.9999
Deno	0.4911	0.3841	0.9904	0.9938	0.9901	0.9941
Reveal.js	0.5728	0.5679	0.8965	0.9294	0.8974	0.9275

Nextcloud, Vue, Deno, Reveal.js, where the upper parts do not contain female developers. For example, the coreness distribution of the Vscode project is shown in Figure 4.2. As can be seen, there is no female developer in the top 10% of the scale. The coreness values obtained through hierarchy metric show similar patterns on the eigenvector centrality metric. Therefore, we present these results only in the main thesis.

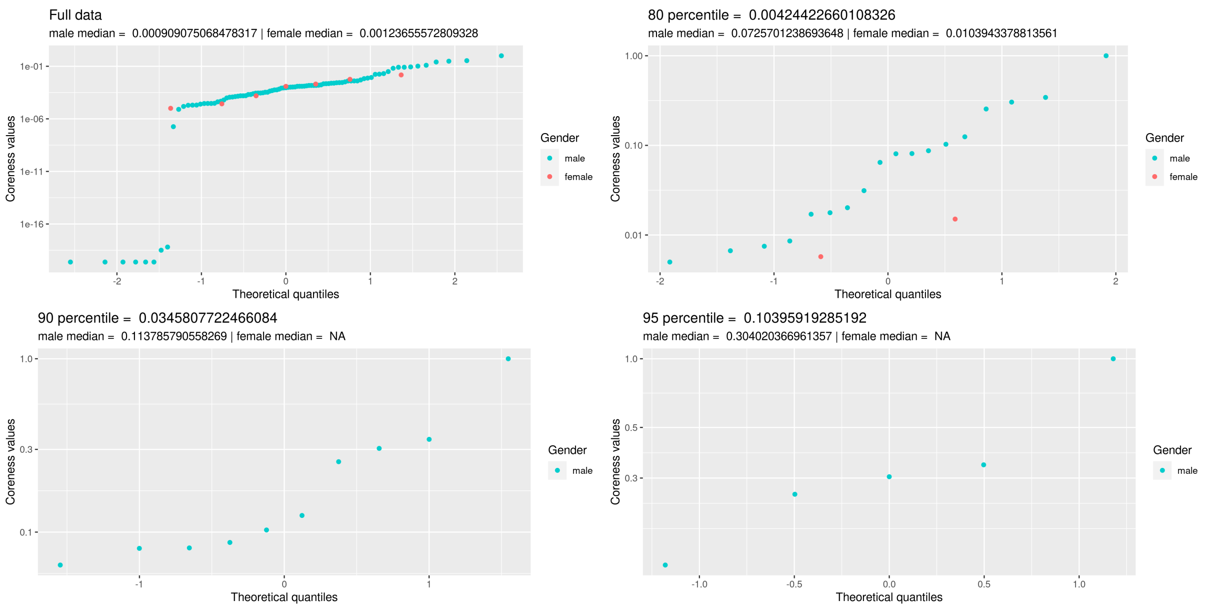


Figure 4.2: QQ-plots for the coreness values of male and female developers in the Vscode project. The coreness metric is eigenvector centrality. The network type is co-change. The data is shown on a logarithmic scale.

We show overview of the distribution of the coreness values in the issue networks through Figure 4.3. The figure contains data of the Vscode project. The main difference between the co-change and issue networks is that we have much more data points in the issue networks. Unlike the co-change networks, female developers exist in all parts of the distribution in the issue networks. Despite these differences, the main similarity is the distribution shape. As such in the co-change networks, coreness values of male and female developers follow a similar distribution shape in the issue networks, too.

As we explained in Section 2.4, a co-change-issue network is the combination of co-change and issue networks. Figure 4.3 evidence that the issue networks have much more data points than the co-change networks. Therefore, the co-change-issue networks are similar to the issue networks. Overall, the results of the co-change-issue networks are quite similar to the results of the issue networks.

Furthermore, as evident in Figure 4.4, the male group have more outliers in all types of networks. These patterns can be seen in all the projects. Furthermore, the

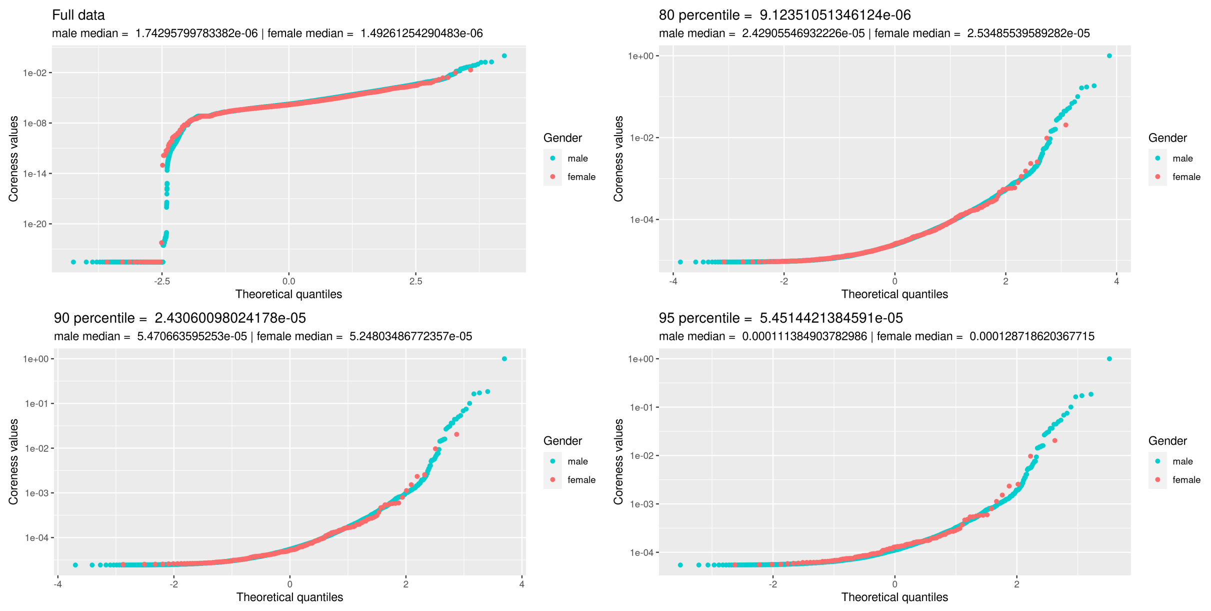


Figure 4.3: QQ-plots for the coreness values of male and female developers in the VSCODE project. The coreness metric is eigenvector centrality. The network type is issue. The data is shown on a logarithmic scale.

range of coreness scale differs from project to project. The certain projects have wide ranges of coreness scale, whereas others have narrow ranges of coreness scale.

The results suggest that the coreness values of both metrics follow a similar distribution for both gender in all the projects. Overall, we **accept** *H11*.

Lastly, we present the results of the third step of the analysis explained in Section 3.2.7. In the co-change networks, projects show different results. An interesting point is that the upper parts of VSCODE, NEXTCLOUD, VUE, DENO, REVEAL.JS contain only male developers, while all parts of other projects contain both genders. For example, Figure 4.5 shows the distribution of the VSCODE project in the co-change network. In this example, the coreness values are calculated using the eigenvector centrality metric. According to this figure, we can not say that both gender groups are distributed equally in each range. As shown in the figure, 50% of male and 42% of female developers fall within the lowest range, 0% - 50%. It means that 50% of male and 42% of females developers' coreness values are less than 50th percentile of the coreness scale and this inequality between genders is shown in other ranges. Even the top 10% of the coreness scale belong to only male developers. Overall, there is no distribution pattern that all the projects follow. Each project has different gender distribution over the coreness scale. One coreness distribution we need to address here belong to the KERAS project which is described in Figure 4.6. KERAS is the only project where distributions of both genders are quite similar to each other. The difference in the lowest range of the coreness scale, 0% - 50%, is a little bit high.

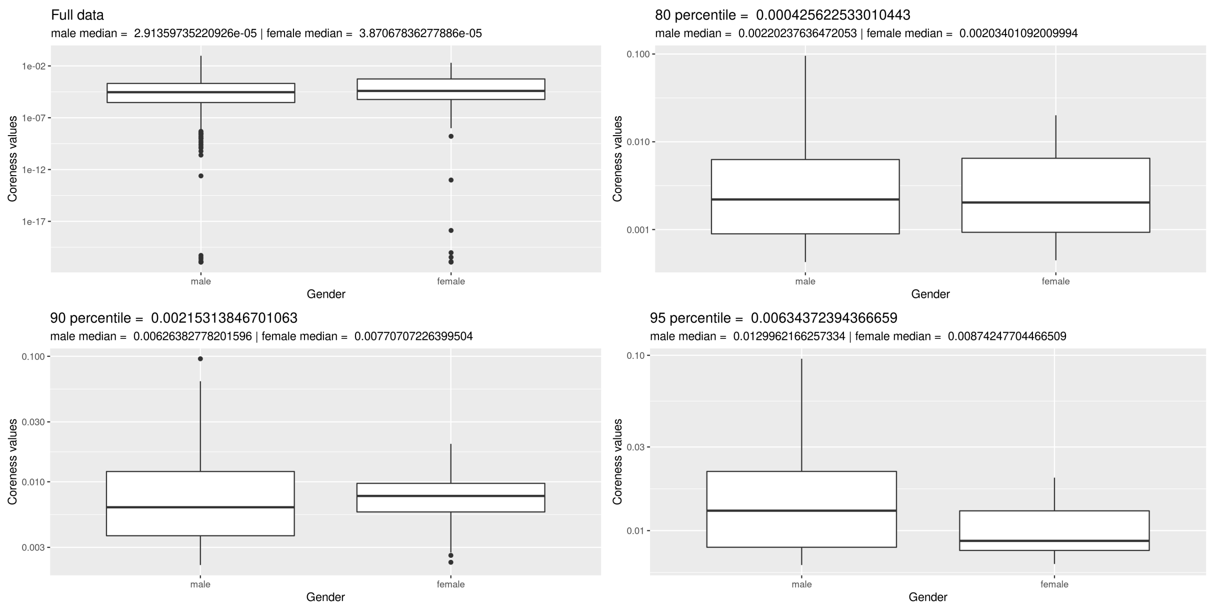


Figure 4.4: Boxplots for the coreness values of male and female developers in the TENSORFLOW project. The coreness metric is eigenvector centrality. The network type is co-change. The data is shown on a logarithmic scale.

However, we can say that the percentages of both genders are almost equal in all ranges of the coreness scale.

The coreness values based on the hierarchy metric show similar patterns as the coreness values of the eigenvector centrality metric. Each project shows different distribution of of male and female developers over the coreness scale. The projects whose upper ranges contain only male developers in the comparison based on eigenvector centrality metric, show the same pattern also in the comparison according to hierarchy metric. Only THREE.JS is a project where both genders show quite similar percentages in each range of the coreness scale based on the hierarchy metric. We show this similarity in Figure 4.7.

The results of the issue networks are more interesting. Unlike co-change networks, all ranges of all the projects contain both genders. Additionally, both genders are shared almost equally in all ranges. We show a summary of the distributions of genders over the coreness scale in the issue networks in Figure 4.8 taking the VSCODE project as an example. As can be seen in the figure, the highest difference between male and female developers is about 4% which exists in the 0%-50% range. The coreness values of the hierarhcy metric show similar results like the coreness values of the eigenvector centrality metric. Although distribution of both genders are quite similar in each range, ATOM, THREE.JS, DENO, REVEAL.JS projects show the difference more than 5% in the lowest range of both metrics, 0% - 50%. Moreover, NEXTCLOUD and DENO show the difference more than 5% in the 51% - 75% range. The difference in the NEXTCLOUD project is related to the hiearchy metric, while the difference of

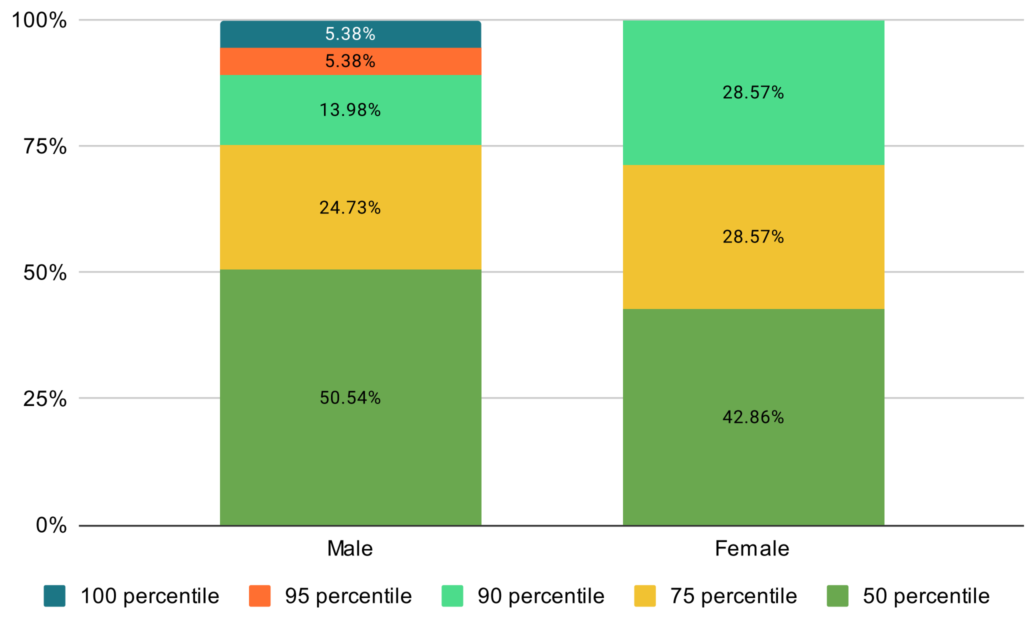


Figure 4.5: Distribution of the male and female developers over the coreness scale of the VSCODE project. The coreness metric is eigenvector centrality. The network type is co-change.

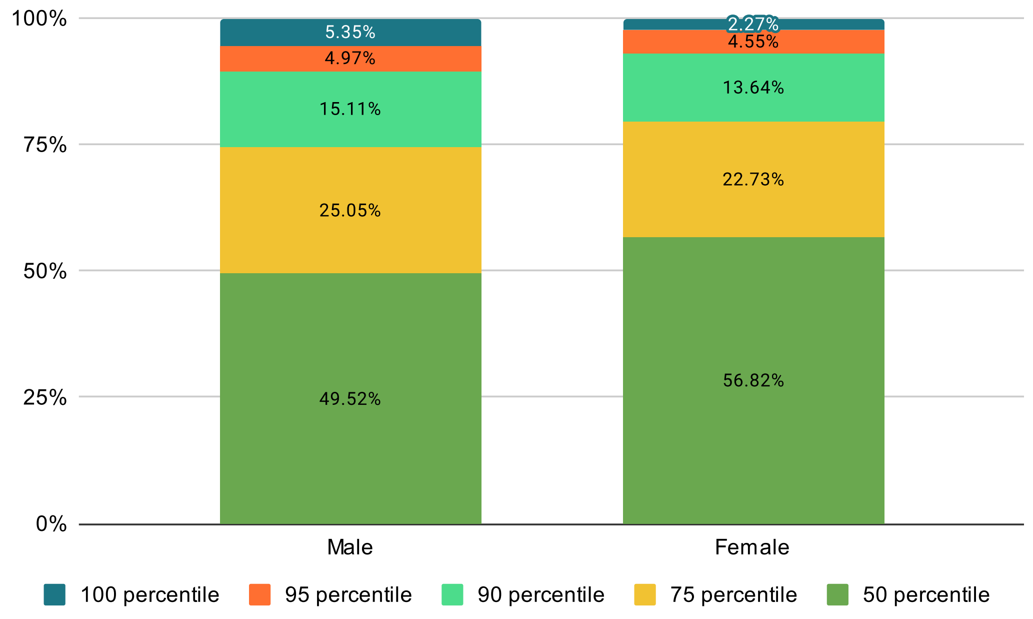


Figure 4.6: Distribution of the male and female developers over the coreness scale in the KERAS project. The coreness metric is eigenvector centrality. The network type is co-change.

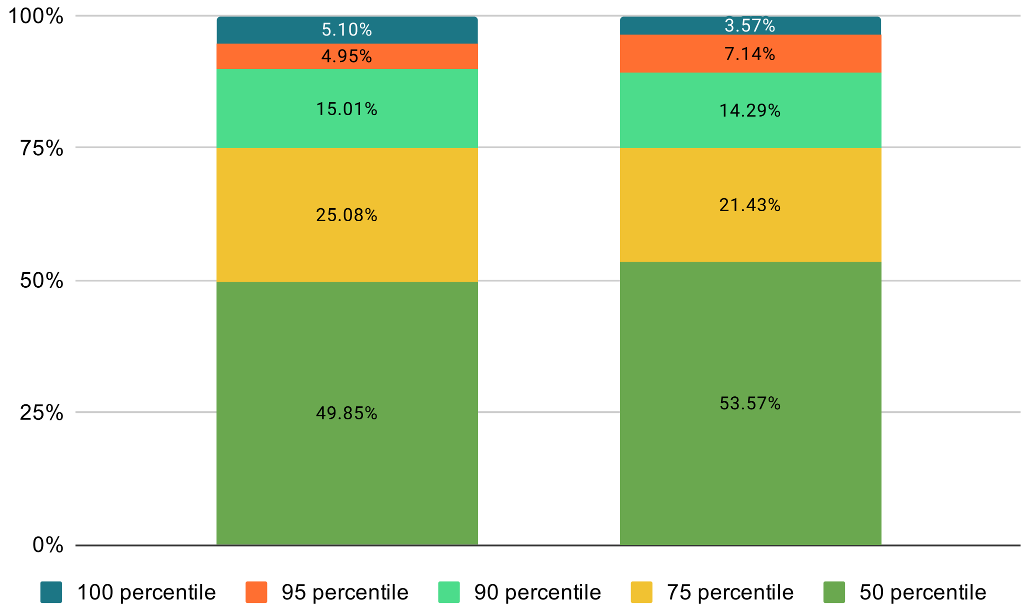


Figure 4.7: Distribution of the male and female developers over the coreness scale in the THREE.JS project. The coreness metric is hierarchy. The network type is co-change.

DENO is related to the eigenvector centrality metric. All the differences in the other ranges is less than 5%. The related figures can be found in Appendix A. Like other results of the co-change-issue networks, these results are similar to the results of the issue network.

As far as we described all the results, it is time to evaluate *H12*. As we mentioned above, the results of the coreness values based on the eigenvector centrality metric are similar to the results of the hierarchy metric. That is why, the conclusion is the same for the coreness values of both metrics.

Male and female developers are not shared equally in the ranges of the coreness scale in the co-change networks. Contrary to this, the percentages of both genders are almost equal in the issue and co-change-issue networks. It means that the distribution of each gender over different ranges is completely different in the co-change networks, while the distribution is similar for both genders in the issue and co-change-issue networks. Overall, we can **accept** *H12* for only the issue and co-change-issue networks, not the co-change networks.

4.2 DISCUSSION

The results presented in Section 4.1 revealed several interesting facts. Below we discuss the relevance and potential explanations of the results.

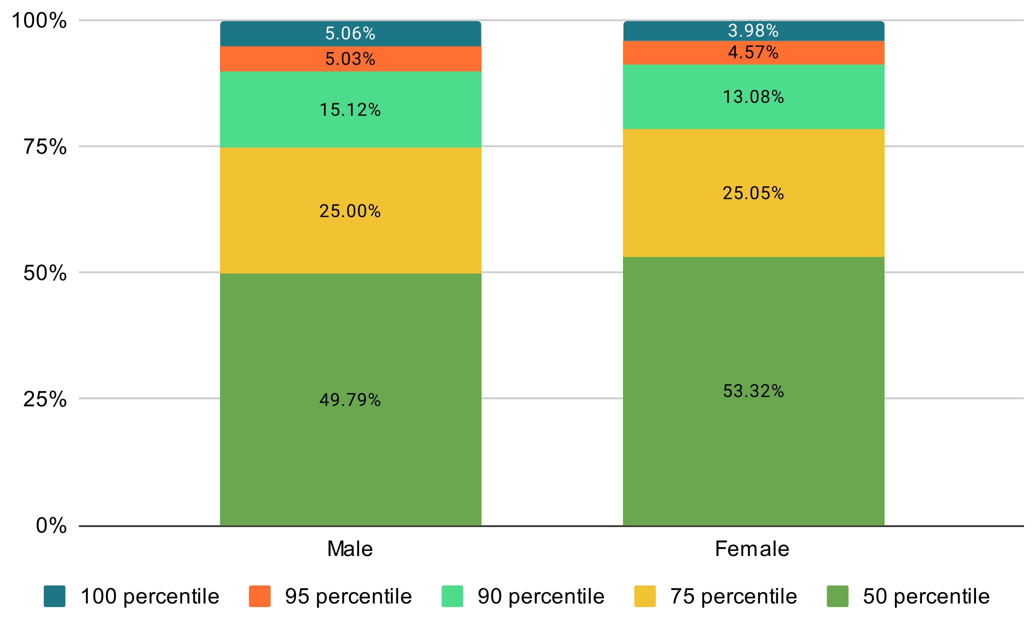


Figure 4.8: Distribution of the male and female developers over the coreness scale in the VSCODE project. The coreness metric is eigenvector centrality. The network type is issue.

4.2.1 *General Contribution Statistics*

In the first part of the analysis, we compare genders in terms of the contribution related parameters such as created pull requests, created issues, merge operations and others. As we explained in Section 3.1.1, we have two different approaches for the comparison. The results show that the differences between male and female developers differ from OSS project to OSS project. The results indicate that merge operations are performed by only male developers in VUE, THREE.JS, and REVEAL.JS. It is an indication that the last decisions about the source code are made by only male developers in these projects. The interesting point we need to address here is that we have much more statistically significant results in the male approach than the female approach. The simplest reason is that there are more male developers than female developers in all the projects as shown in Table 3.6. Although we have many supported hypotheses in the male approach, we can not accept any of them, since the most supported hypothesis are supported by only five projects. The supported hypotheses are related to the created issues, issue comments, and commits. Looking at these results, we can say that there is no statistically significant difference between genders in terms of code contribution, since only one project shows statistical significance in the comparison in terms of commits. On the other hand, male developers participate actively in issues of most of the projects. In the ATOM, DENO, and REVEAL.JS projects, they create more issues than females, whereas they write more comments on the issues in the VsCode, Atom, Keras, Three.js, and Deno.js projects. In the female approach, only the results for Tensorflow show statistically significance in the results. Although there is a significant difference between the number of male and female developers in the Tensorflow project, female developers show better results in terms of different parameters such as created pull requests and merged pull requests. About this project, we can say that female developers are much more active in the code contribution process.

4.2.2 *Developer Coreness*

The next differences between genders we want to discuss are related to coreness values. The p-values for *H.M10* and *H.F10* in each network type are shown through Table 4.3 and Table 4.4. In the co-change networks, only one results supports *H.M10* in each approach. This underlines the result we get from the comparison of the general contribution statistics and indicates that none of the genders predominates the other one in terms of developer coreness in the code contribution process. The statistically significant result in the co-change networks of the female approach is related to the Tensorflow project, while the significant result of the male approach is related to the Keras project. In fact, the result of Tensorflow is an expectable result. Since the results of the general contribution statistics show that female developers are more active in the code contribution process of Tensorflow. The most noticeable result is that *H.M10* is supported in the issue networks of most projects. This result gives us another evidence that strengthens the result we get from the analysis of the general contribution statistics. The result shows that male developers take on more significant roles in the issue related contributions. Although the co-change-issue networks show approximately the same results, so we do not draw a conclusion about these results. Considering the results of both networks, we estimate that significant results of the co-change-issue networks come from the issue networks. Since as explained in Section 2.4, a co-change-issue network is a combination of co-change and issue networks. Moreover, the co-change networks have only one statistical significant result, while the issue networks show multiple significant results.

The last hypotheses of this thesis are related to the distribution of the coreness values. In one of them, *H11*, we hypothesize that both genders follow a similar distribution. The supported results allow us to accept *H11* and conclude that both genders follow a similar distribution in terms of the coreness values in all types of the developer networks. In the middle and lower parts of the scale, we see the same distribution for males and females which simply means that in these parts of the distribution gender does not matter, gender matters only in the upper parts. In other words, generally, the most crucial roles are taken by male developers. Even in the co-change networks, the upper ranges of 5 out of 9 projects consist of only male developers. Like other results, also this result indicates that there is no female developer play a significant role in the code contribution process. Unlike the co-change networks, the top 5 percentage of the coreness scale contains both genders in the issue networks of all the projects. This indicates that female developers play standard roles as well as significant roles in the issue related contributions, although, most of the significant roles are taken by male developers. Having more outliers in the male groups shows that there are many male developers who demonstrate different behaviors and get higher or smaller coreness values than others individually. Morover, the range of coreness scale differ from OSS project to OSS project.

The evaluation of the hypothesis *H12* shows that both genders are not distributed equally over coreness scale in the co-change networks. This result indicates that female developers get only lower importance levels and the highest importance levels are dominated by male developers in the VsCode, Nextcloud, Vue, and Deno, Reveal.js projects. However, one gender does not predominate the other one, this demonstrates the importance of male developers in the code contribution process of these projects. Unlike the co-change networks, the issue networks show similar distributions of genders over the coreness scale, although the difference between genders is more than 5% in the lowest range of 4 out of 9 projects. Another difference is that each range contains both genders, which means that female developers take also significant roles like male developers in the issue-related contributions. In the co-change networks, the distribution of male developers follows a pattern. The pattern is that male developers exist in each range of the coreness scale and as the range percentage increases, the percentage of males decreases until the 90th percentile. Unlike male developers, female developers have no such pattern. In projects such as Deno and Reveal.js, there are some middle and upper ranges that do not contain female developer. Additionally, in some cases, the upper ranges have a higher percentage of females. Differently from the co-change networks, both genders follow the pattern explained above in issue and co-change-issue networks.

4.3 THREATS TO VALIDITY

In the last section of the evaluation, we discuss the threats to validity of our analysis.

One of the main threats is our project selection. The most obvious threat is the number of chosen projects. Our analysis is based on nine OSS projects. Although we analyzed different size projects, it is difficult to generalize our results to all OSS projects. Additionally, chosen projects use the pull-request-based tool as a contribution tool. We can not determine whether projects that use different contribution tools show similar results. The next threat is the chosen time range of each project. We analyzed data of the specific time range of each project in which both commit and issue data exist. It is unclear whether we can experience the same results in the whole lifetime of the projects.

For gender detection, we chose the name-to-gender inference method. The name-to-gender method is a threat, since many people use some kind of usernames or do not use real names. Moreover, some names can be both male and female depending on the geography and ethnicity of the person. So, we do not know real genders of all the developers. We just estimate genders from names provided on GitHub. Therefore, it is inevitable to determine some developers' genders incorrectly.

The next big threat is the selected tools. GenderAPI does not predict genders of all the developers. There are lots of developers that are not part of our analysis because of unknown gender. It can be possible to get different results with the full gender data. In addition to GenderAPI, there are other tools that lead to threats for our analysis. We mined data from GitHub. We do not know whether other pull request systems such as Bitbucket, Gitlab show similar results. Another important threat is related to gender detection. Another tool we have to rely on is Codeface, since data extraction forms the base of our analysis.

Some important threats are related to networks we use to represent reality accurately. It is possible that used networks do not reflect the truth. Another threat is the detection of developer coreness values using network metrics. Each network metric relies on different parameters. We are not sure these parameters are good proxies for the developer importance in OSS projects. There may be other parameters that can represent developer importance more accurately.

5 CONCLUDING REMARKS

In this chapter, we summarize our analysis and its results. Additionally, we give an outlook of future work.

5.1 CONCLUSION

In this thesis we analyzed whether there are important differences between male and female developers in OSS projects. The analysis consists of two parts. In each part, we looked at differences between genders in terms of different measurable quantities. In the first part, the main parameters for the comparison are the general contribution statistics which are created pull requests, created issues, merge operations, merged pull requests, pull request comments, issue comments, commits, changed files, and diff size. In the second part, the main parameter is the coreness value which serves as a proxy value for the developer importance.

To analyze the differences, we mined the commit data, issue data, and developer data of nine projects from GitHub and their git repositories. The projects are VSCODE, TENSORFLOW, ATOM, KERAS, NEXTCLOUD, VUE, THREE.JS, DENO, and REVEAL.JS. Before the comparison, we used the GENDERAPI tool to assume the gender of each developer from developer's name. After assigning developers that can be classified as male or female to the corresponding group, we analyzed differences between those groups. For the first part of the analysis, we performed Wilcoxon Mann-Whitney tests to check whether the differences between genders in terms of general contribution statistics are statistically significant. For the second part, we constructed three different types of developer networks: co-change networks, issue networks, and co-change-issue networks. After building networks, we calculated coreness values by applying two different network metrics: eigenvector centrality and hierarchy. Then, we performed Wilcoxon Mann-Whitney tests to understand the differences between genders in terms of the coreness values. Additionally, we created different types of plots to investigate differences between distributions of coreness values of both genders. Lastly, we looked at how genders are distributed over the coreness scale.

Since our findings evidence that different OSS projects show various differences between male and female developers, we can not accept or reject all the hypotheses completely. One result we need to address here is that all pull requests of three projects are merged by only male developers, which means that male developers make the final decision about the code. Another important point is that the number of male developers is more than that of female developers in all the projects. Despite this important difference, none of the genders dominate the code contribution process, which means that the differences between genders in terms of commits, changed files, diff size are not significant in this process. On the other hand, male developers predominate female developers in the issue-related contributions of most of the projects and create more issues, and write more comments on the issues. In both code and issue-related contributions, male developers take the most crucial roles. In 5 out of 9 projects, there is no female developer on top of the organizational hierarchy which indicates women marginalization in the code contribution process. Although male developers generally have higher coreness values in 6 out of 9 projects, female developers take standard as well as crucial roles in the issue-related contributions. Additionally, gender has an influence on only the top 10% of the coreness values, not lower percentages. Another interesting outcome of our analysis is that in all the types of developer networks, there are male developers with values all over the coreness scale, however, female developers are not as predominantly present in the top of the coreness scale. Unlike male developers, female developers show the same pattern only in the issue and co-change-issue networks, not the co-change networks. Furthermore, both genders are shared almost equally in different ranges of the coreness scale in the issue and co-change-issue networks. The last finding is that there are many male developers who get much higher coreness values in comparison with other male developers. Though they are not as many as male developers, there are some females who show the same pattern among female developers.

In conclusion, our goal for this thesis was to understand differences between male and female developers in terms of contributions and developer importance. According to our results, we cannot find general differences between male and female developers. However, there are more male developers than female developers in the upper parts of the coreness scale. The only significant result we found is that coreness values of both male and female developers follow a similar distribution shape until reaching the top of the scale. The top of the scale is dominated by male developers. Furthermore, we cannot completely accept or reject other hypotheses, since some projects support them, while other projects suggest the opposite. The differences between genders depend on the projects and parameters chosen for the comparison.

5.2 FUTURE WORK

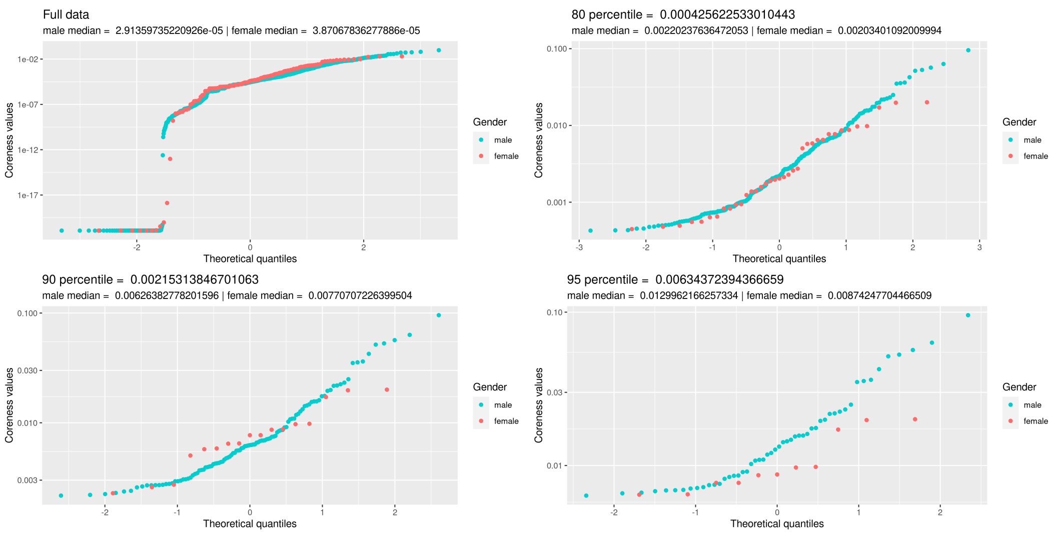
The first thing we need to do is to analyze more OSS projects. This could help to get more generalizable results. Moreover, we can investigate differences in the projects which use patch-based contribution tools. This would help us to generalize our results independently of the used contribution tool. In addition to different contribution tools, it would be possible to perform our analysis using different gender detection tools, even different gender detection methods. We have to find a way to reduce data loss because of the unknown gender. Therefore, it would also be good to find some kind of combination of different gender detection methods or tools to reduce the number of developers whose genders are unknown. This would increase the validity of the analysis.

Another good idea to do in the future is to split the projects into different time ranges and looking at differences in each range individually. This would help us to understand how the differences evolve during the development of the projects.

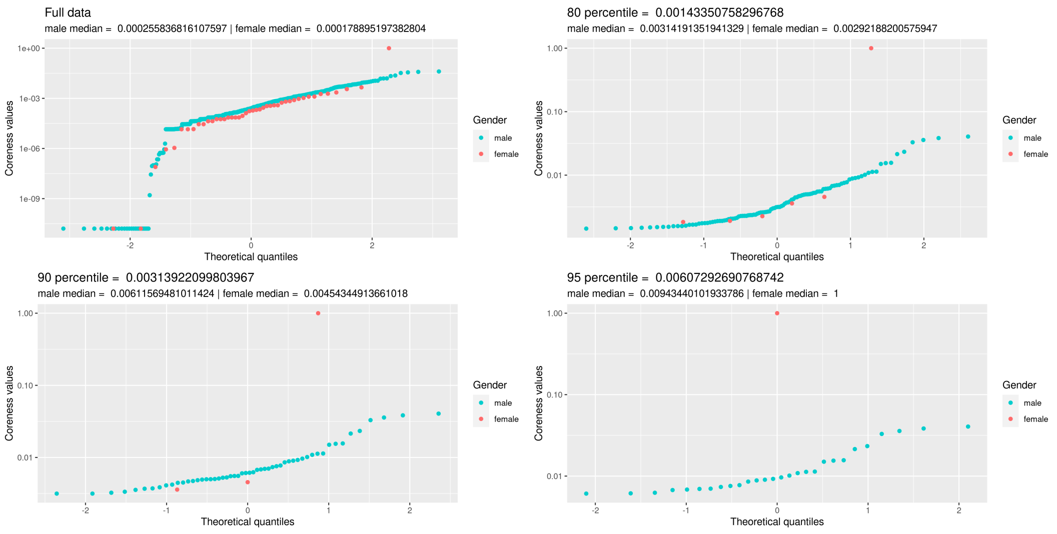
Lastly, projects can be grouped under different categories such as frontend, backend, mobile. Then, the differences between genders can be analyzed in each group. It might be possible that differences between genders depend on the project type, which means that the differences between genders in a frontend project might be smaller than the differences in a backend project or vice versa.

A

APPENDIX

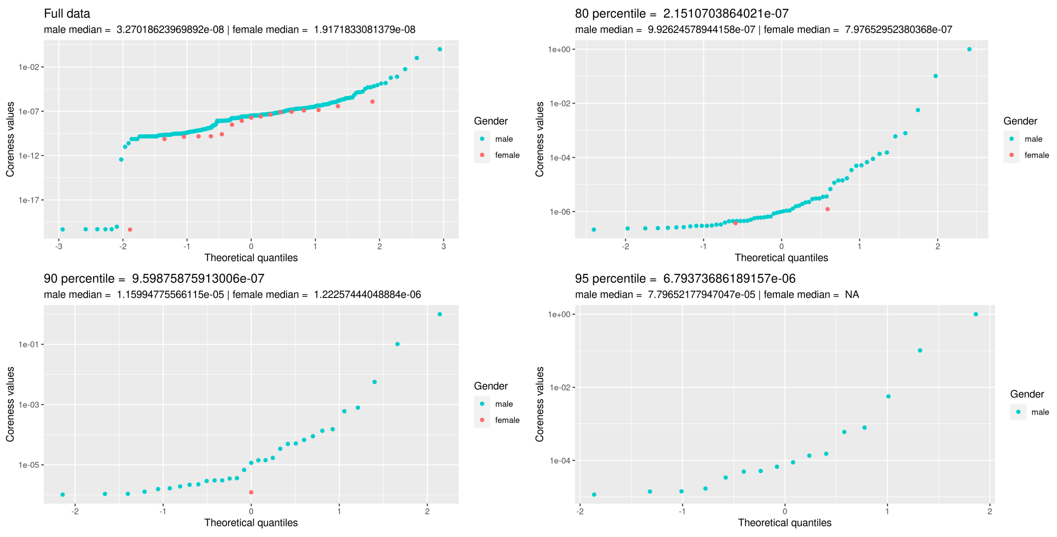


Atom

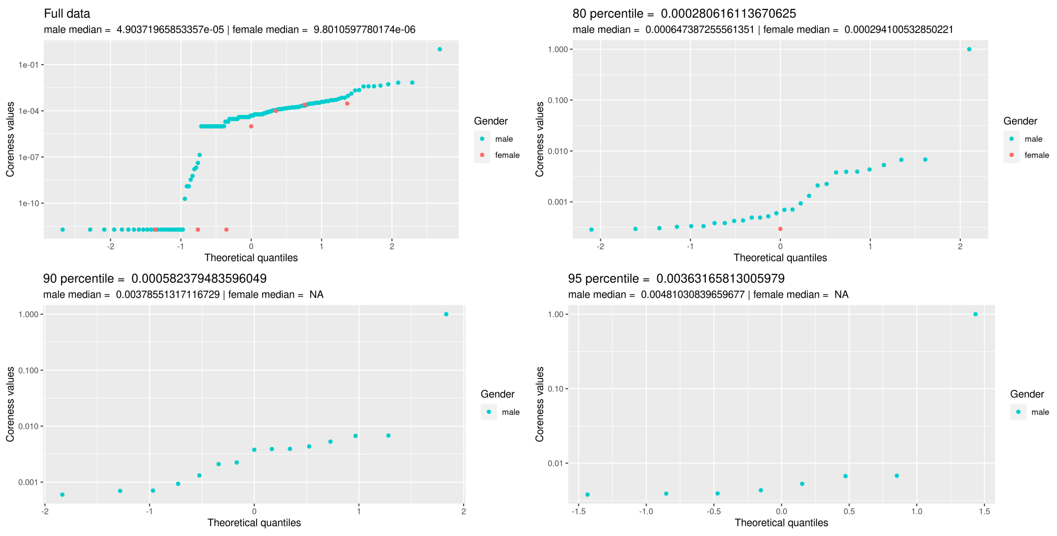


Keras

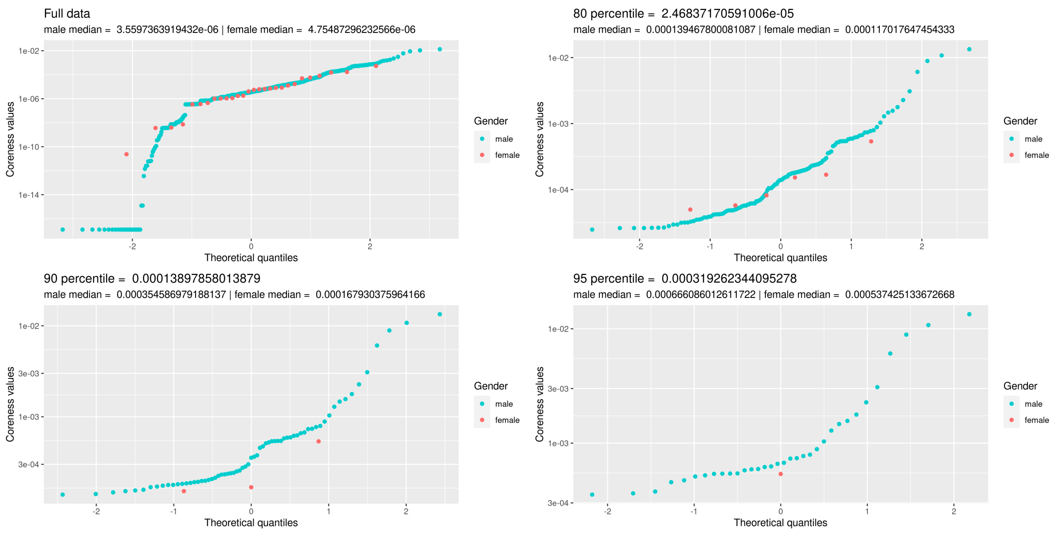
Figure A.1: QQ-plots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is co-change. The data is shown on a logarithmic scale. The projects are ATOM and KERAS.



Nextcloud

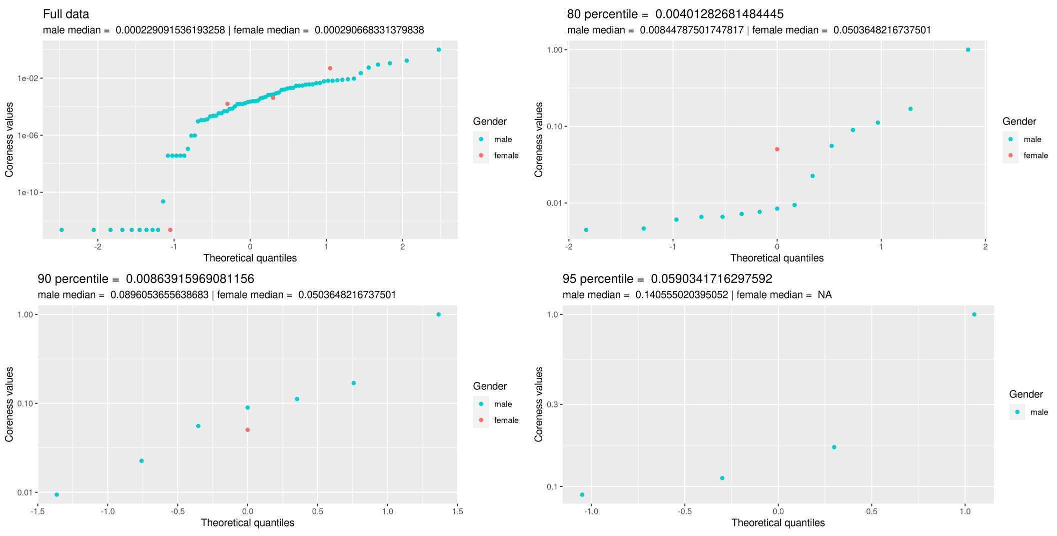


Vue

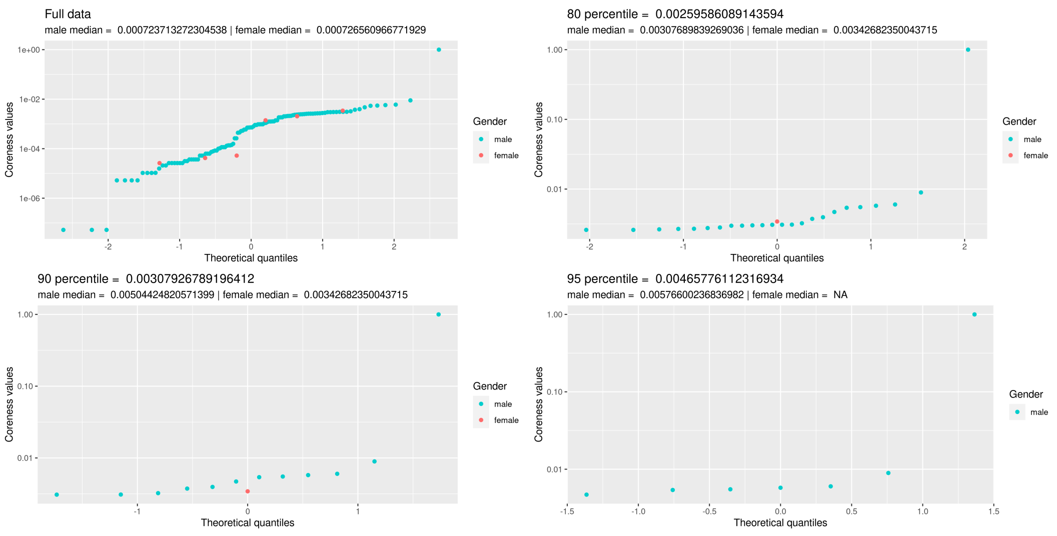


Three.js

Figure A.2: QQ-plots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is co-change. The data is shown on a logarithmic scale. The projects are NEXTCLOUD, VUE and THREE.JS.

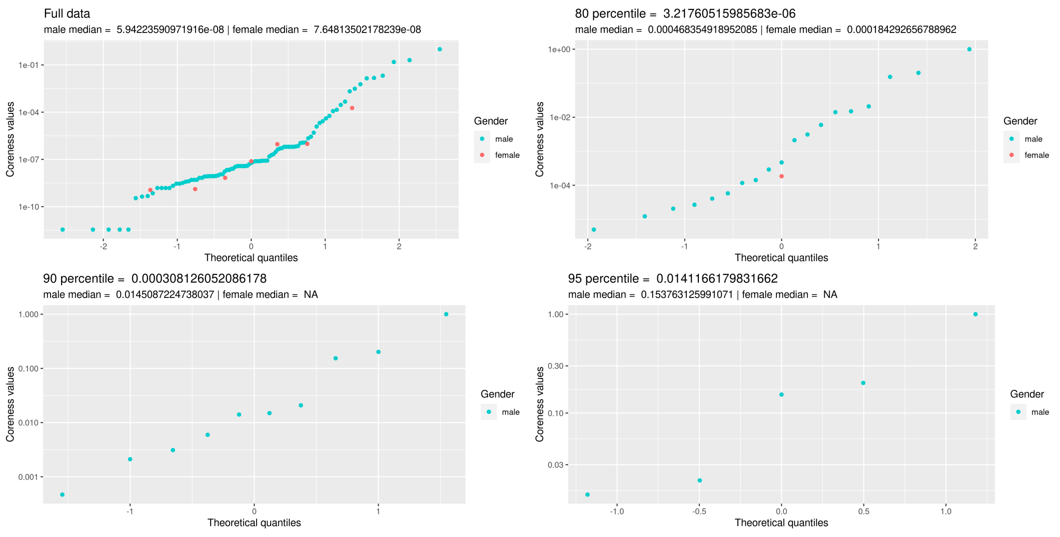


Deno

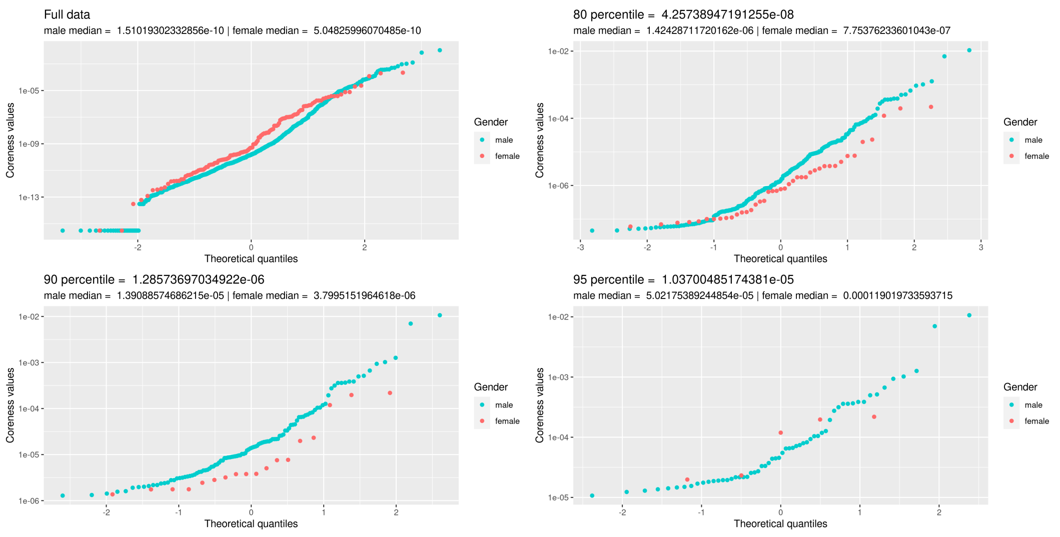


Reveal.js

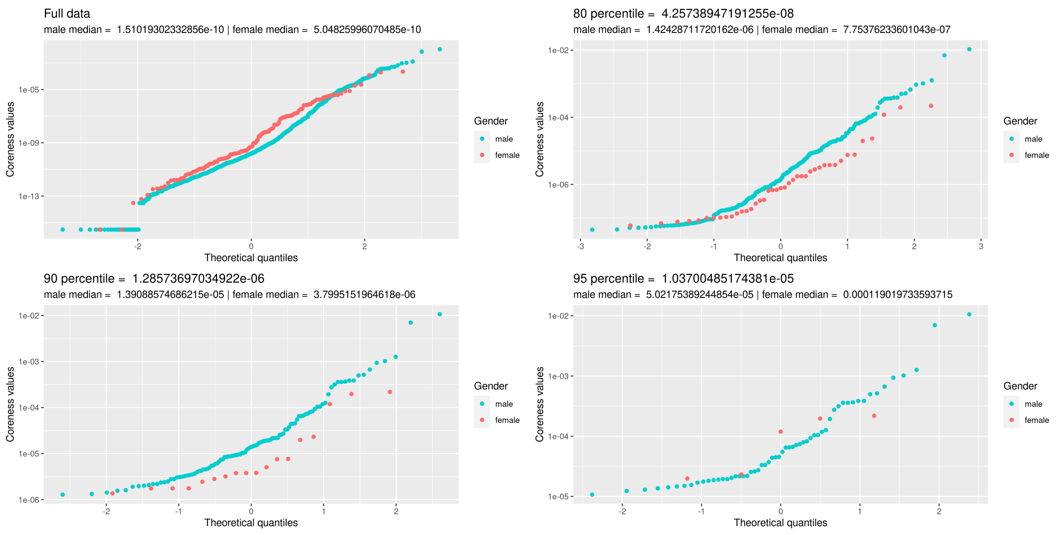
Figure A.3: QQ-plots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is co-change. The data is shown on a logarithmic scale. The projects are Deno and Reveal.js.



Vscode

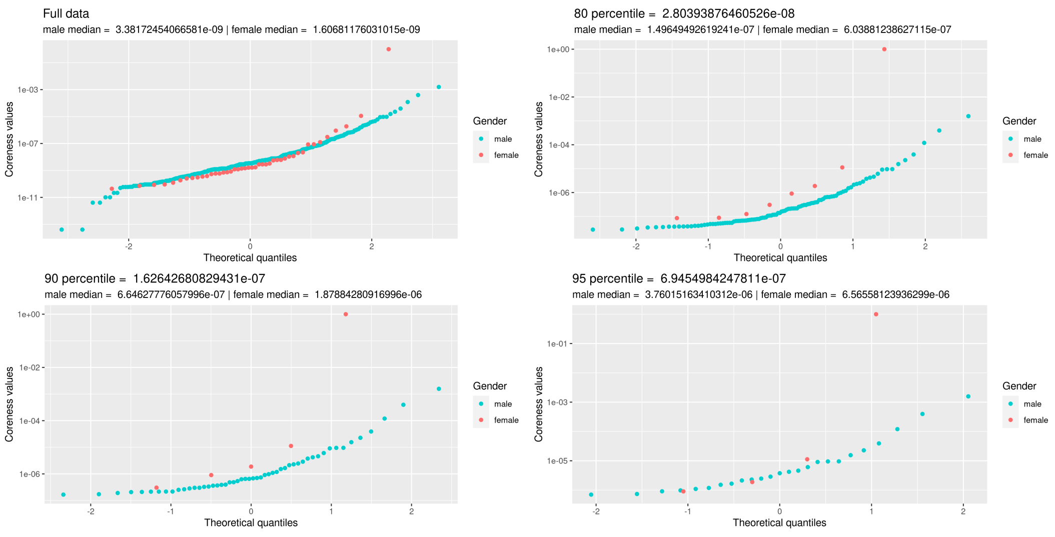


Tensorflow

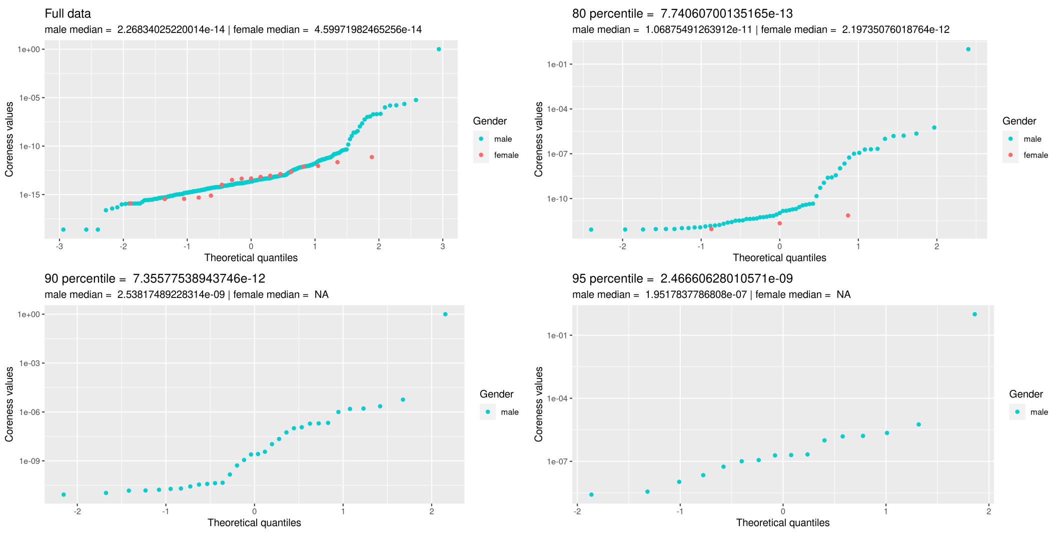


Atom

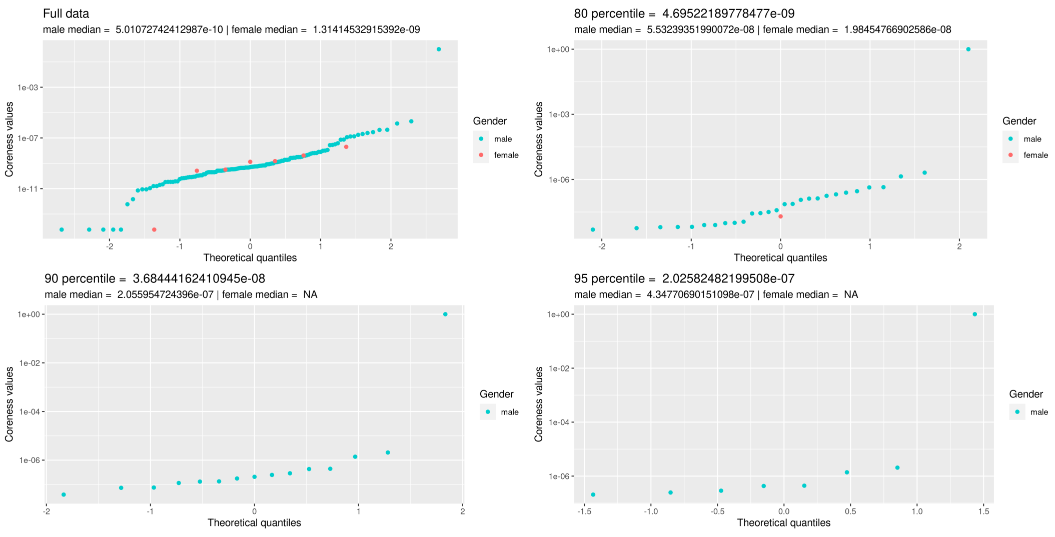
Figure A.4: QQ-plots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is co-change. The data is shown on a logarithmic scale. The projects are Vscode, Tensorflow, and Atom.



Keras

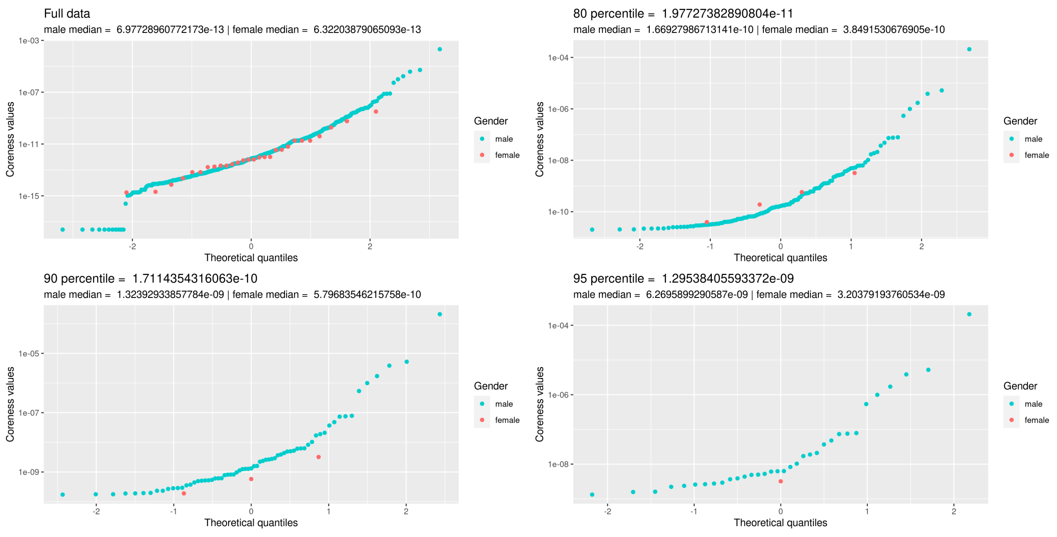


Nextcloud

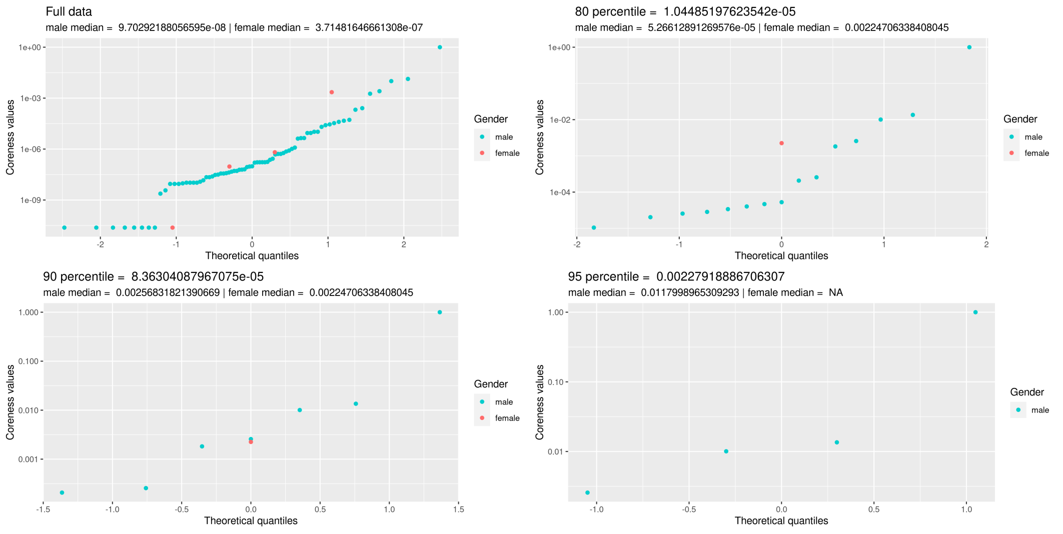


Vue

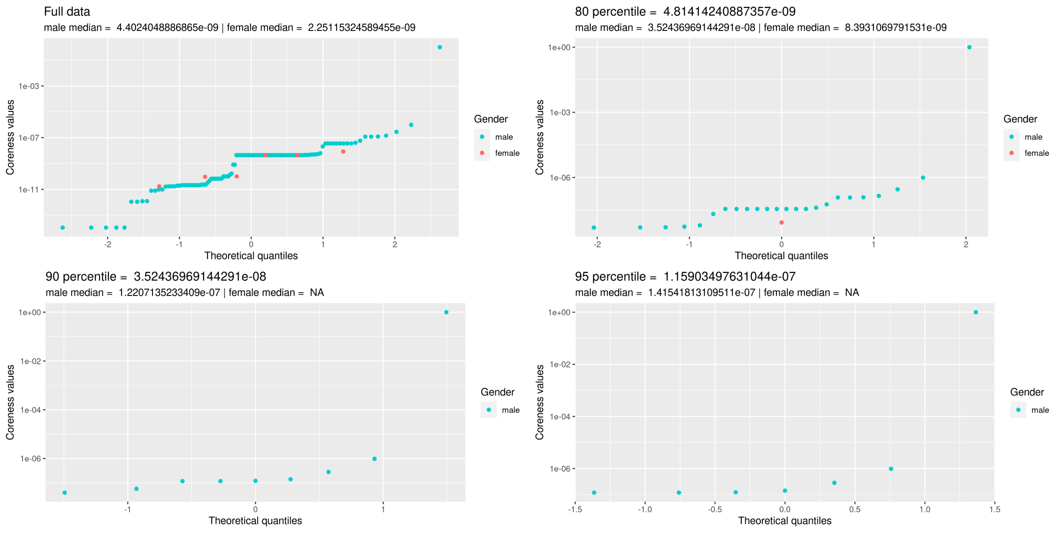
Figure A.5: QQ-plots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is co-change. The data is shown on a logarithmic scale. The projects are KERAS, NEXTCLOUD, and VUE.



Three.js

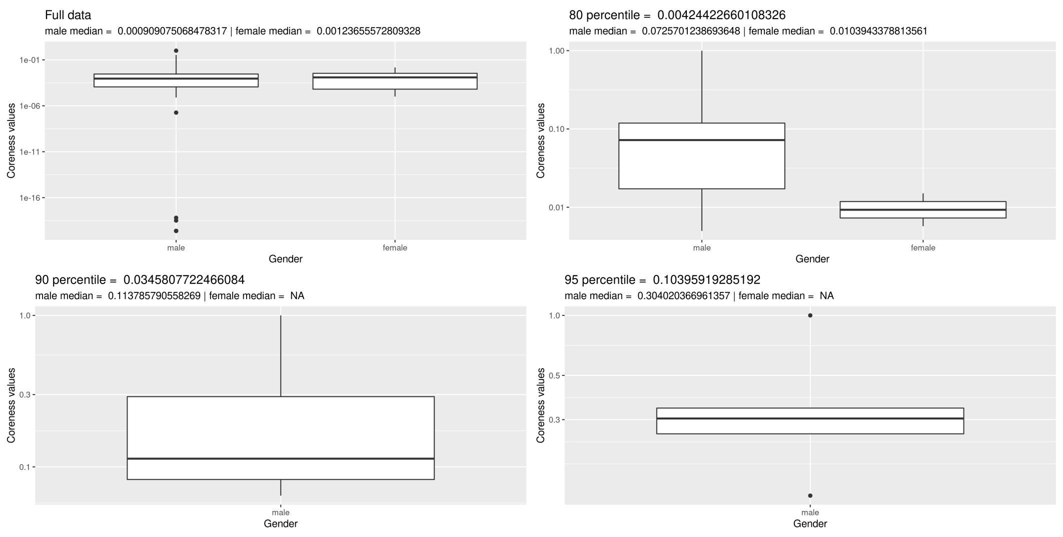


Deno

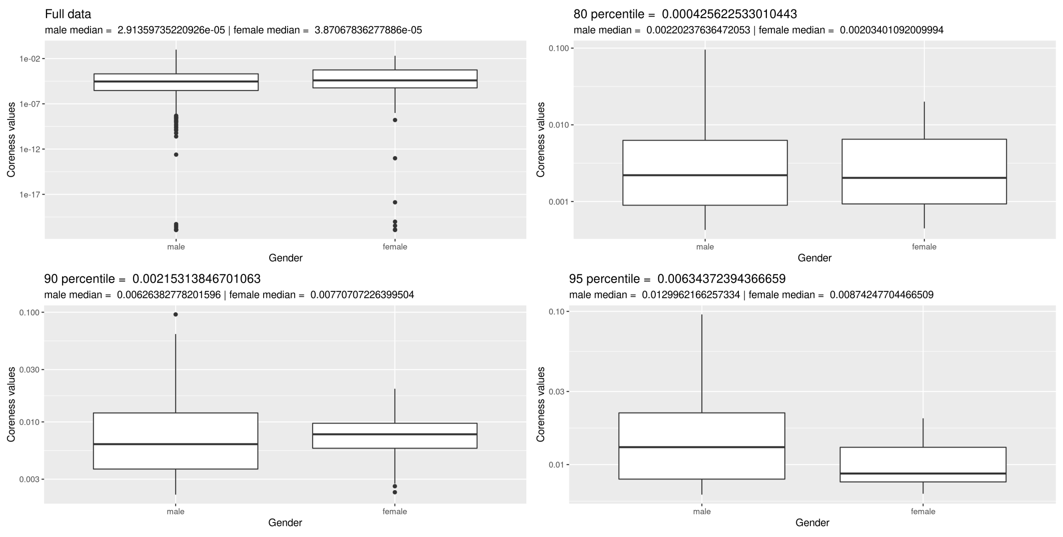


Reveal.js

Figure A.6: QQ-plots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is co-change. The data is shown on a logarithmic scale. The projects are Three.js, Deno and Reveal.js.

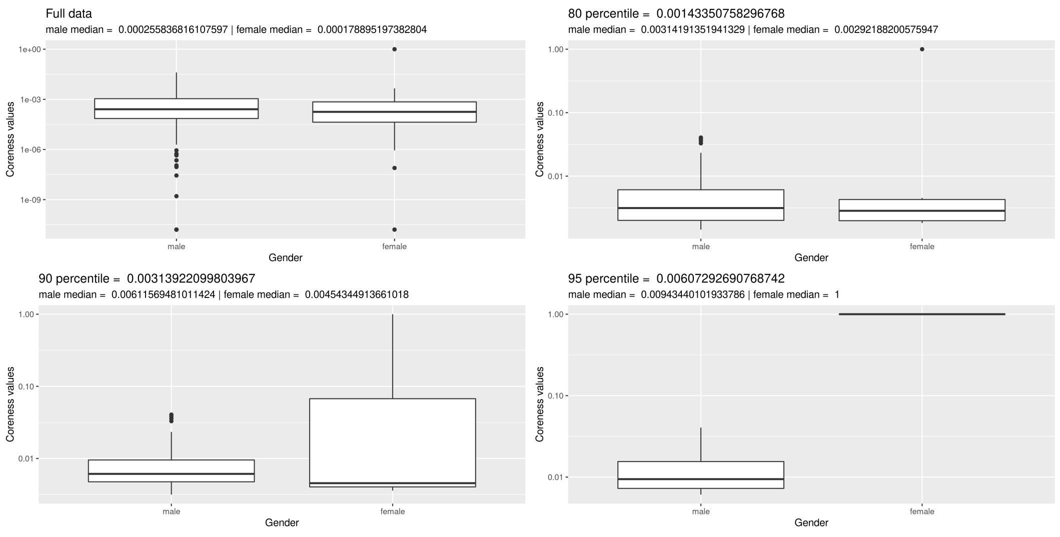


Vscode

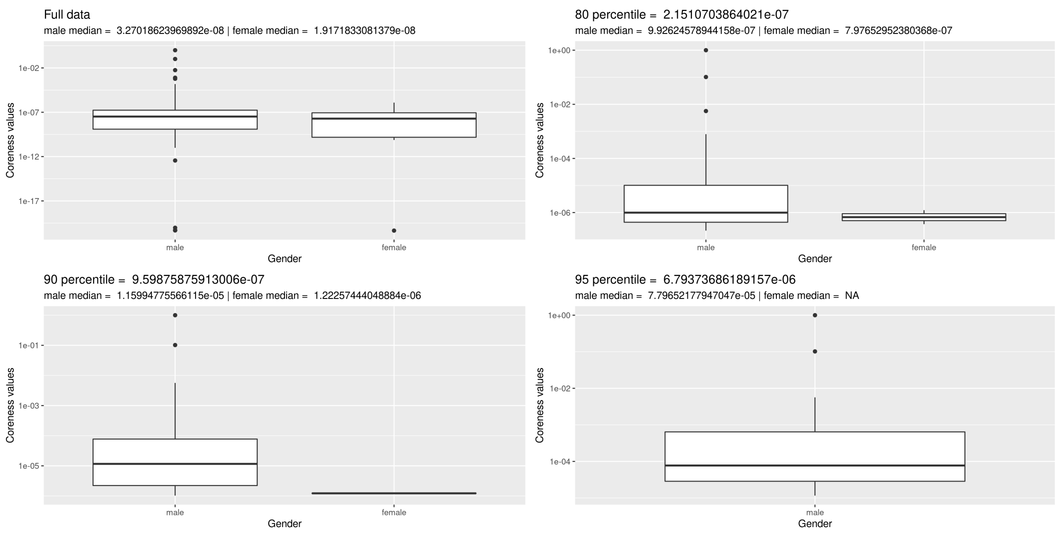


Atom

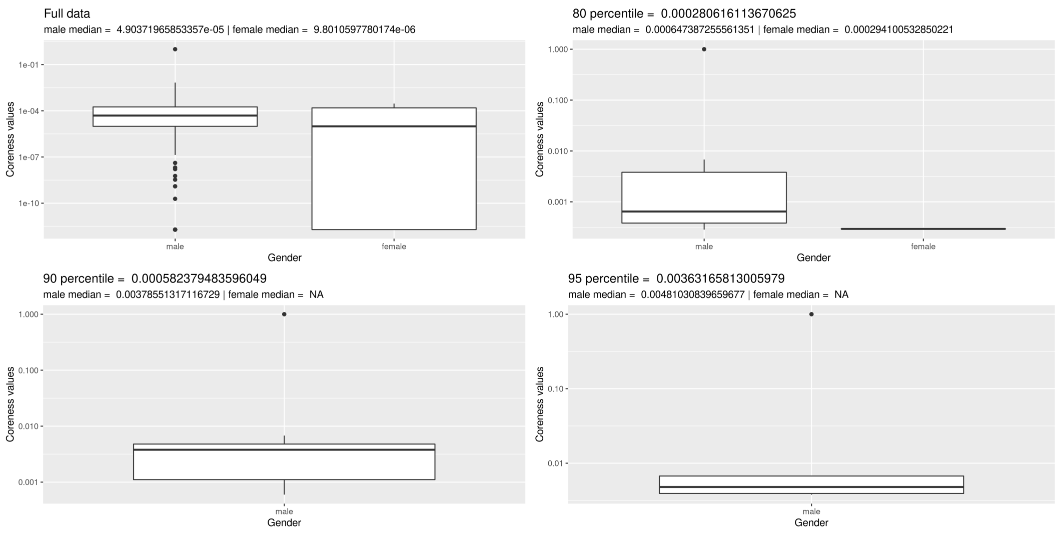
Figure A.7: Boxplots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is co-change. The data is shown on a logarithmic scale. The projects are Vscode and Atom.



Keras

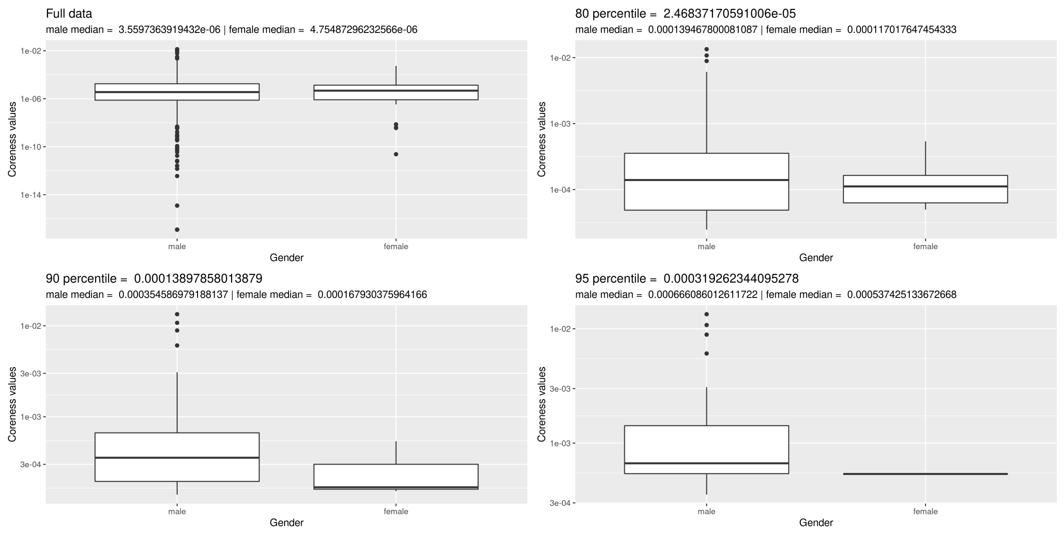


Nextcloud

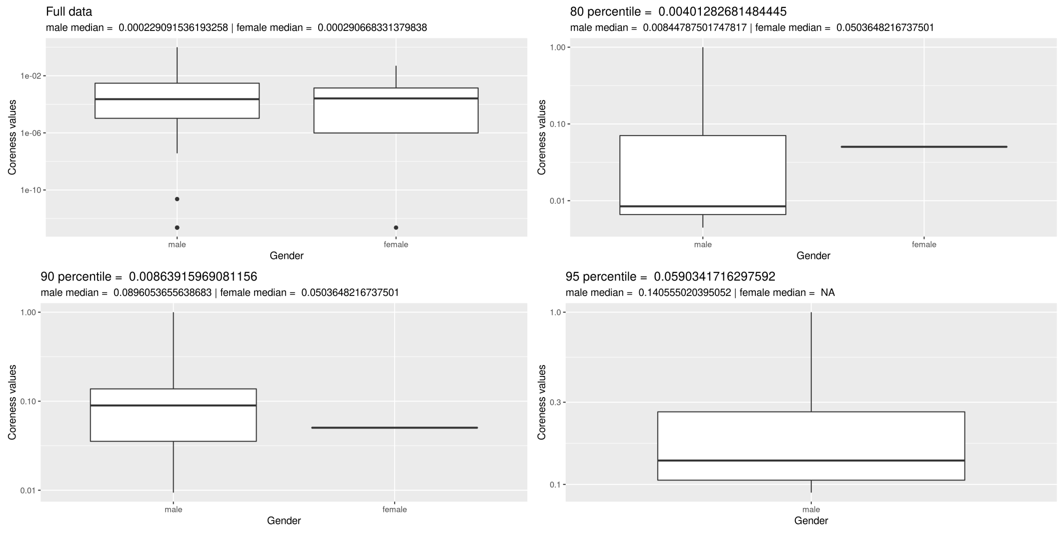


Vue

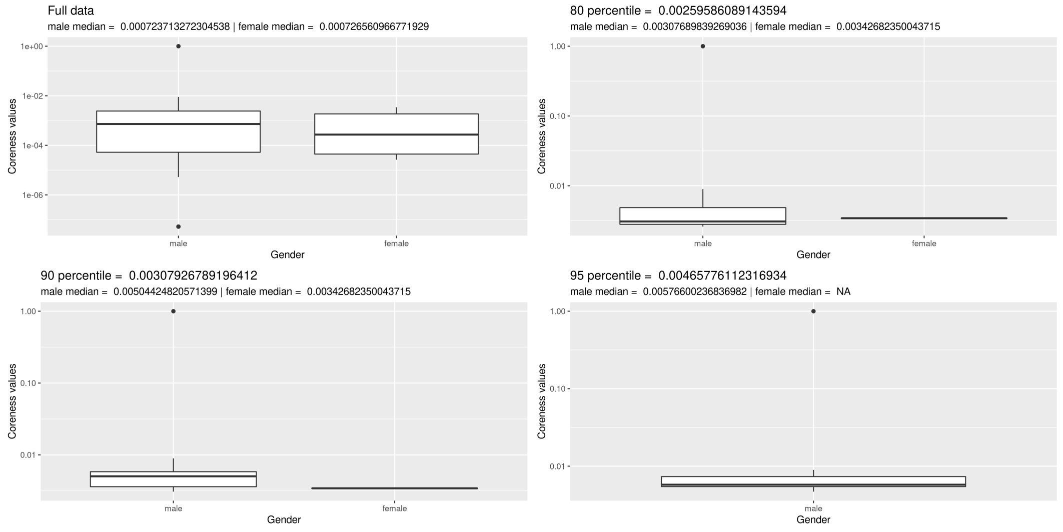
Figure A.8: Boxplots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is co-change. The data is shown on a logarithmic scale. The projects are KERAS,NEXTCLOUD, and VUE.



Three.js

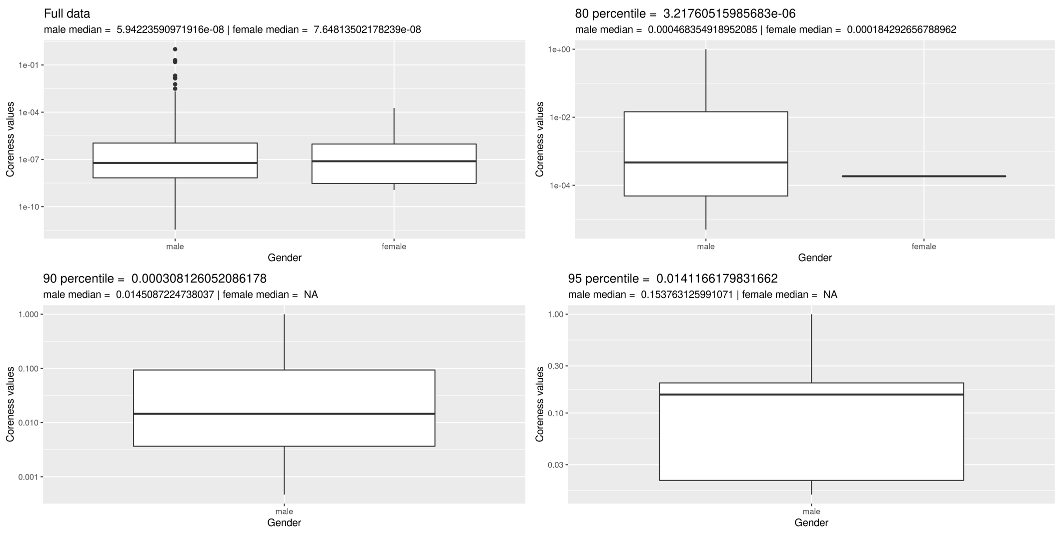


Deno

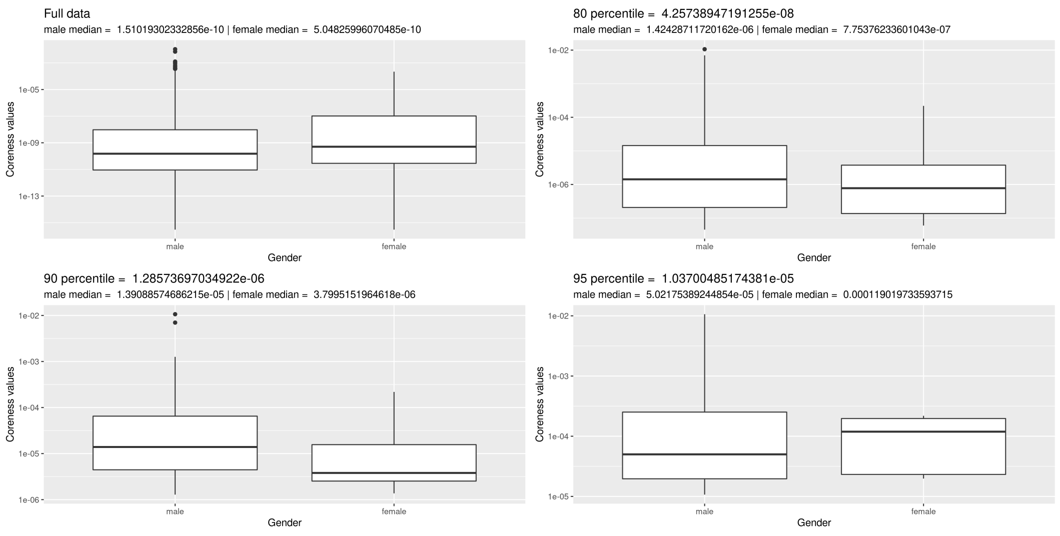


Reveal.js

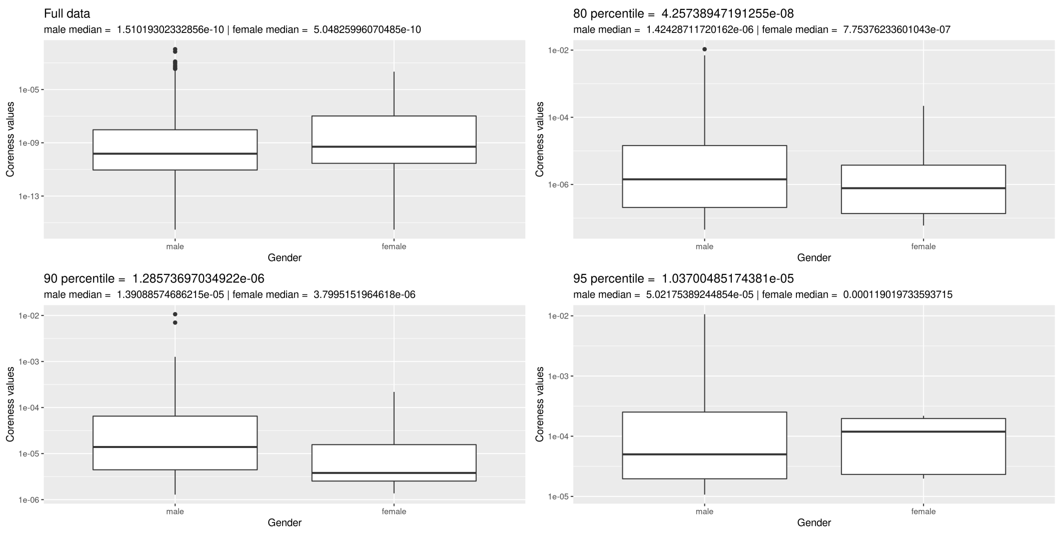
Figure A.9: Boxplots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is co-change. The data is shown on a logarithmic scale. The projects are Three.js, Deno, and Reveal.js.



Vscode

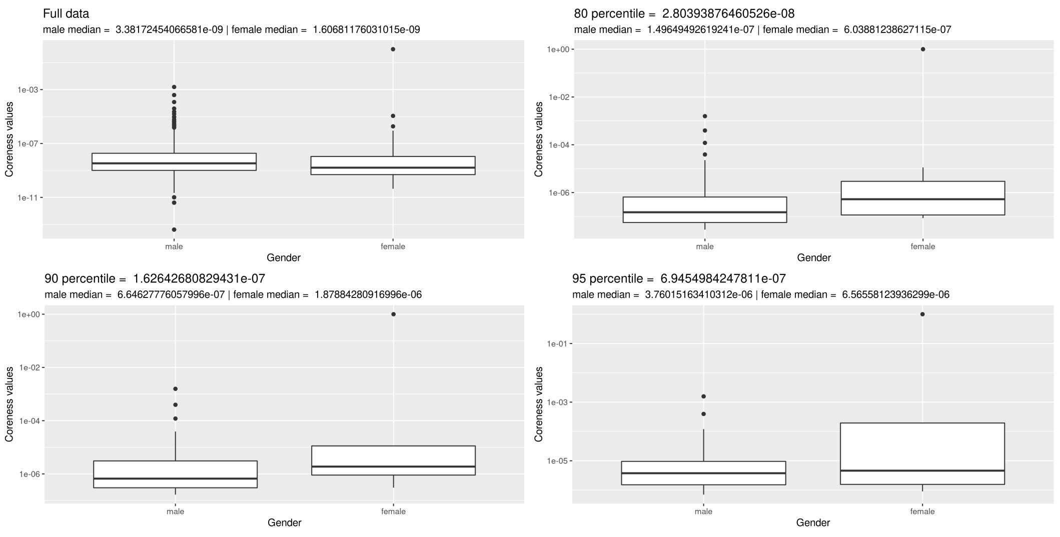


Tensorflow

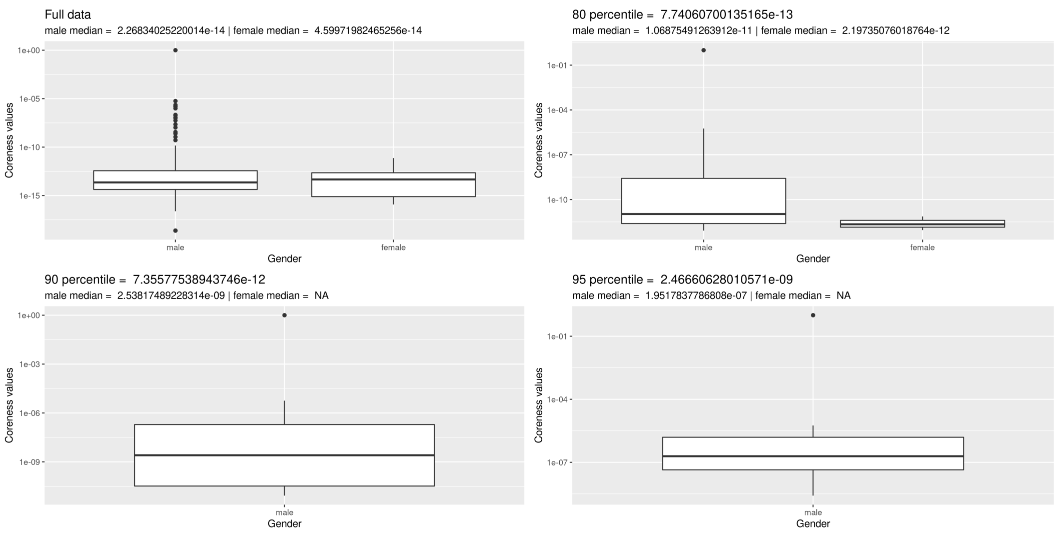


Atom

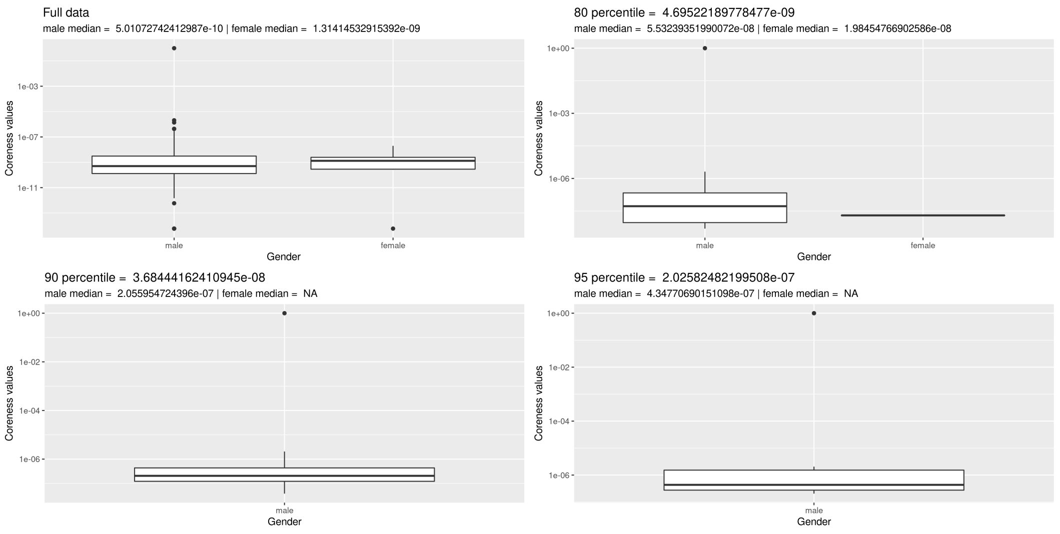
Figure A.10: Boxplots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is co-change. The data is shown on a logarithmic scale. The projects are VSCODE, TENSORFLOW, and ATOM.



Keras

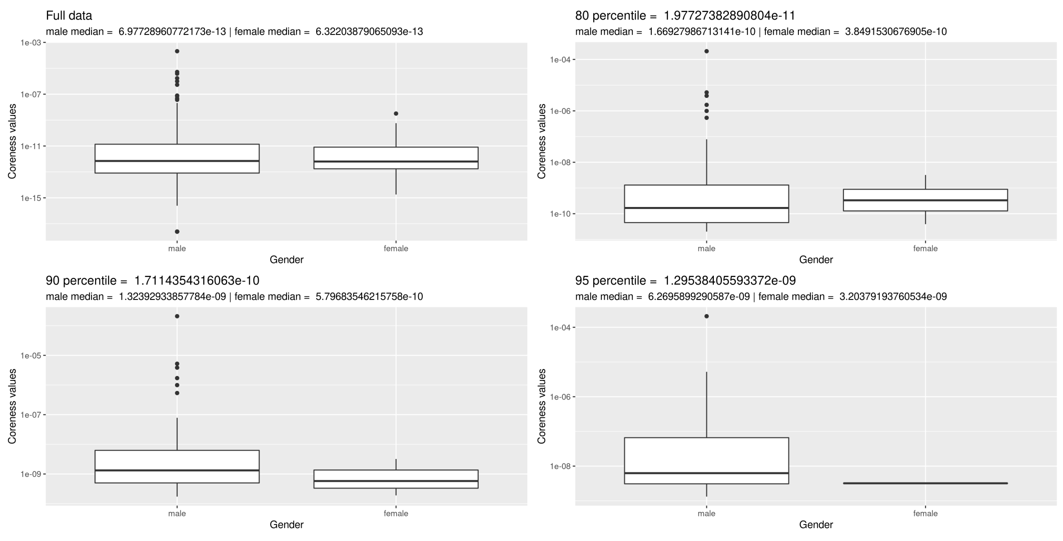


Nextcloud

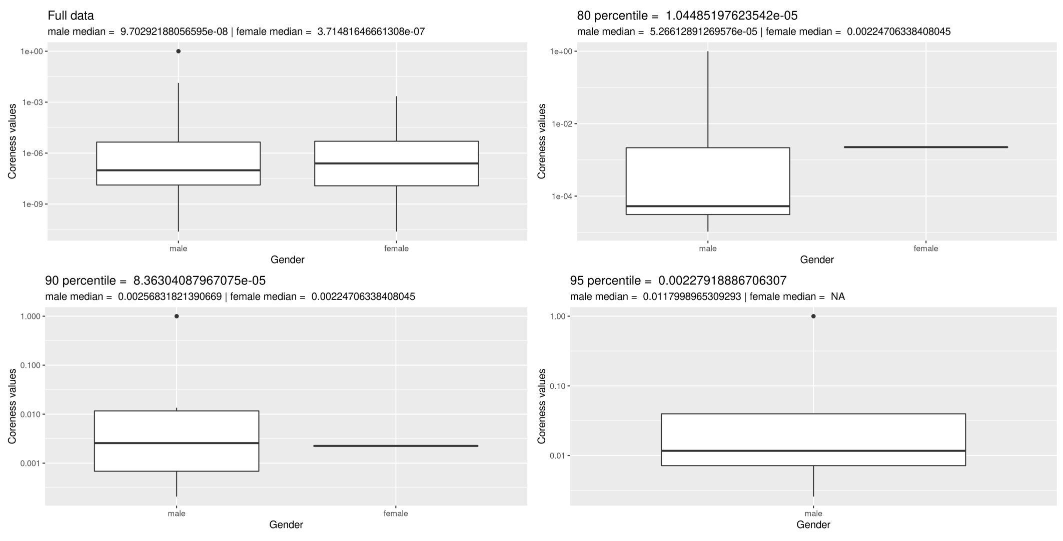


Vue

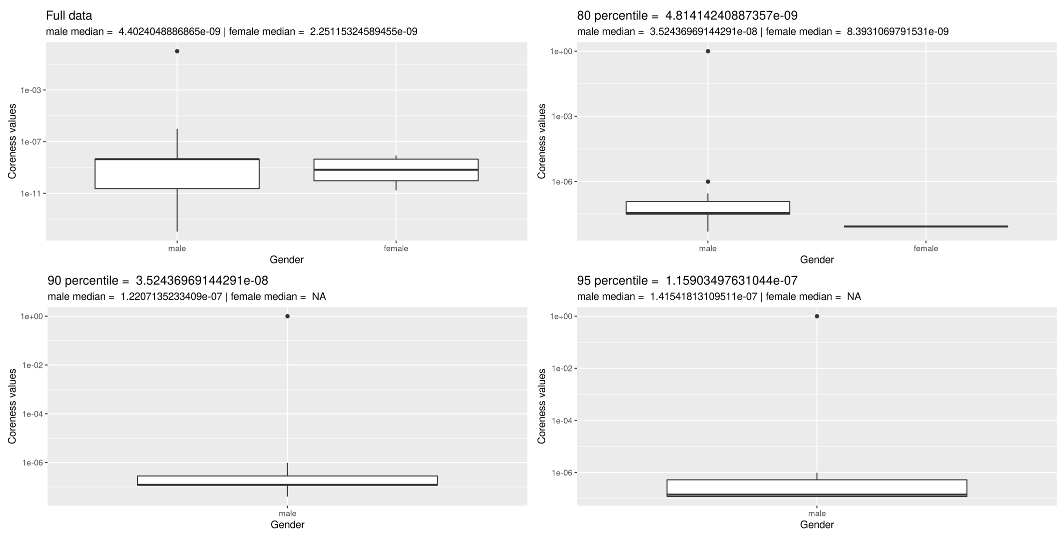
Figure A.11: Boxplots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is co-change. The data is shown on a logarithmic scale. The projects are KERAS, NEXTCLOUD, and VUE.



Three.js



Deno



Reveal.js

Figure A.12: Boxplots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is co-change. The data is shown on a logarithmic scale. The projects are Three.js, Deno, and Reveal.js.

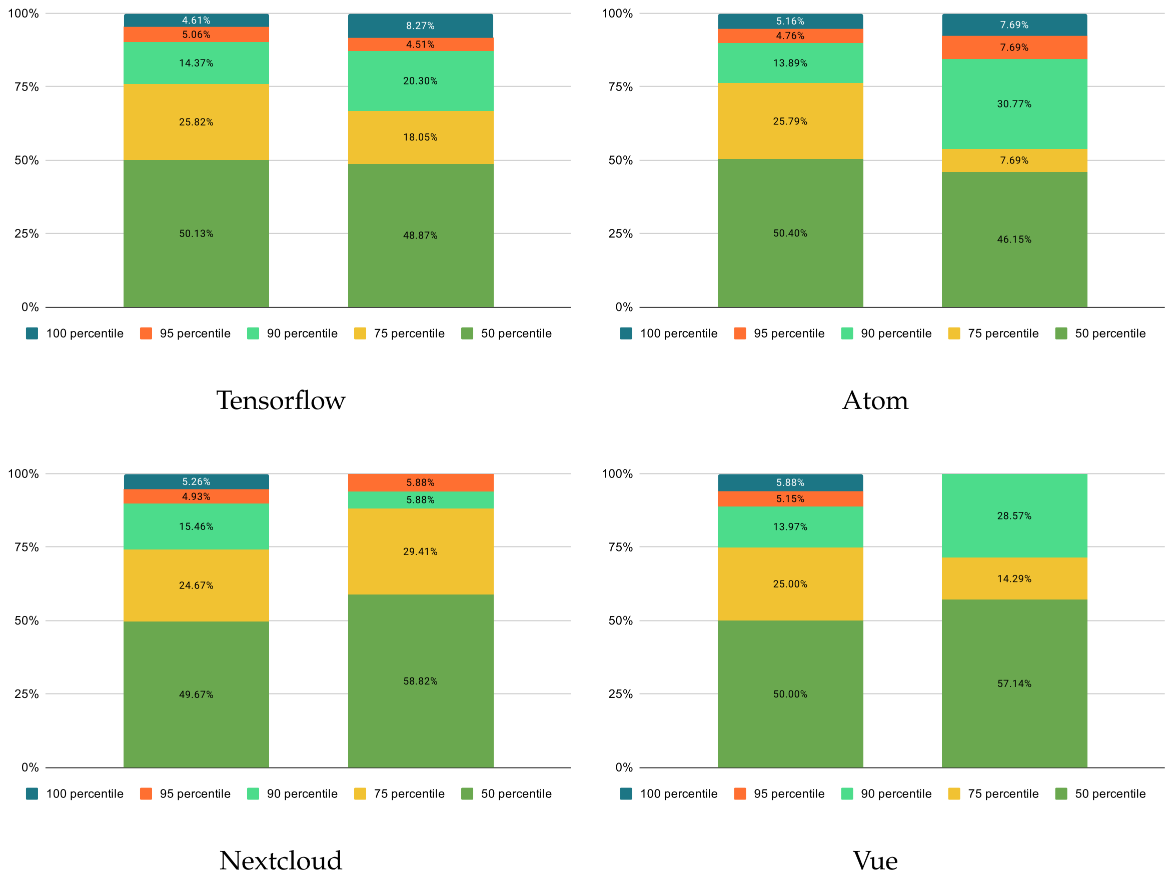
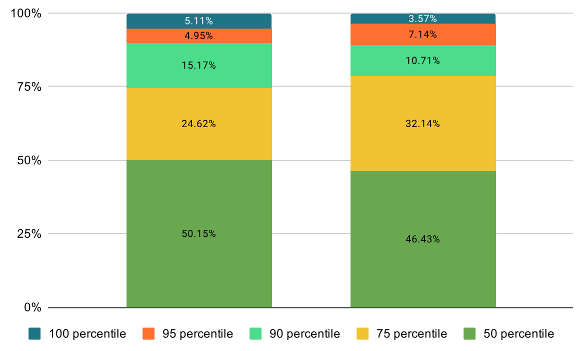
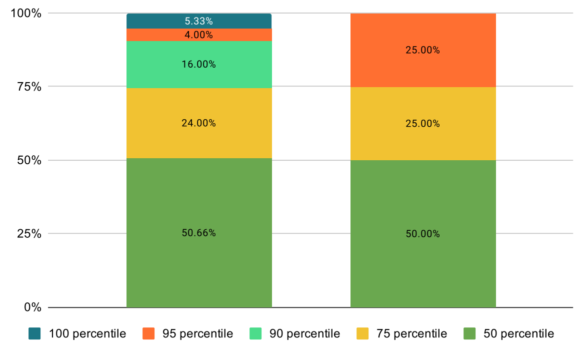


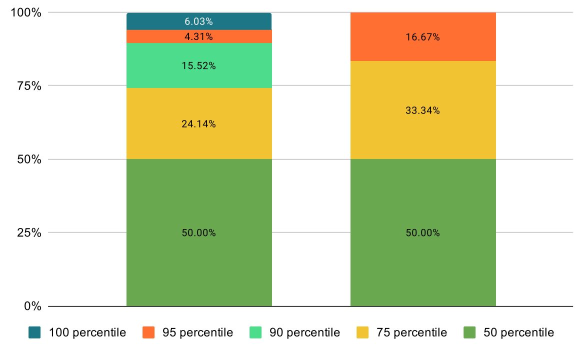
Figure A.17: Barplots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is co-change. The projects are Tensorflow, Atom, Nextcloud, and Vue.



Three.js

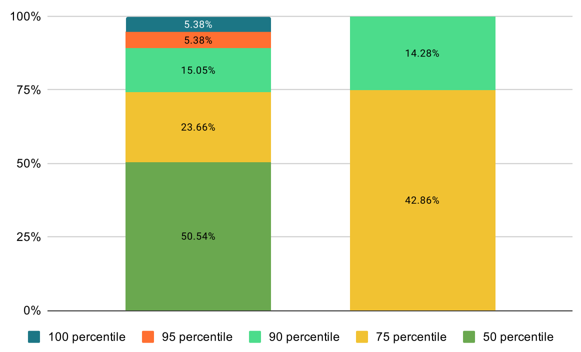


Deno

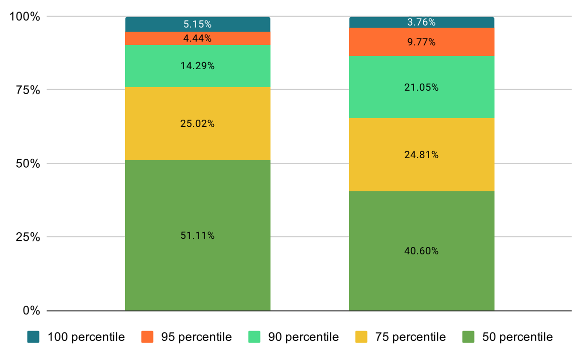


Reveal.js

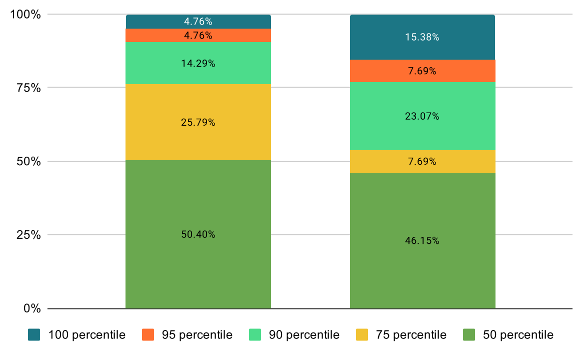
Figure A.21: Barplots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is co-change. The projects are THREE.JS, DENO, and REVEAL.JS.



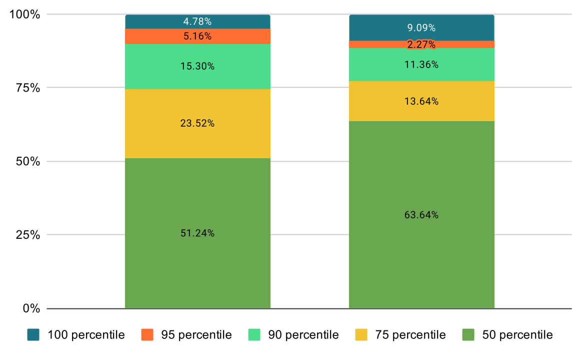
Vscode



Tensorflow



Atom



Keras

Figure A.26: Barplots for the coreness values of male and female developers. The coreness metric is hierarchy . The network type is co-change. The projects are Vscode, Tensorflow,Atom, and Keas.

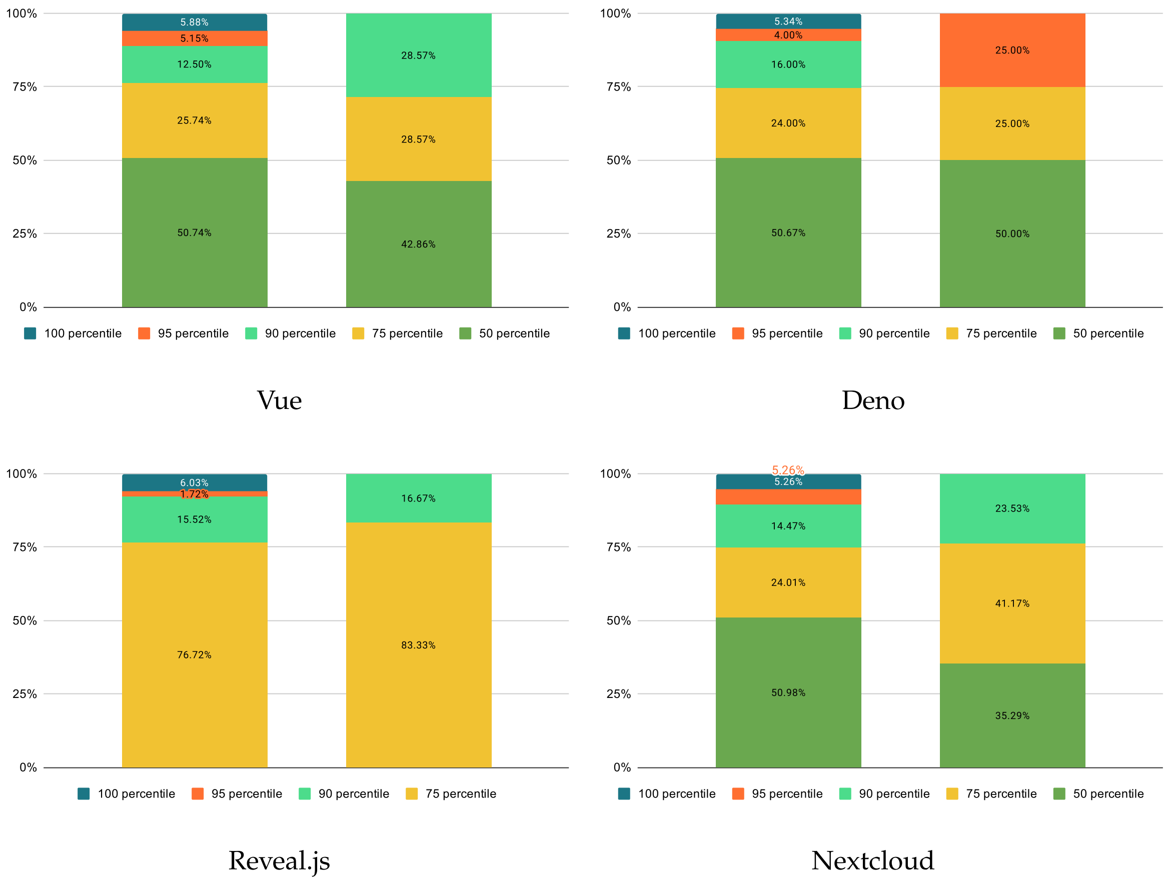
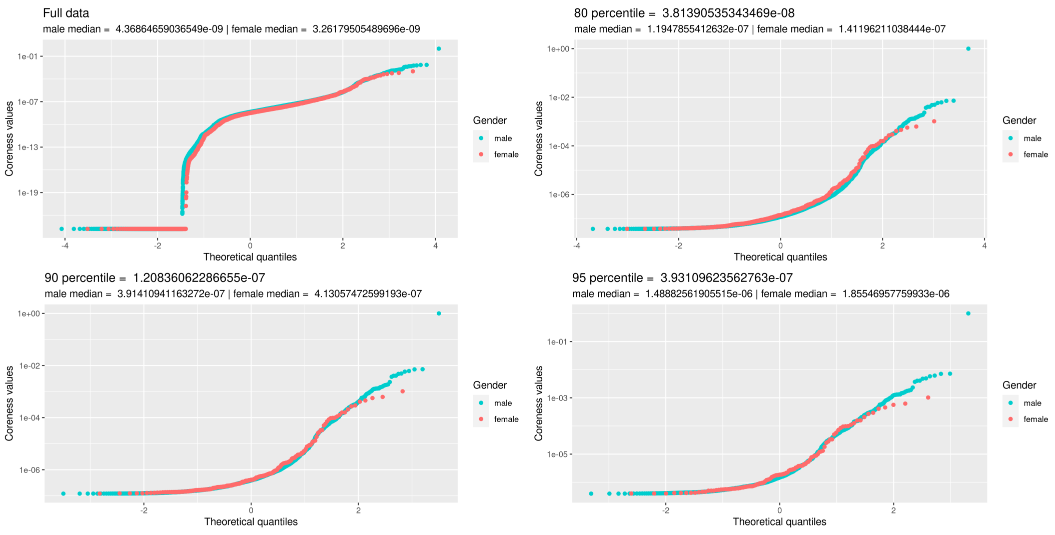
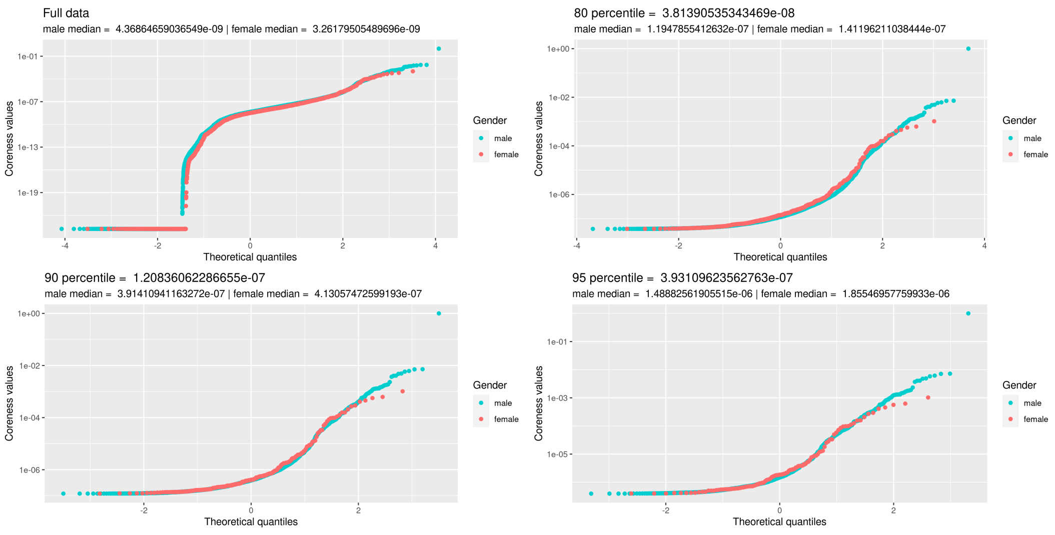


Figure A.31: Barplots for the coreness values of male and female developers. The coreness metric is hierarchy . The network type is co-change. The projects are Vue, Deno, Reveal.js, and Nextcloud.

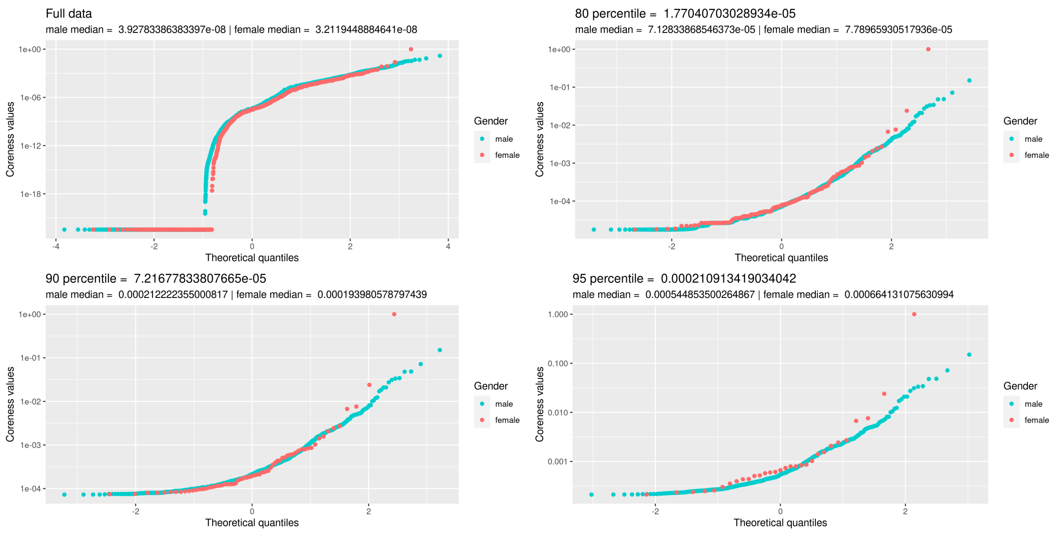


Tensorflow

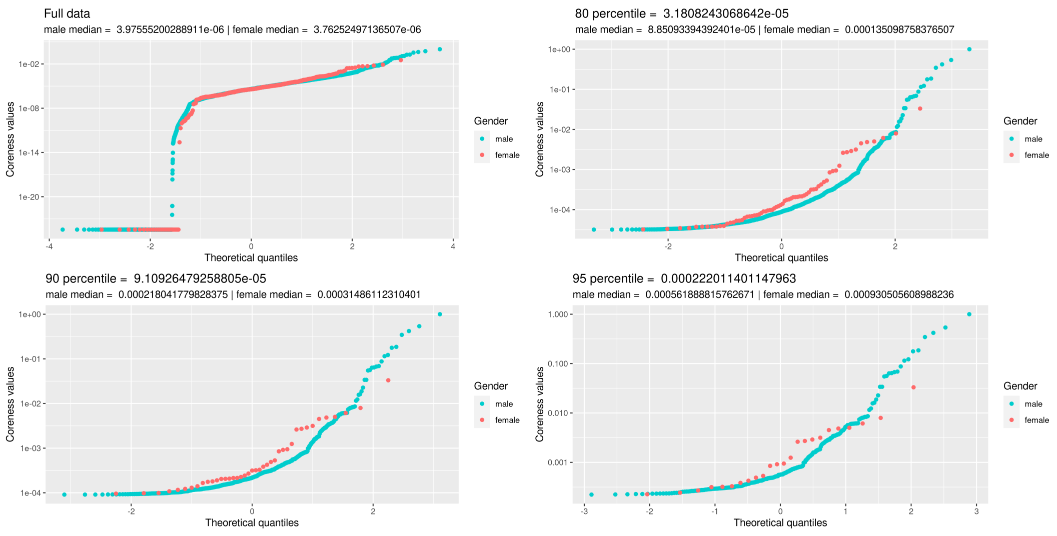


Atom

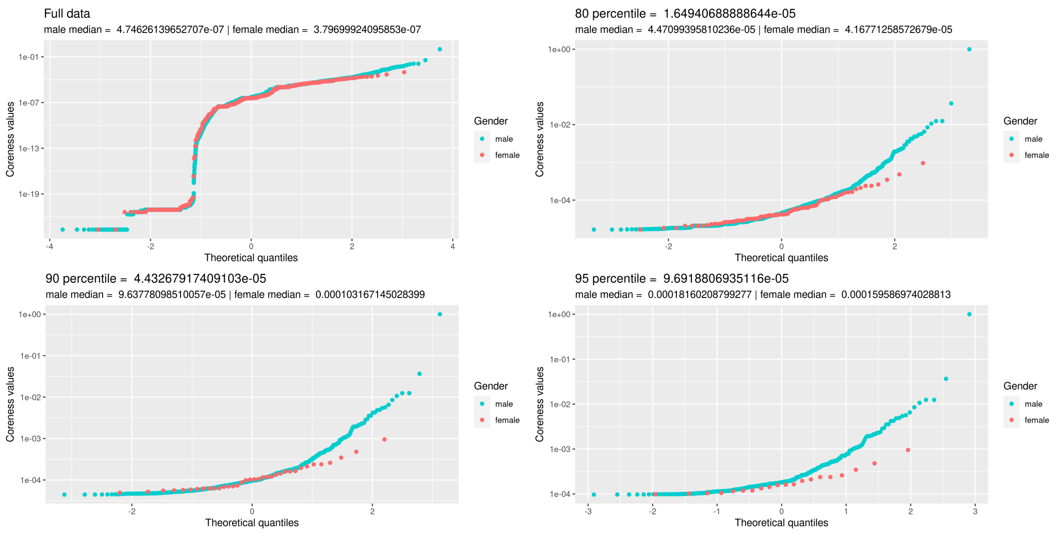
Figure A.32: QQ-plots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is issue. The data is shown on a logarithmic scale. The projects are TENSORFLOW and ATOM.



Keras

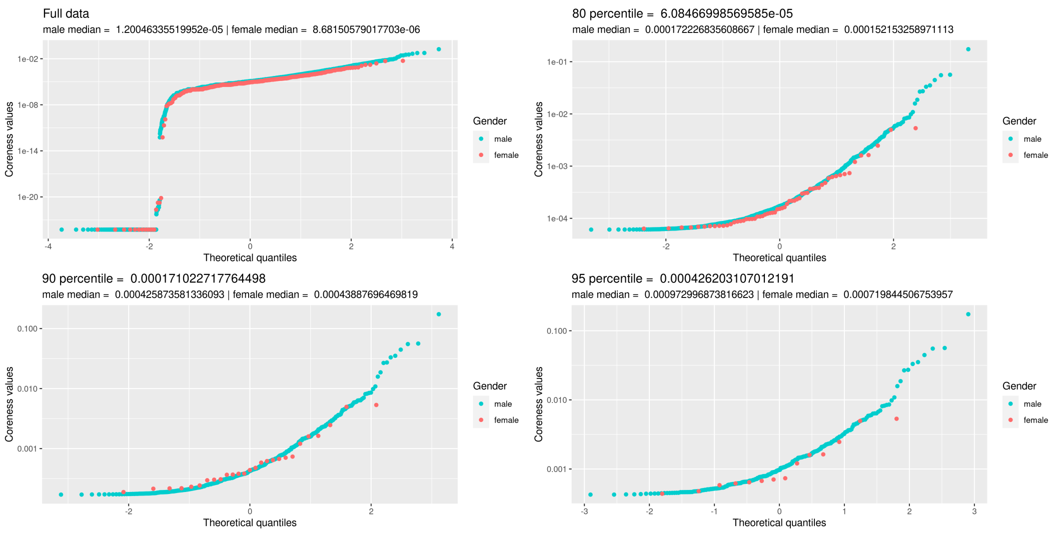


Nextcloud

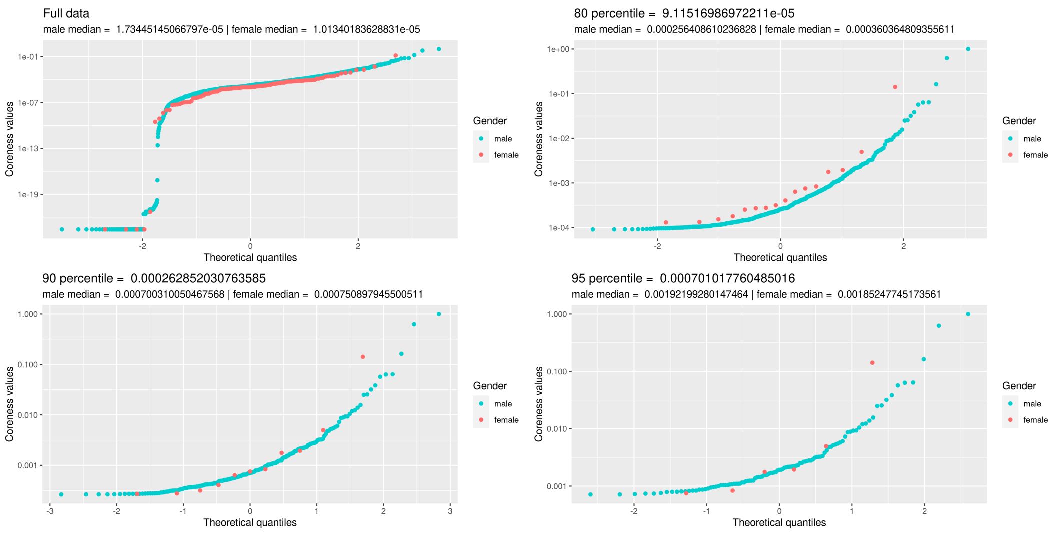


Vue

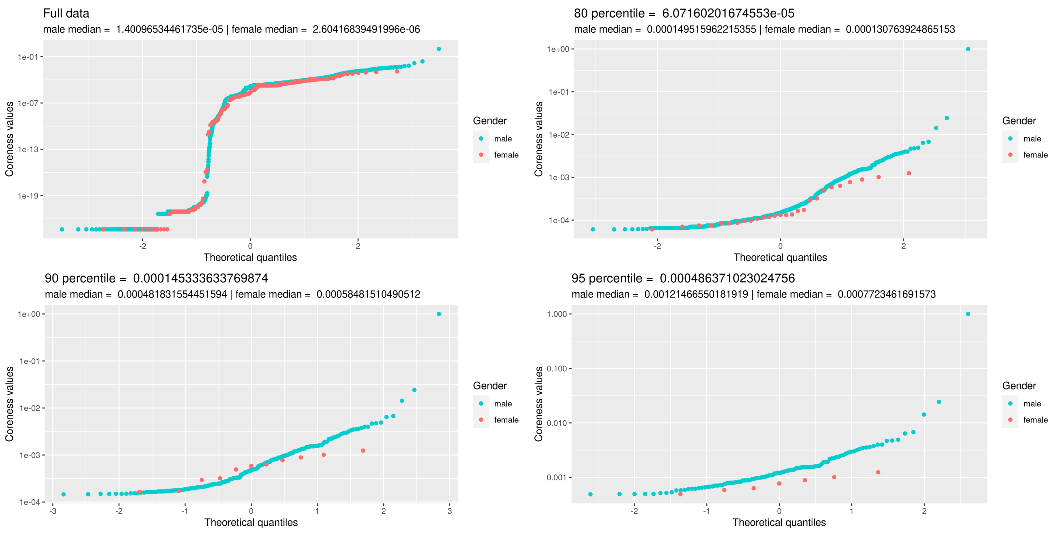
Figure A.33: QQ-plots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is issue. The data is shown on a logarithmic scale. The projects are KERAS, NEXTCLOUD, and VUE.



Three.js

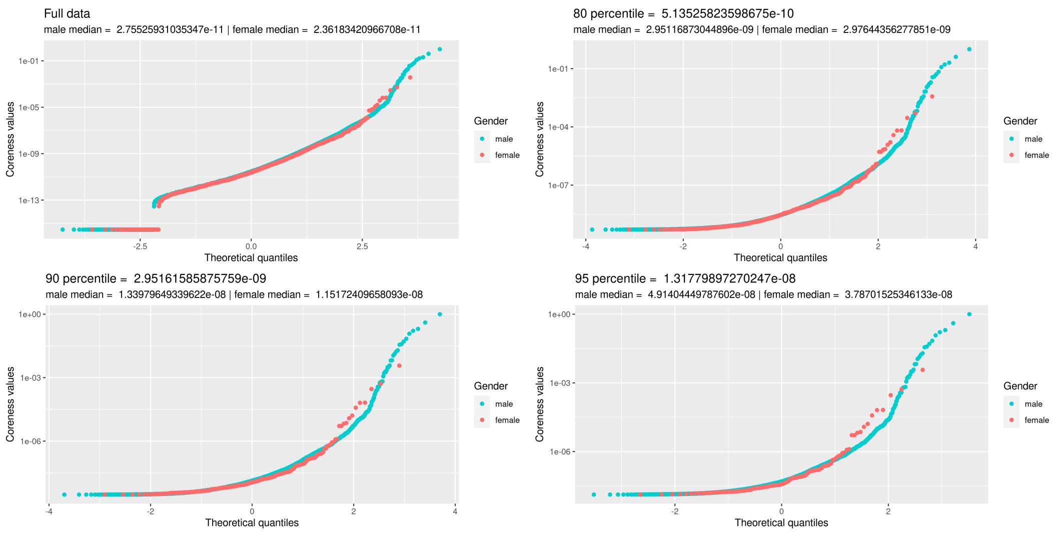


Deno

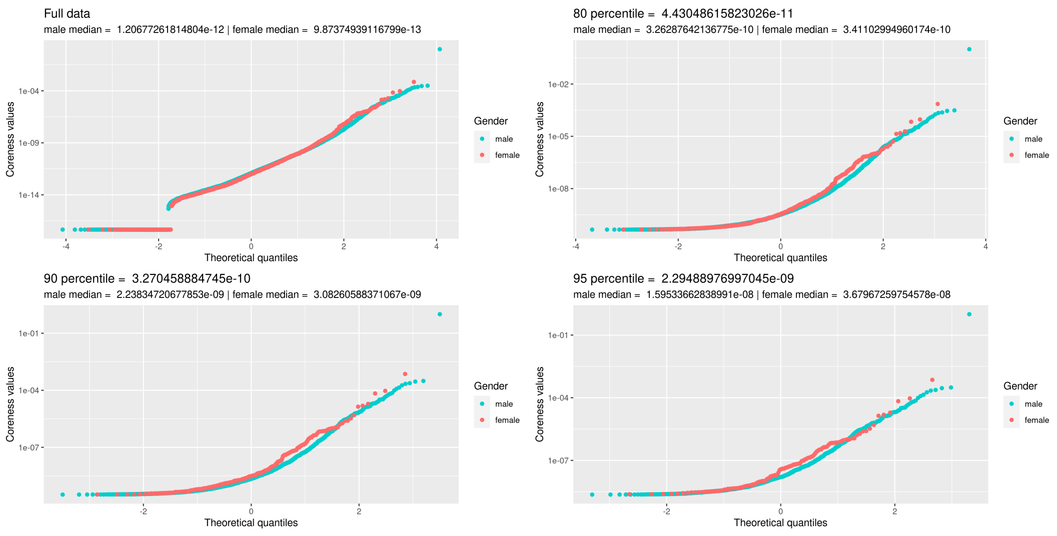


Reveal.js

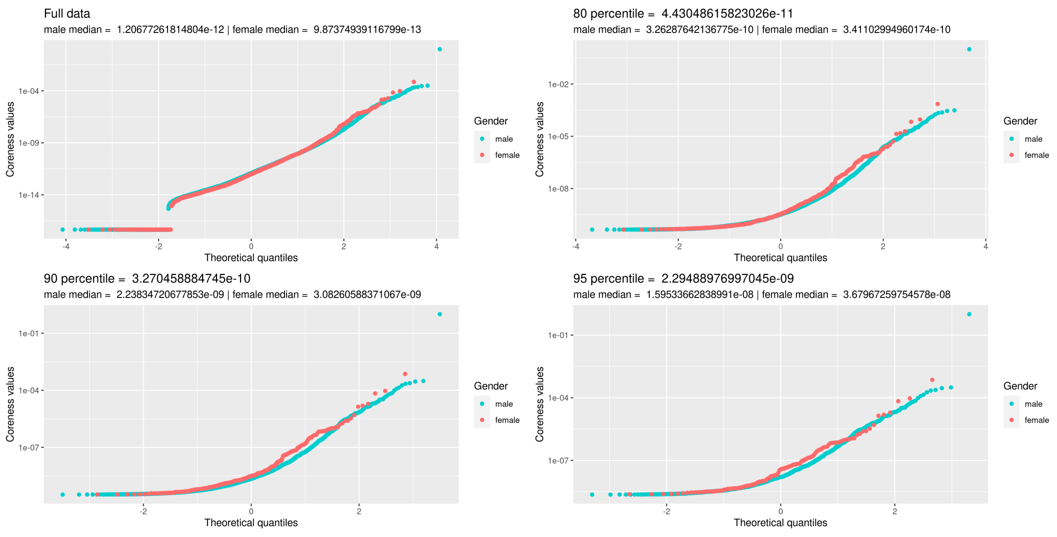
Figure A.34: QQ-plots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is issue. The data is shown on a logarithmic scale. The projects are Three.js, Deno, and Reveal.js.



Vscode

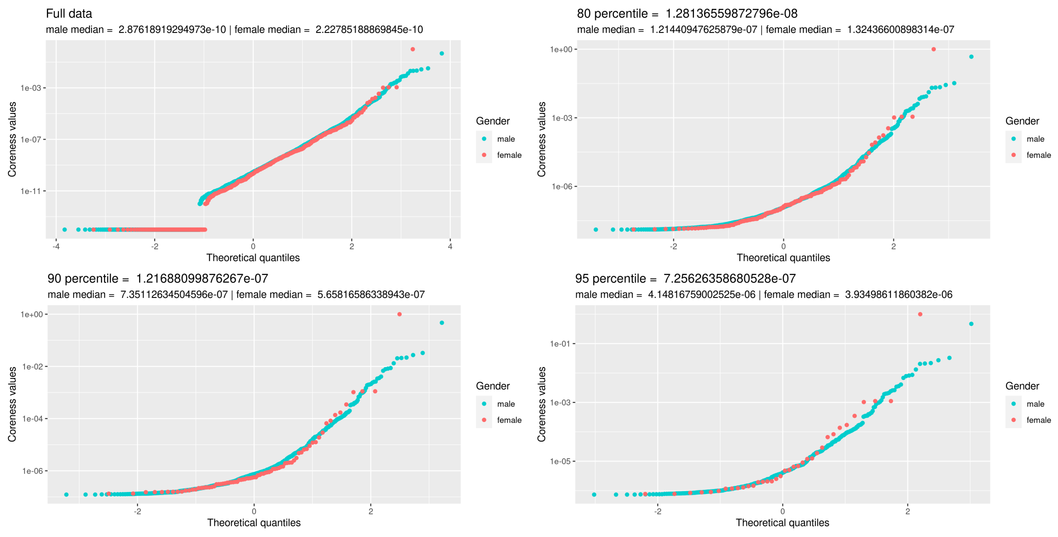


Tensorflow

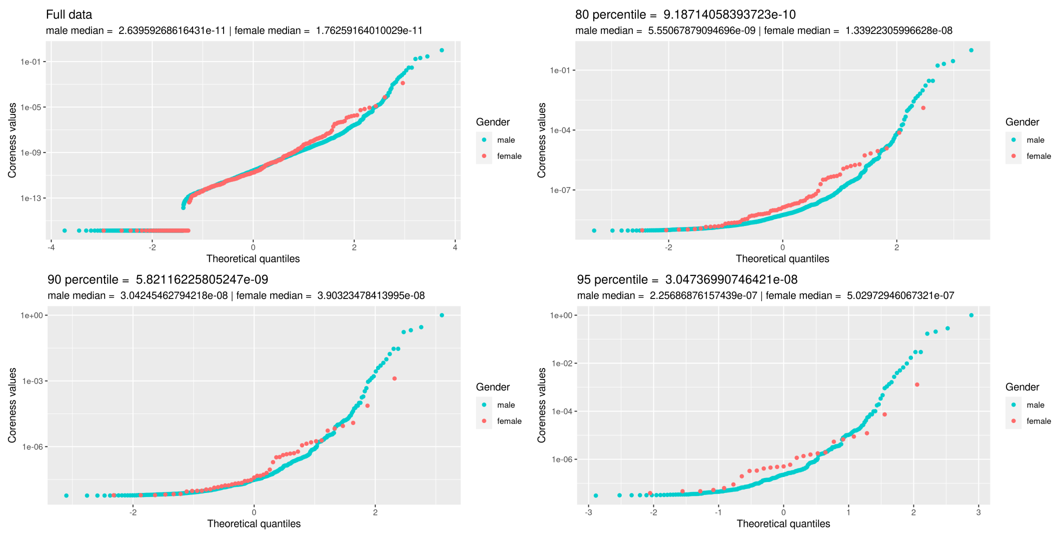


Atom

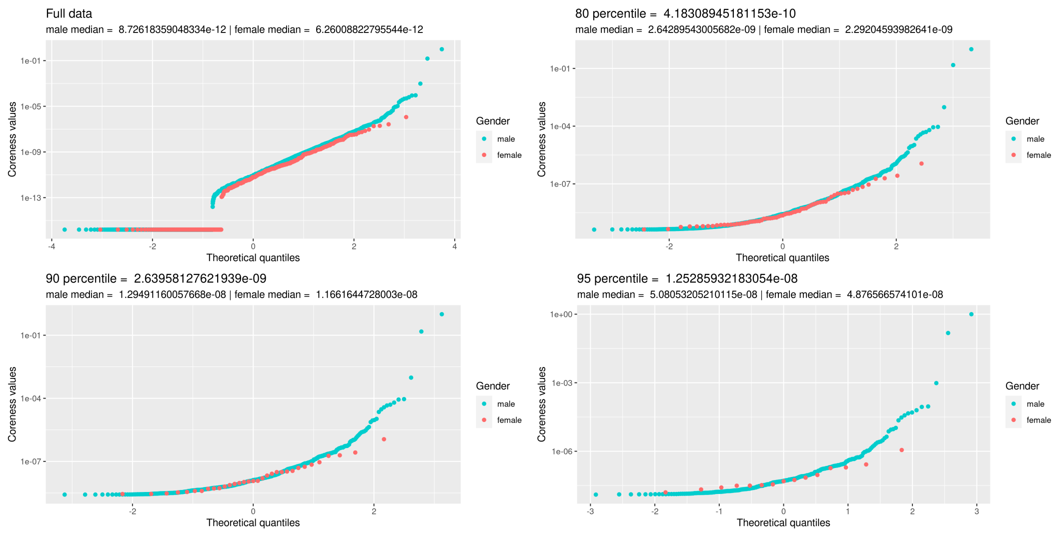
Figure A.35: QQ-plots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is issue. The data is shown on a logarithmic scale. The projects are Vscode, Tensorflow, and Atom.



Keras

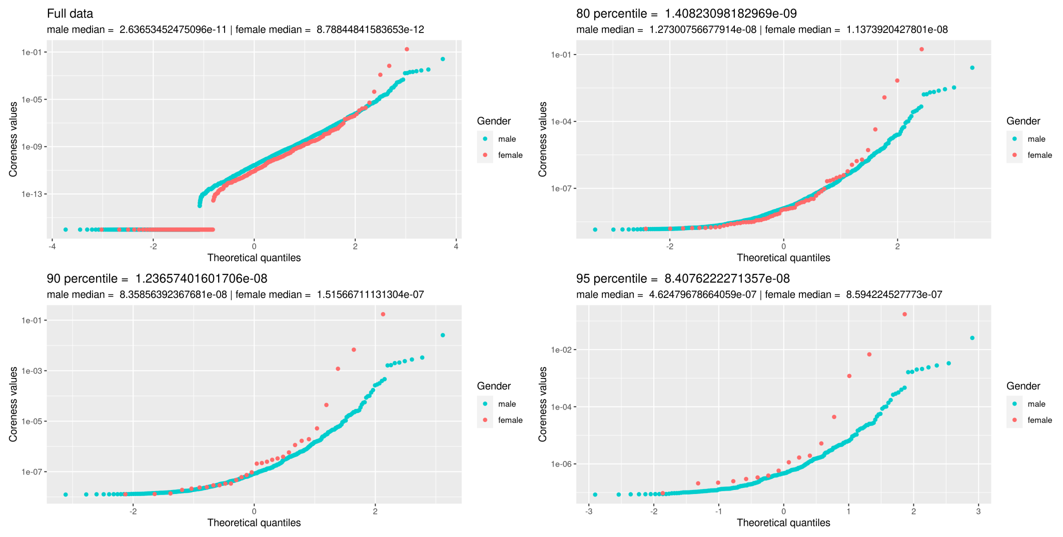


Nextcloud

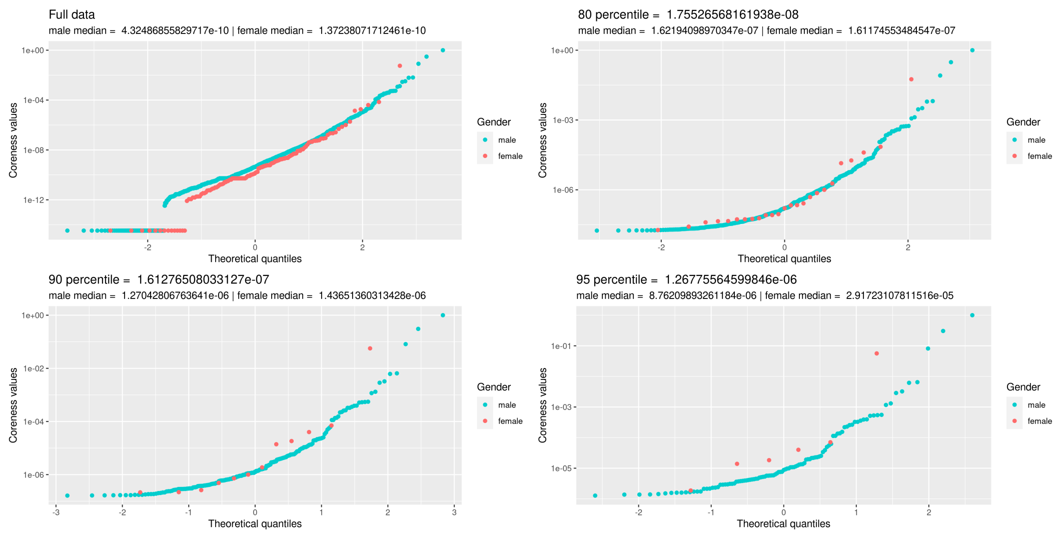


Vue

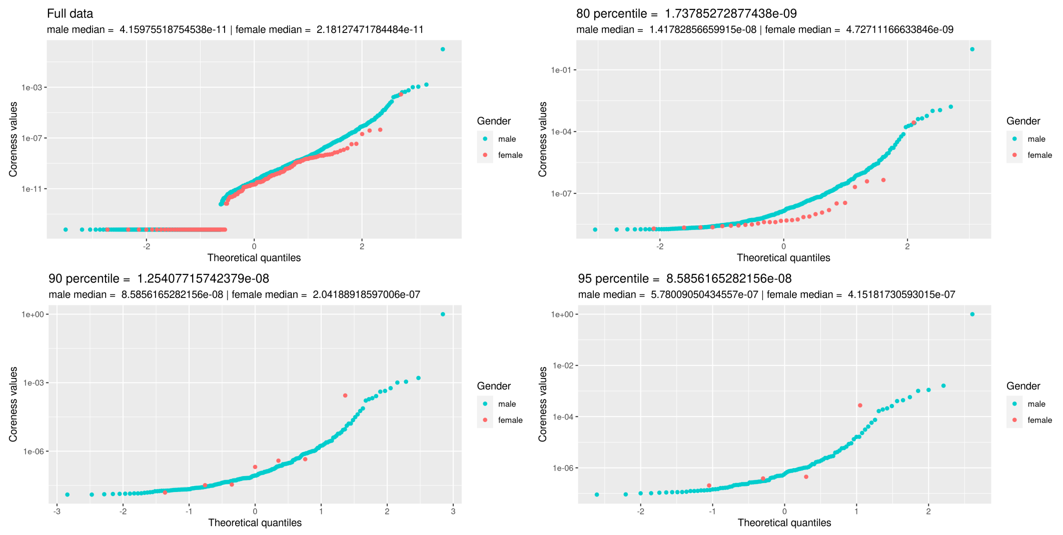
Figure A.36: QQ-plots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is issue. The data is shown on a logarithmic scale. The projects are KERAS, NEXTCLOUD, and VUE.



Three.js

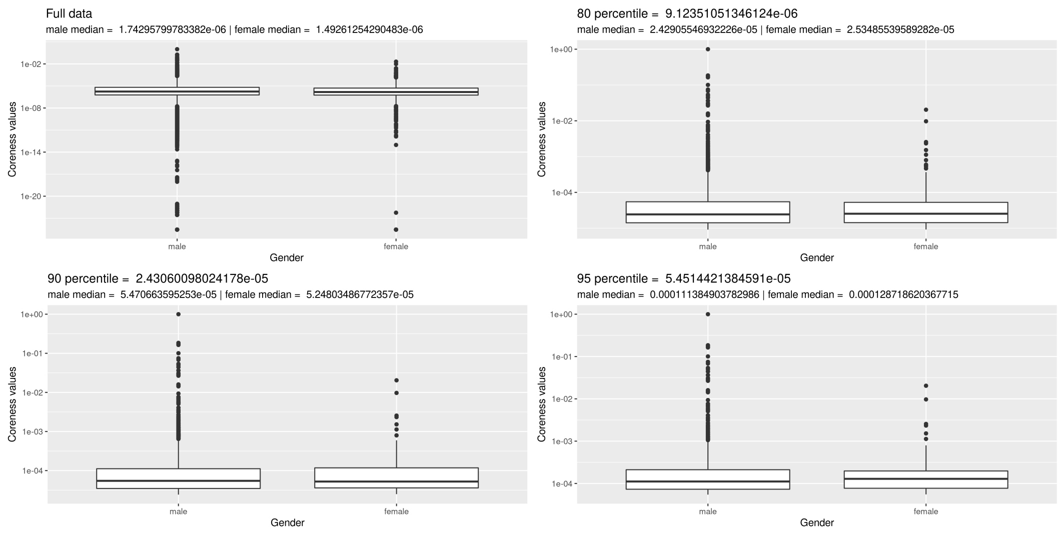


Deno

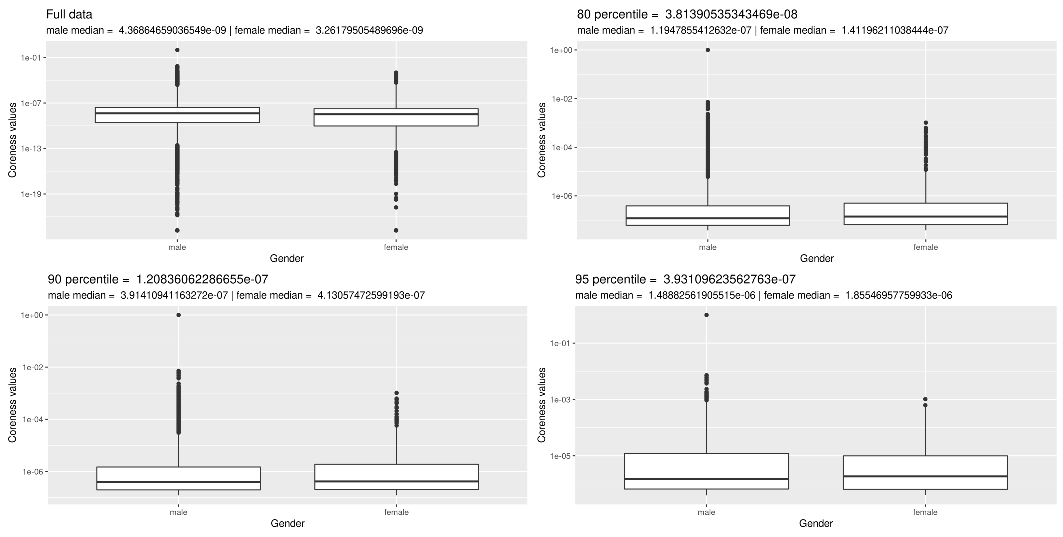


Reveal.js

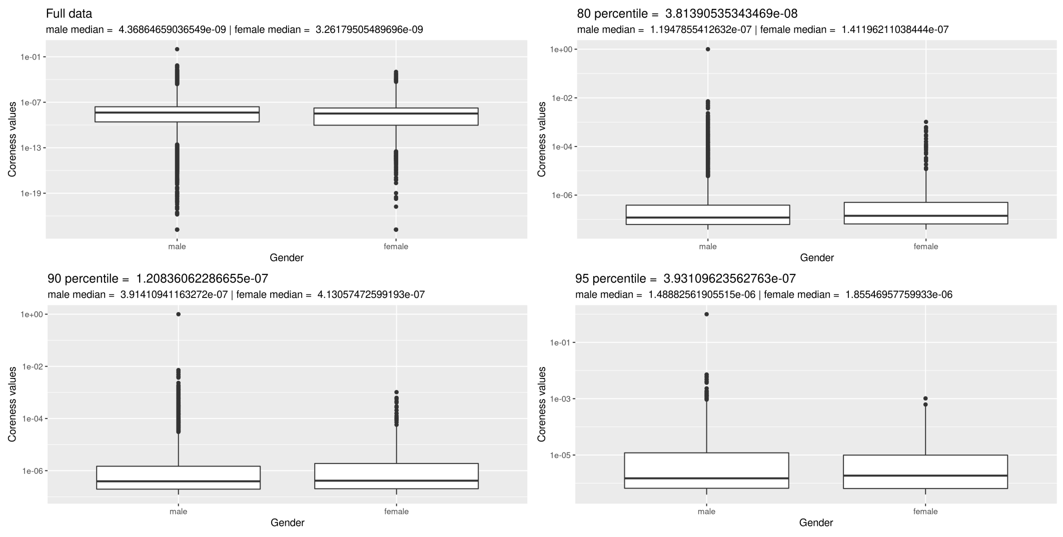
Figure A.37: QQ-plots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is issue. The data is shown on a logarithmic scale. The projects are Three.js, Deno, and Reveal.js.



Vscode

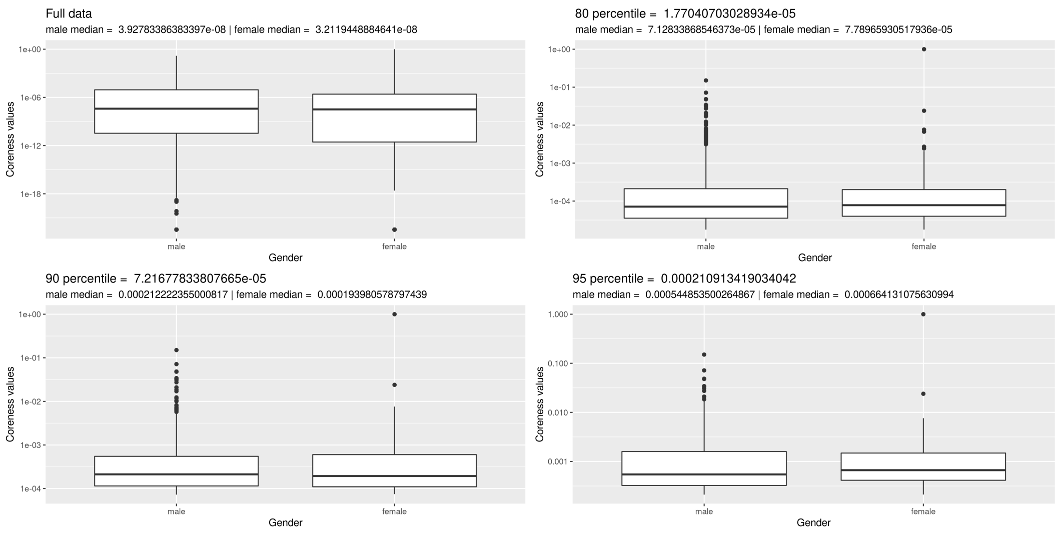


Tensorflow

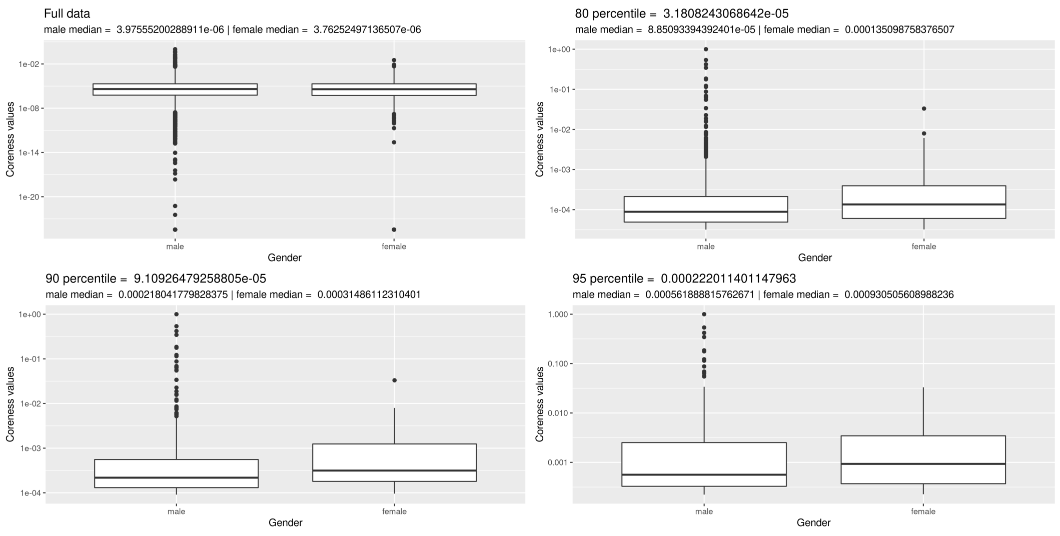


Atom

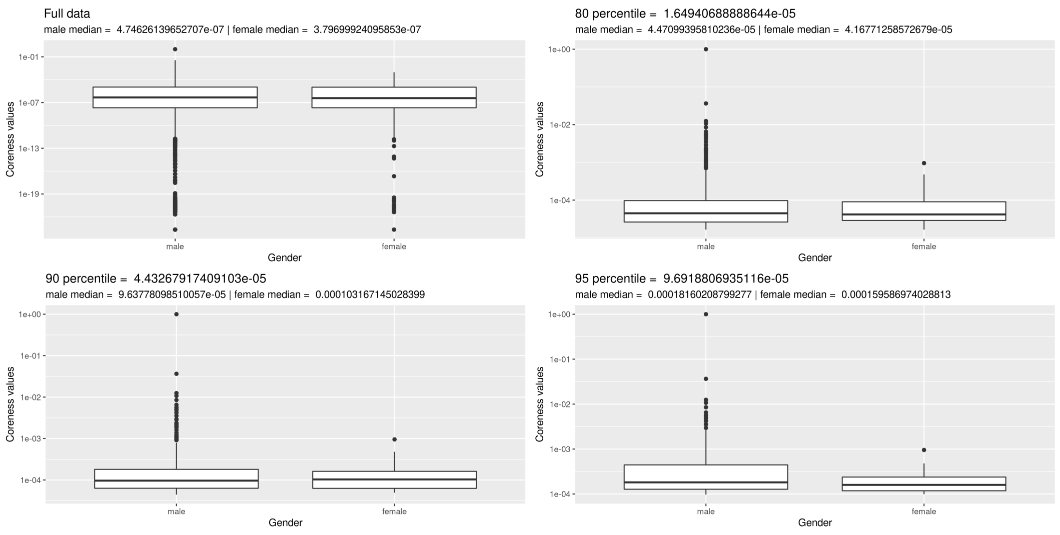
Figure A.38: Boxplots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is issue. The data is shown on a logarithmic scale. The projects are VSCODE, TENSORFLOW, and ATOM.



Keras

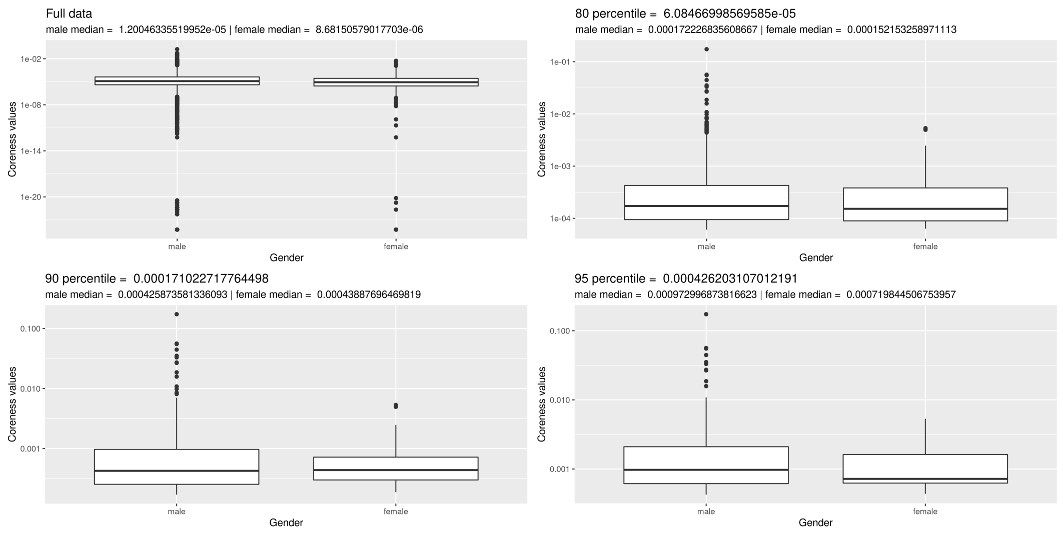


Nextcloud

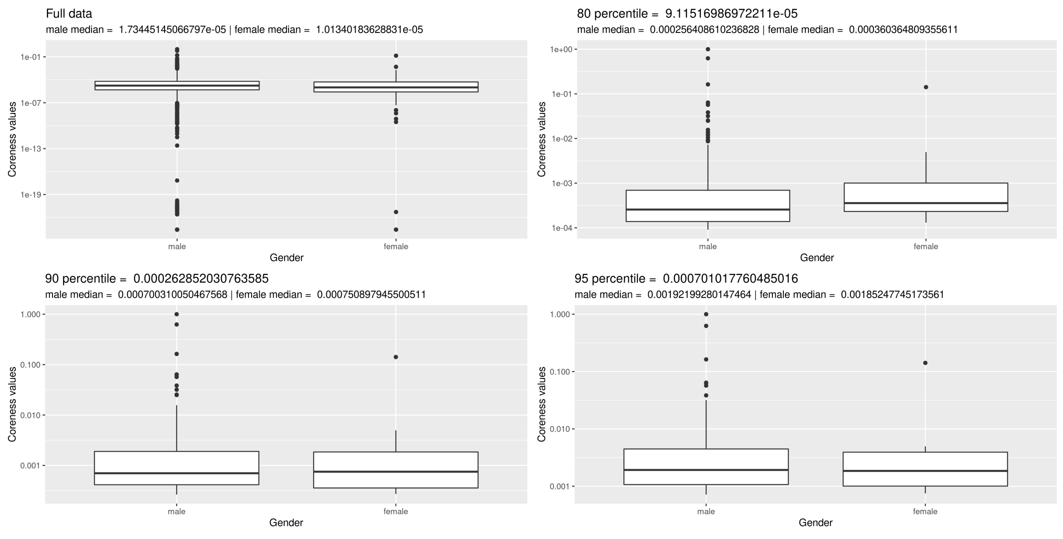


Vue

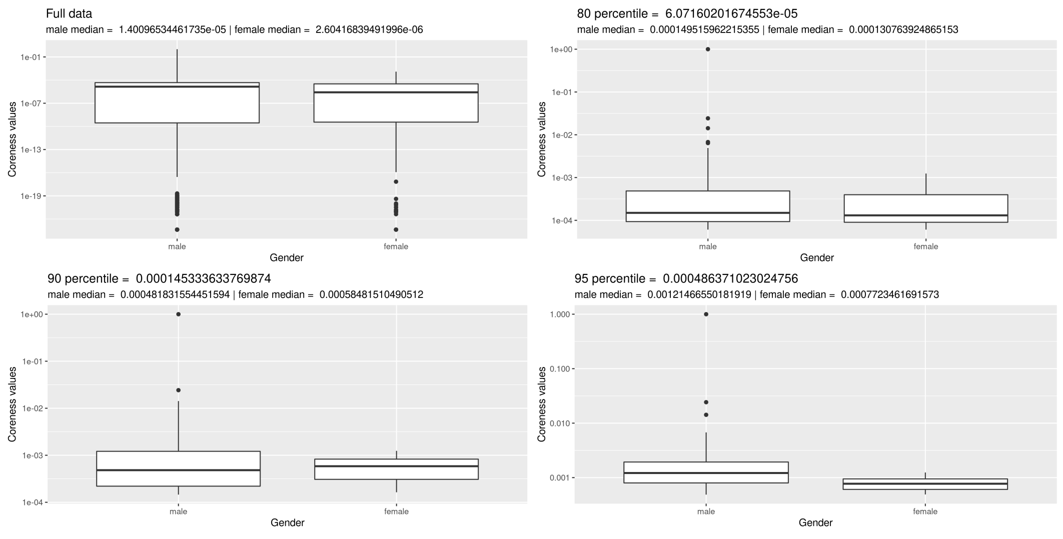
Figure A.39: Boxplots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is issue. The data is shown on a logarithmic scale. The projects are KERAS, NEXTCLOUD, and VUE.



Three.js

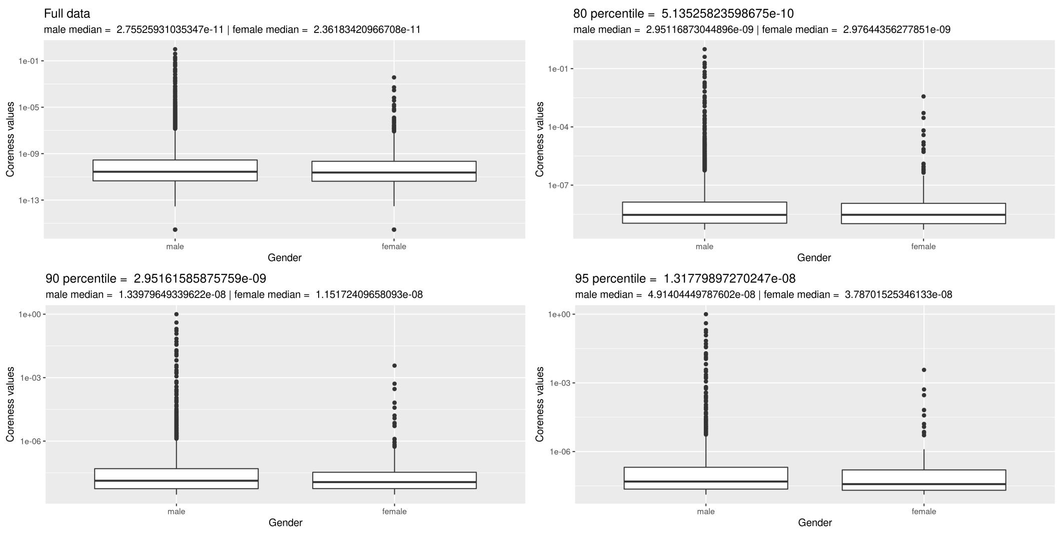


Deno

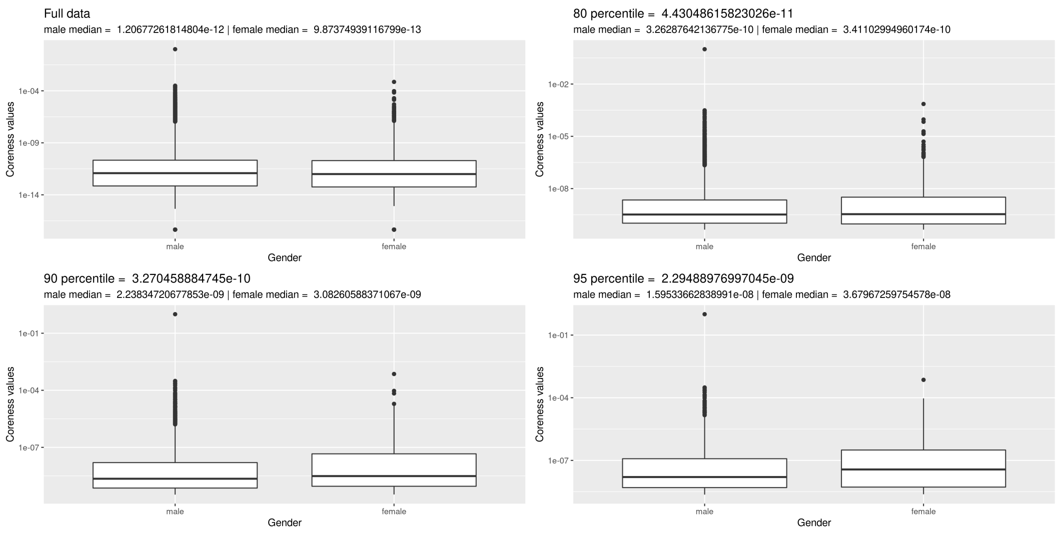


Reveal.js

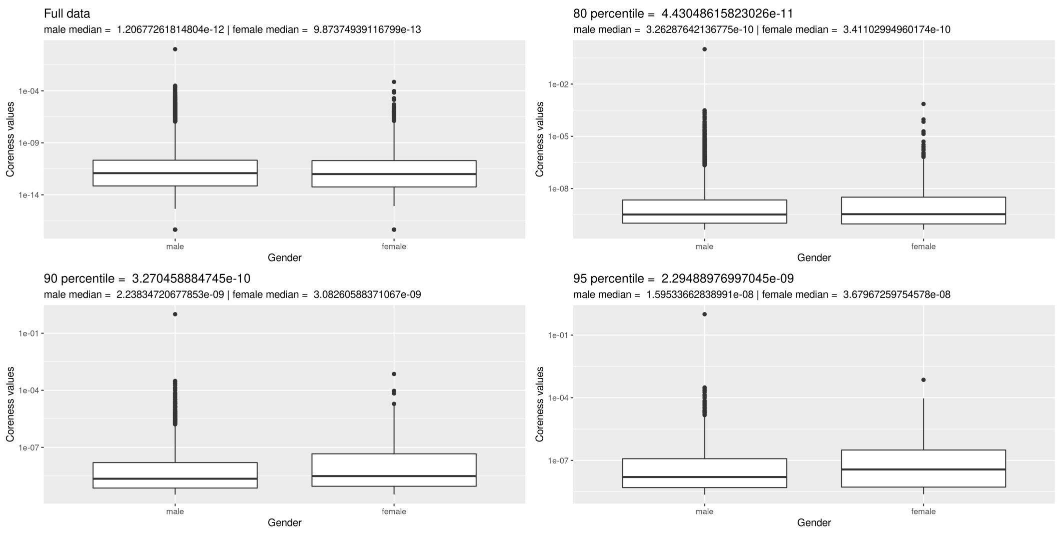
Figure A.40: Boxplots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is issue. The data is shown on a logarithmic scale. The projects are Three.js, Deno, and Reveal.js.



Vscode

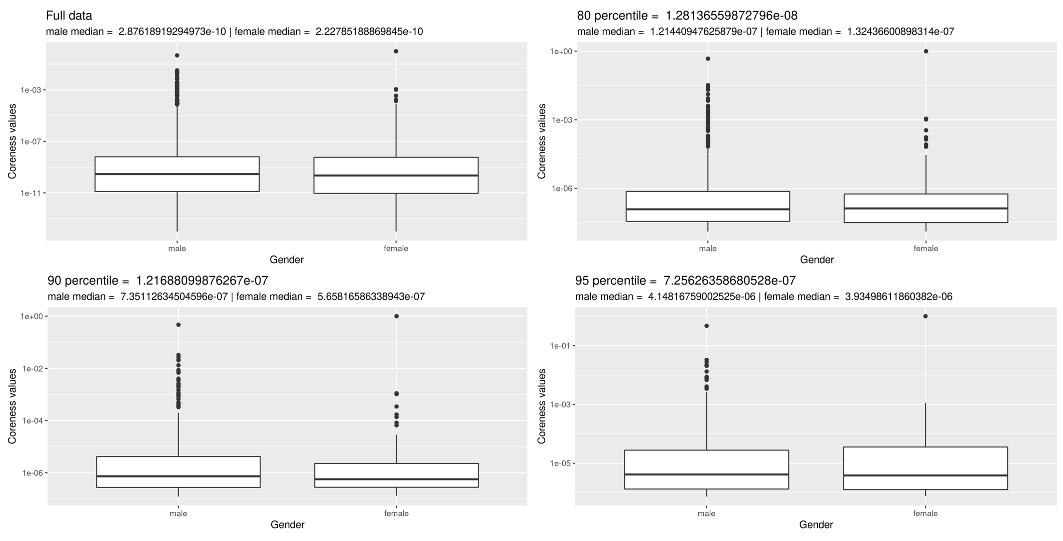


Tensorflow

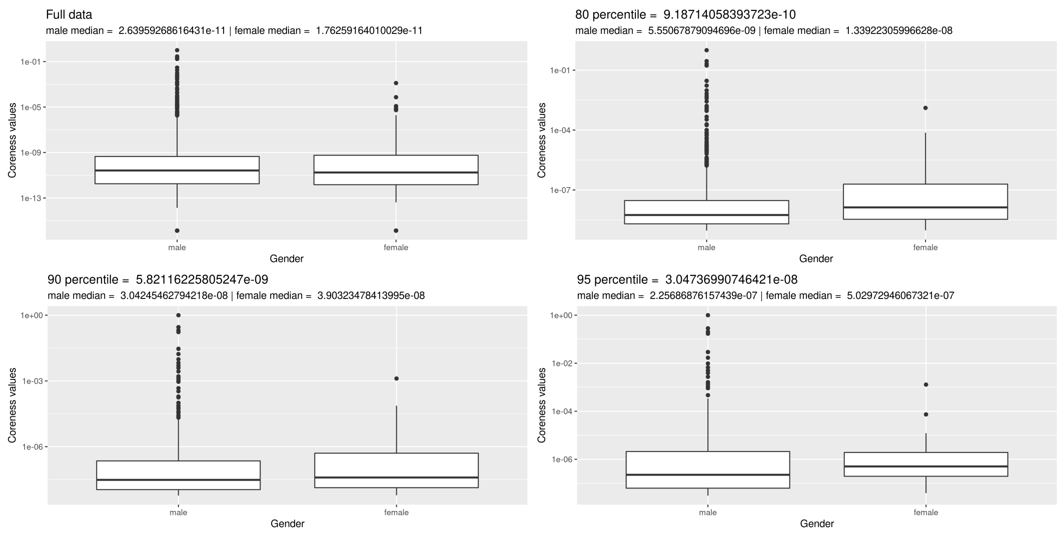


Atom

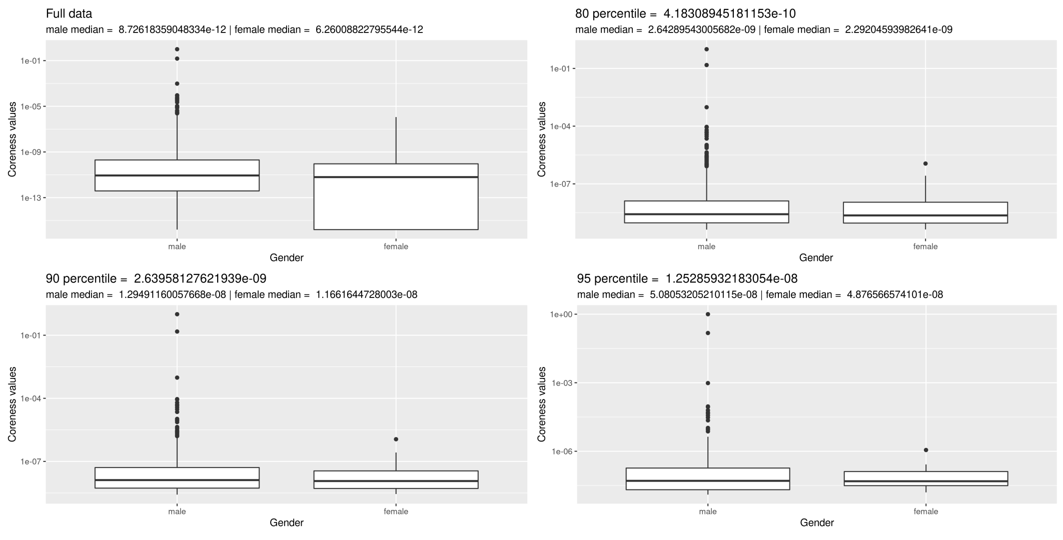
Figure A.41: Boxplots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is issue. The data is shown on a logarithmic scale. The projects are Vscode, Tensorflow, and Atom.



Keras

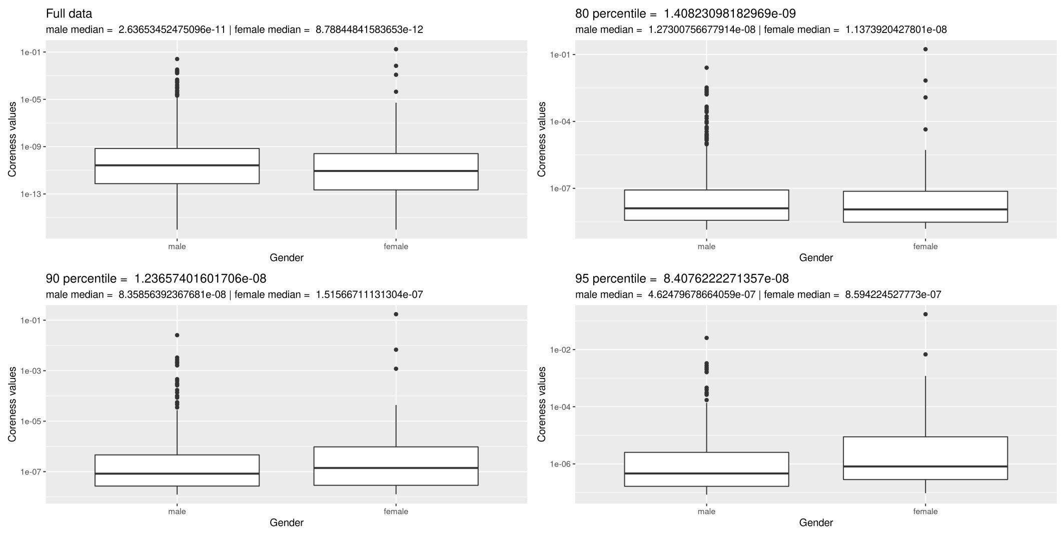


Nextcloud

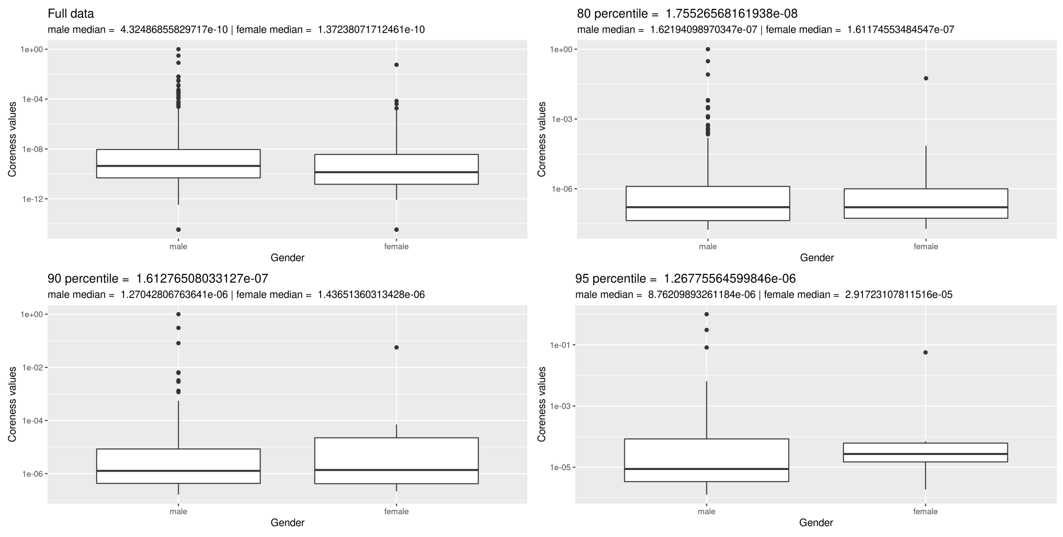


Vue

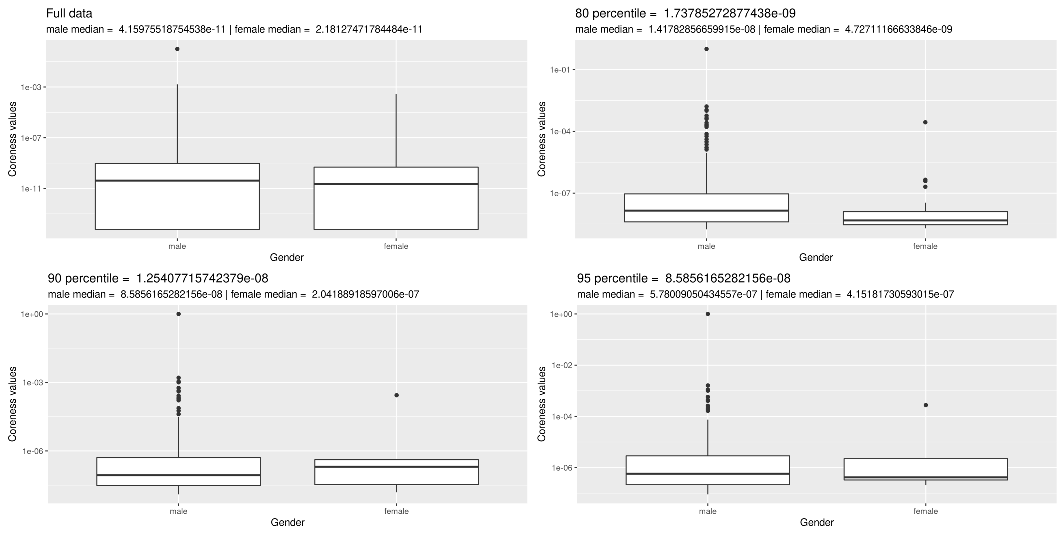
Figure A.42: Boxplots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is issue. The data is shown on a logarithmic scale. The projects are KERAS, NEXTCLOUD, and VUE.



Three.js

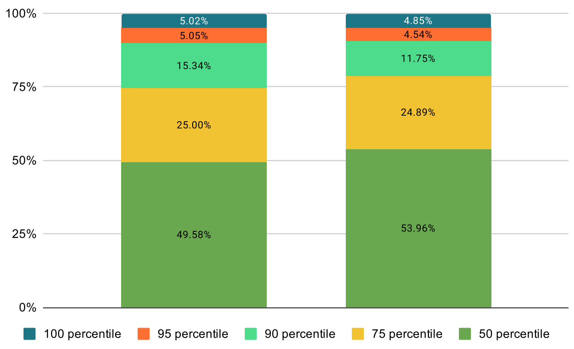


Deno

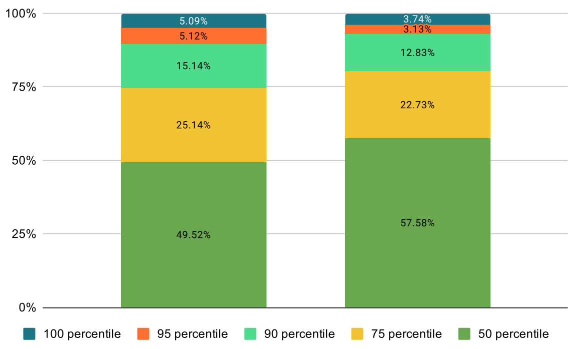


Reveal.js

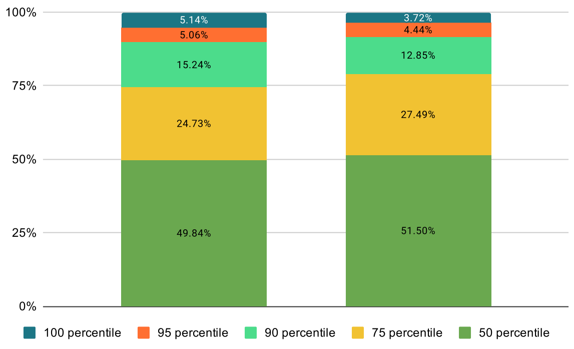
Figure A.43: Boxplots for the coreness values of male and female developers. The coreness metric is hierarchy. The network type is issue. The data is shown on a logarithmic scale. The projects are Three.js, Deno, and Reveal.js.



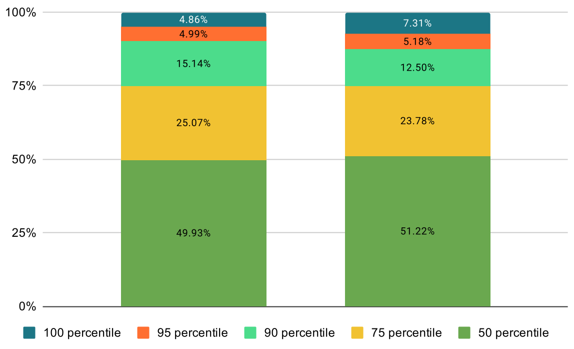
Tensorflow



Atom



Keras



Nextcloud

Figure A.48: Barplots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is issue. The projects are Tensorflow, Atom, Keras, and Nextcloud.

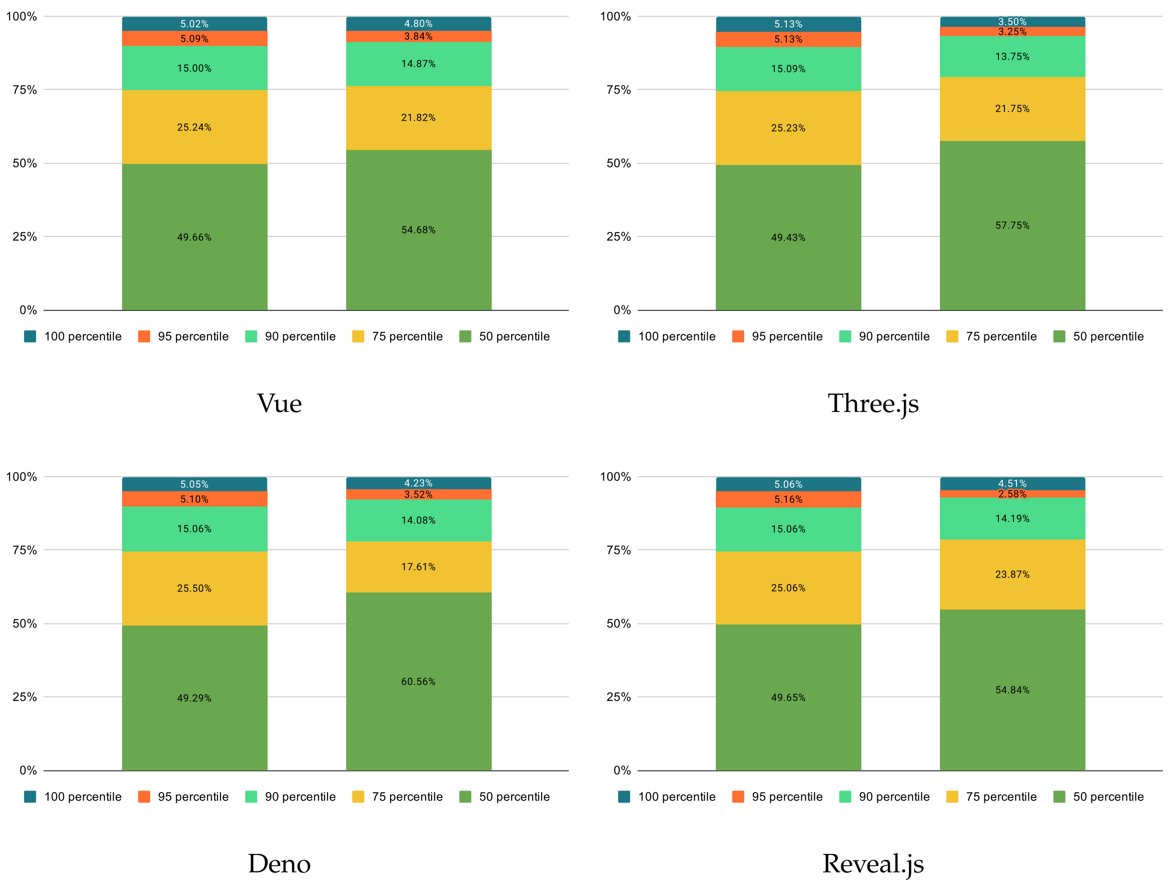


Figure A.53: Barplots for the coreness values of male and female developers. The coreness metric is eigenvector centrality. The network type is issue. The projects are Vue, Three.js, Deno, and Reveal.js.

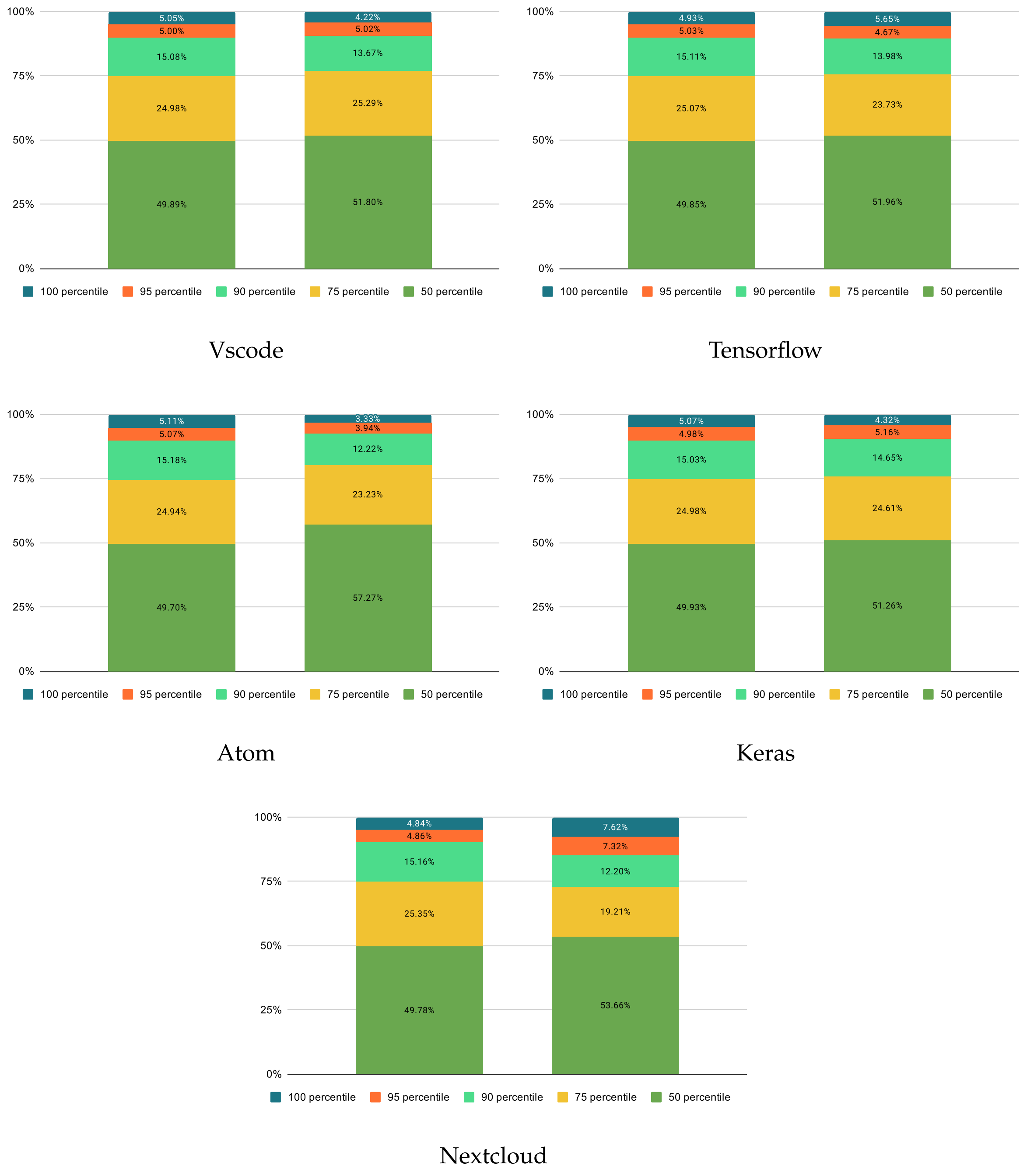
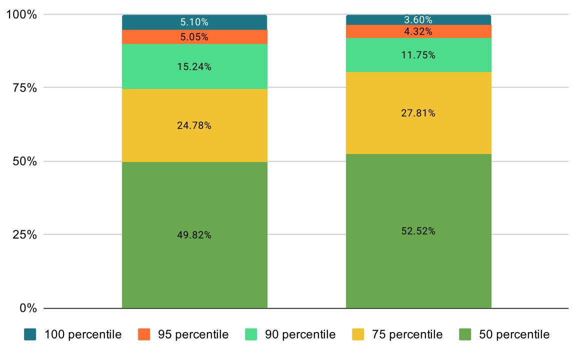
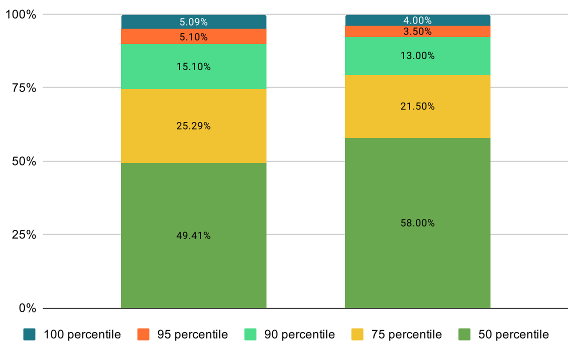


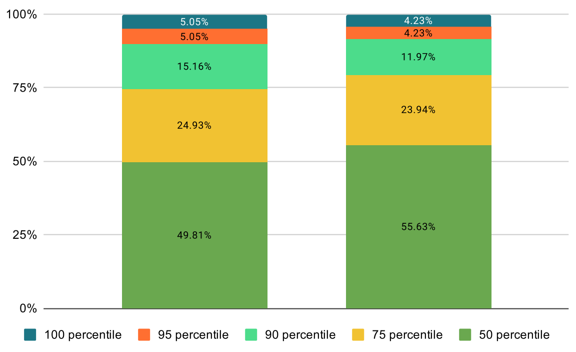
Figure A.59: Barplots for the coreness values of male and female developers. The coreness metric is hierarchy . The network type is issue. The projects are Vscode, Tensorflow, Atom, Keras, and Nextcloud.



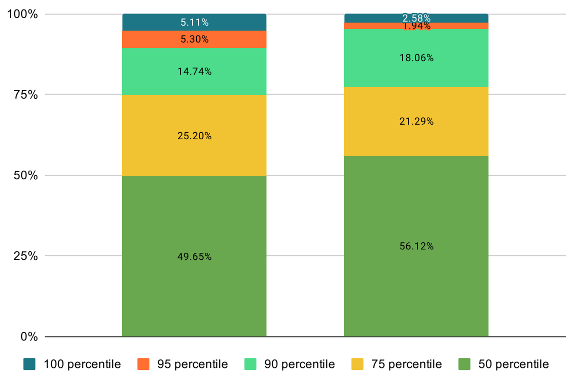
Vue



Three.js



Deno



Reveal.js

Figure A.64: Barplots for the coreness values of male and female developers. The coreness metric is hierarchy . The network type is issue. The projects are VUE, THREE.JS, DENO, and REVEAL.JS.

BIBLIOGRAPHY

[1] Shaosong Ou Alexander Hars. “Working for free? Motivations for participating in open-source projects.” In: *International journal of electronic commerce* 6.3 (2002), pp. 25–39.

[2] Morten Andersen-Gott, Gheorghita Ghinea, and Bendik Bygstad. “Why do commercial companies contribute to open source software?” In: *International journal of information management* 32.2 (2012), pp. 106–117.

[3] Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. “Gender, genre, and writing style in formal written texts.” In: *Text & talk* 23.3 (2003), pp. 321–346.

[4] Catherine Ashcraft, Brad McLain, and Elizabeth Eger. *Women in tech: The facts*. National Center for Women & Technology (NCWIT), 2016.

[5] Christian Bird, David Pattison, Raissa D’Souza, Vladimir Filkov, and Premkumar Devanbu. “Latent social structure in open source projects.” In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. 2008, pp. 24–35.

[6] Andrea Bonaccorsi and Cristina Rossi. “Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business.” In: *Knowledge, Technology & Policy* 18.4 (2006), pp. 40–64.

[7] Amiangshu Bosu and Jeffrey C Carver. “Impact of developer reputation on code review outcomes in oss projects: An empirical investigation.” In: *Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement*. 2014, pp. 1–10.

[8] Ulrik Brandes. *Network analysis: methodological foundations*. Vol. 3418. Springer Science & Business Media, 2005.

[9] Andrea Capiluppi, Alexander Serebrenik, and Leif Singer. “Assessing technical candidates on the social web.” In: *IEEE software* 30.1 (2012), pp. 45–51.

[10] Kevin Crowston, Kangning Wei, Qing Li, and James Howison. “Core and periphery in free/libre and open source software team communications.” In: *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS’06)*. Vol. 6. IEEE. 2006, 118a–118a.

[11] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. "Social coding in GitHub: transparency and collaboration in an open software repository." In: *Proceedings of the ACM 2012 conference on computer supported cooperative work.* 2012, pp. 1277–1286.

[12] Valerii Fedorov, Frank Mannino, and Rongmei Zhang. "Consequences of dichotomization." In: *Pharmaceutical Statistics: The Journal of Applied Statistics in the Pharmaceutical Industry* 8.1 (2009), pp. 50–61.

[13] Sander Hoogendoorn, Hessel Oosterbeek, and Mirjam Van Praag. "The impact of gender diversity on the performance of business teams: Evidence from a field experiment." In: *Management Science* 59.7 (2013), pp. 1514–1528.

[14] Mitchell Joblin. "Structural and Evolutionary Analysis of Developer Networks." PhD thesis. Universität Passau, 2017.

[15] Mitchell Joblin, Sven Apel, Claus Hunsen, and Wolfgang Mauerer. "Classifying developers into core and peripheral: An empirical study on count and network metrics." In: *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE).* IEEE. 2017, pp. 164–174.

[16] Mitchell Joblin, Sven Apel, and Wolfgang Mauerer. "Evolutionary trends of developer coordination: A network approach." In: *Empirical Software Engineering* 22.4 (2017), pp. 2050–2094.

[17] Bruce Kogut and Anca Metiu. "Open-source software development and distributed innovation." In: *Oxford review of economic policy* 17.2 (2001), pp. 248–264.

[18] Karim R Lakhani and Robert G Wolf. "Why hackers do what they do: Understanding motivation and effort in free/open source software projects." In: (2003).

[19] Wendy Liu and Derek Ruths. "What's in a name? using first names as features for gender inference in twitter." In: *2013 AAAI Spring Symposium Series.* Citeseer. 2013.

[20] Xiaoguang Lu, Hong Chen, and Anil K Jain. "Multimodal facial gender and ethnicity identification." In: *International conference on biometrics.* Springer. 2006, pp. 554–561.

[21] Andrew Meneely, Laurie Williams, Will Snipes, and Jason Osborne. "Predicting failures with developer networks and social network analysis." In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering.* 2008, pp. 13–23.

[22] Alan Mislove, Sune Lehmann, Yong-Yeol Ahn, Jukka-Pekka Onnela, and James Rosenquist. "Understanding the demographics of Twitter users." In: *Proceedings of the International AAAI Conference on Web and Social Media.* Vol. 5. 1. 2011.

[23] Baback Moghaddam and Ming-Hsuan Yang. "Learning gender with support faces." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (2002), pp. 707–711.

[24] Kumiyo Nakakoji, Yasuhiro Yamamoto, Yoshiyuki Nishinaka, Kouichi Kishida, and Yunwen Ye. "Evolution patterns of open-source software systems and communities." In: *Proceedings of the international workshop on Principles of software evolution*. 2002, pp. 76–85.

[25] Brunelli Poggio, R Brunelli, and T Poggio. "HyberBF networks for gender classification." In: (1992).

[26] Gregorio Robles and Jesus M Gonzalez-Barahona. "Contributor turnover in libre software projects." In: *IFIP International Conference on Open Source Systems*. Springer. 2006, pp. 273–286.

[27] Gregorio Robles, Laura Arjona Reina, Jesús M González-Barahona, and Santiago Dueñas Domínguez. "Women in free/libre/open source software: The situation in the 2010s." In: *IFIP International Conference on Open Source Systems*. Springer. 2016, pp. 163–173.

[28] Pamela Samuelson. "IBM's pragmatic embrace of open source." In: *Communications of the ACM* 49.10 (2006), pp. 21–25.

[29] Lucía Santamaría and Helena Mihaljević. "Comparison and benchmark of name-to-gender inference services." In: *PeerJ Computer Science* 4 (2018), e156.

[30] Pankaj Setia, Balaji Rajagopalan, Vallabh Sambamurthy, and Roger Calantone. "How peripheral developers contribute to open-source software development." In: *Information Systems Research* 23.1 (2012), pp. 144–163.

[31] Antonio Terceiro, Luiz Romario Rios, and Christina Chavez. "An empirical study on the structural complexity introduced by core and peripheral developers in free software projects." In: *2010 Brazilian Symposium on Software Engineering*. IEEE. 2010, pp. 21–29.

[32] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. "Gender differences and bias in open source: Pull request acceptance of women versus men." In: *PeerJ Computer Science* 3 (2017), e111.

[33] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. "Perceptions of diversity on git hub: A user survey." In: *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE. 2015, pp. 50–56.

[34] Bogdan Vasilescu, Daryl Posnett, Baishakhi Ray, Mark GJ van den Brand, Alexander Serebrenik, Premkumar Devanbu, and Vladimir Filkov. "Gender and tenure diversity in GitHub teams." In: *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. 2015, pp. 3789–3798.

[35] Balazs Vedres and Orsolya Vasarhelyi. "Gendered behavior as a disadvantage in open source software development." In: *EPJ Data Science* 8.1 (2019), p. 25.

[36] Yunwen Ye and Kouichi Kishida. "Toward an understanding of the motivation of open source software developers." In: *25th International Conference on Software Engineering, 2003. Proceedings.* IEEE. 2003, pp. 419–429.

[37] Yue Yu, Huaimin Wang, Gang Yin, and Charles X Ling. "Reviewer recommender of pull-requests in GitHub." In: *2014 IEEE International Conference on Software Maintenance and Evolution*. IEEE. 2014, pp. 609–612.

[38] Xin Zhang, Yang Chen, Yongfeng Gu, Weiqin Zou, Xiaoyuan Xie, Xiangyang Jia, and Jifeng Xuan. "How do multiple pull requests change the same code: A study of competing pull requests in github." In: *2018 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2018, pp. 228–239.

[39] Jiaxin Zhu, Minghui Zhou, and Audris Mockus. "Effectiveness of code contribution: From patch-based to pull-request-based tools." In: *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 2016, pp. 871–882.