DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering and Analytics

Measuring Code Review in the Linux Kernel

Başak Erdamar

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Data Engineering and Analytics

Measuring Code Review in the Linux Kernel

Vermessung von Code Reviews im Linux Kernel

Author:	Başak Erdamar
Supervisor:	Prof. Jens Großklags, Ph.D.
Advisor:	Fabian Franzen, M.Sc.
Submission Date:	15.03.2021

I confirm that master's thesis is my own work and I have documented all sources and material used.

Munich, 15.03.2021 Başak Erdamar

Acknowledgments

I would like to thank Lukas Bulwahn for extensive feedback and discussions. I especially thank Anmol Singh for her guidance and insights. I thank the participants of the ELISA Workshop for their inputs and feedback. I also thank Shuah Khan for her insights on the Linux kernel development process. I thank Ralf Ramsauer for his technical guidance.

Abstract

The Linux kernel development is a prominent example of software development at large scale, i.e., a large code base, a large number of individual contributors, a large organisational structure for integrating changes and a public code review activity open to all peers and stakeholders. The review process data is made available through public e-mail archives, which enables access to information about the process including the last few years of development.

There is no defined measure of error proneness of code changes in the Linux kernel. Nevertheless, public availability of the review process offers many tools to analyze the depth of investigation that each chunk of code goes through ahead of being integrated into the Linux kernel mainline. This thesis tries to answer the question of whether the review process provides enough information to determine the resulting software errors. For this purpose, it considers various variables to establish code review depth and quality. By doing so, it investigates whether the lack of in-depth review indicates potential errors after integration, in an attempt to develop a measure of software development quality. This software development quality measure could be used as input for assessing software security risks.

Throughout the development process usage of various emails and commit message tags are encouraged to be used, as well as complying with the patch submission and review guidelines[1]. This thesis investigates compliance with these guidelines in the review process, by doing so, attempts to construct relations with the resulting security risks in the Linux kernel.

The resources for the investigation includes commit messages in the Linux kernel and review emails. These resources are provided by PaStA tool in a structured manner, which this thesis uses for data mining purposes. These resources are then utilized to construct variables that potentially define the review process.

Contents

1. Introduction

1.1. On this thesis

Each patch in the Linux kernel development process goes through certain steps before being integrated into the Linux kernel. A patch is typically submitted to a relevant mailing list once it is designed and developed by its author. Early review by other developers on the mailing lists happens at this step. Next, a patch is integrated into relevant kernel maintainer's integration tree. Once it is integrated, it moves through more comprehensive maintainer trees, until it is integrated into the mainline. Wider exposure happens on this last part, as the potential problems about the patch is discovered through integration [2].

This thesis looks into various factors which can impact the amount of review in the early review step mentioned above. Number of responses is used as a measure of the extent of review. Figure 1.1 shows the log-scaled distribution of the number of unique responses over patches that were eventually included in the repository, for the kernel releases v5.0 to v5.8. Throughout this thesis, the reasons behind this distribution is investigated.

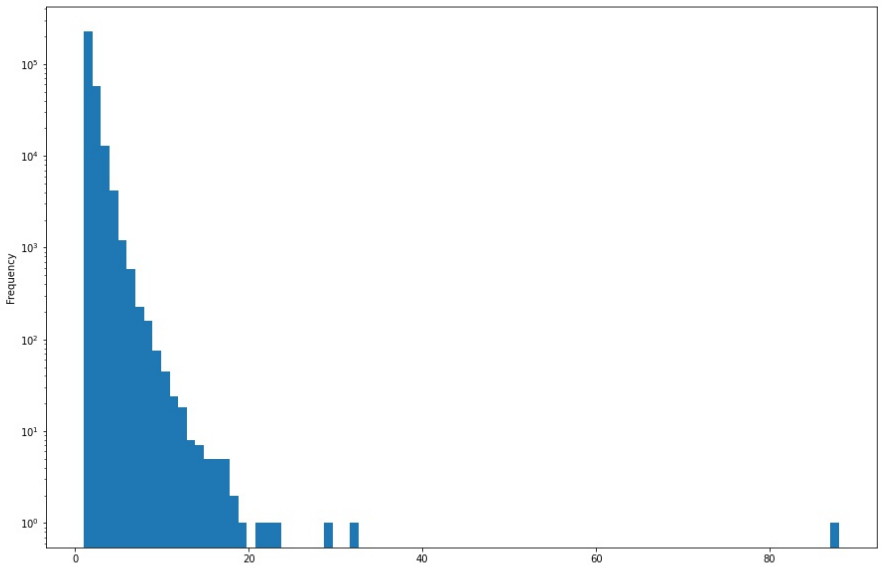

Figure 1.1.: Distribution of number of responses

1.2. Further Reading

Reader can find more extensive information on Linux kernel development process from its documentation website, kernel.org/doc/. Specifically, detailed guidelines about the development and review process are provided under Working with the kernel development community topic.

Annual reports, which contain more statistics about the activities of the Linux Foundation can be found on linuxfoundation.org.

Statistical testing and visualization tools used in this thesis can be found on scipy.org. The library used for machine learning algorithms can be found on scikit-learn.org. The mailing list archives which were used for the analysis were mined and structured with PaStA tool, which can be found on github.com/lfd/PaStA.

2. Research Questions

We hypothesize that, who the patch author is and where he or she is active impacts the number of review emails he or she receives. For this purpose, we state the following questions:

- Does the number of responses increase as the patch developer is more experienced?
- Do maintainers get fewer or more responses than others, when they author a patch?
- Do patch developers who have previously been active in some areas of the kernel get more responses than developers who have been active in other areas?

We also investigate various characteristics of the patches themselves; such as files, sections and mailing lists. For this purpose we state the following questions:

- Does the number of responses increase or decrease with the number of files a patch proposes to change?
- Does the number of responses increase or decrease with the number of maintainer sections to which changed files belong to?
- Does a patch get more responses if it is submitted to more mailing lists?
- Do some mailing lists or maintainer sections lead to larger numbers of responses than others?

3. Authoring Activity

For this chapter, patches between January 2018 and August 2020 and releases between v5.0 to v5.8 are considered. Developer statics active months, commits and last associations are gathered separately using the gitdm tool and the complete git history until the release v5.8.

3.1. Comparing Author Activity in Kernel

In this section, we attempt to associate developer competence with the amount of reviews a developer's patches receive.

The gitdm tool provides statistics on developers and commits based on the git history of the repository given [3]. The following statistics were derived from the tool using the git history of the Linux kernel from the start to the release v5.8. The tool also provides mapping from people to companies given a configuration file [4].

Sasha Levin and the commits he has authored has been excluded from the results in figure 3.1 and figure 3.2 because of his extraordinary activity. Furthermore, bots and self responses are detected and filtered out in the following analysis, in order to more precisely measure the human review activity and the characteristics of the developers in the process. Bot activity is investigated separately in chapter 6.

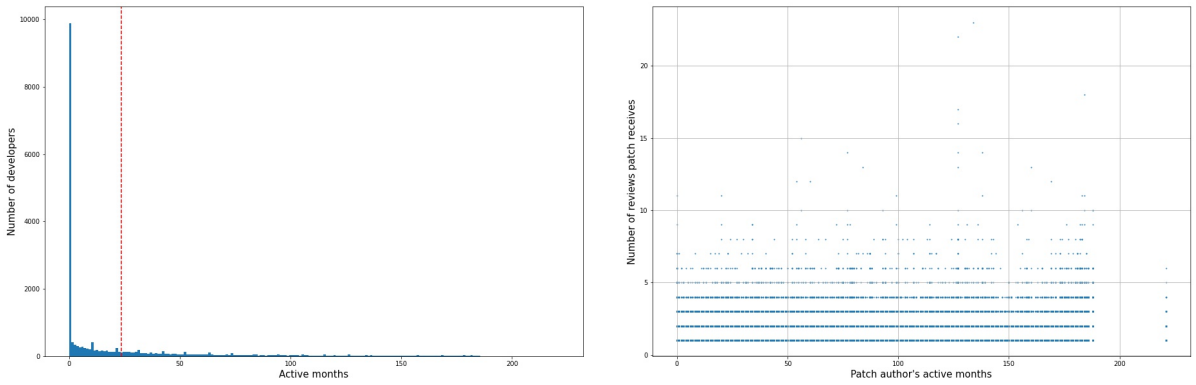

Figure 3.1.: Author activity in terms of months

Active months is a measure that was derived from statistics which are outputs of the gitdm tool. Gitdm generates dates of the first and the last commits authored by each developer. These dates are then refined and used to calculate the number of months each author has been active for.

The histogram in figure 3.1 shows the distribution of active months among developers. The red line marks the average number of active months among developers. A jump can be observed on the left side of the histogram which represents the developers newly joining the kernel community together with shortly active committers to the kernel. In addition, each point the scatter plot in 3.1 shows represents a patch in the kernel. The x and y axes show the number of responses received and the active months of its author respectively. There no clear relation observed between these two quantities while outliers where many responses are received are seen.

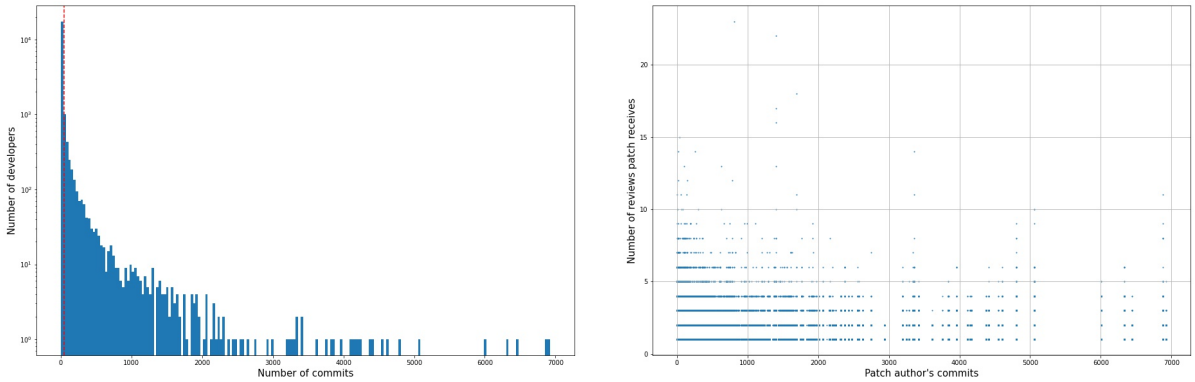

Figure 3.2.: Author activity in terms of commits

Number of commits made by a developer is an alternative measure to quantify developer experience in the Linux kernel. Histogram in figure 3.2 shows the logarithm scaled distribution of the number of commits across developers. Similar to figure 3.1, a skew to the left of the distribution graph is seen. The second graph shows the relation between a the number of responses received and the patch author's active months. As a result of the amount of the developers with fewer commits, the lower left part of the plot is quite dense. No clear relationship between the two measures is observed. However, since many developers appear to have less experience, the distribution plot suggests looking into less experienced developers in the kernel. Further investigation on their activity in the kernel is included in the following section.

Figure 3.3 shows the top companies were most frequently associated to and the corresponding number of newcomer or shortly active developers. The newcomer developers or shortly active developers are defined as the developers with less than 2 active months. This aggregation demonstrates the significance of these companies in the process of engaging new people with the kernel development. 20.23% of the developers with less than 2 active months and 28.14% of all developers were associated with these companies. Figure 3.4 shows the percentages of developers these companies hire. As it is seen in the graph, the leading company in the newcomer and shortly active developers, Intel, also hired a significant portion of the kernel developers. 6.04% of all developers were hired by them. Similarly, the following two companies in the highest number of newcomer and shortly active ranking, Google and IBM, hired 2.12% and 2.23% of all developers respectively.

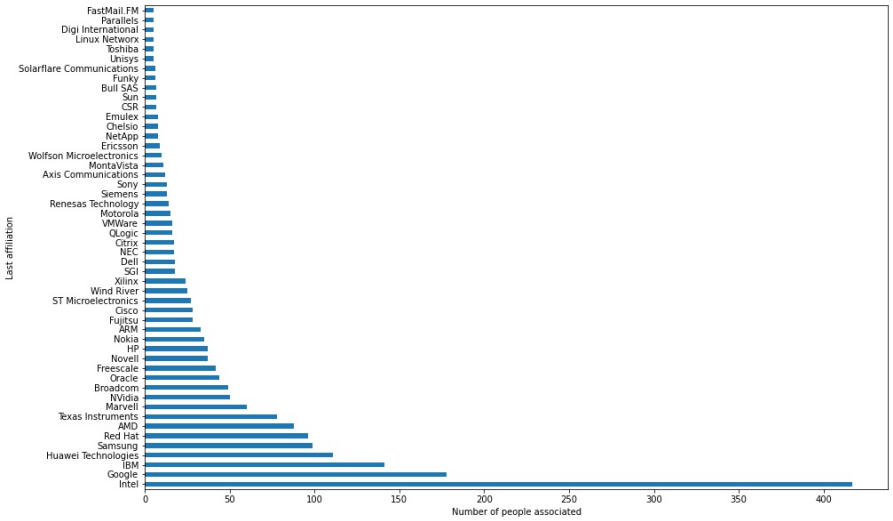

Figure 3.3.: Top companies the shortly active developers are associated to

In addition to the companies hiring the shortly active or newcomer developers, large proportions of the developers associated with them are among the developers with fewer active months. Figure 3.5 displays the top 50 companies who employ the largest numbers of kernel developers. The x axis shows the number of the corresponding developers. The plot is colored based on the developer's level of experience measured in the number of active months. As it is seen in the graph, most of the developers that are associated to these companies have lower numbers of active months in the kernel. Intel has 34.38% of its developers in the kernel having less than 2 active months. However, there are exceptions to this behavior. For example 76% Red Hat developers in the kernel has larger than 1 active months, 59% has larger that 24 months.

3.2. One Time Committers in the Linux Kernel

Since the fact that committing one time to the kernel can be a quite common practice, as established in the previous section, the focus of the following analysis is the content and the authors of the commits made in such a way.

Figure 3.6 shows the 20 most frequently changed files by the one time committers. MAINTAINERS file includes the maintainer, mailing list and kernel section information for developers to use as a reference for development. This file is not used when building the kernel while being included in the repository [5]. Focusing on the rest of the popular files, the most common directory, drivers, is a popular first step in kernel development for developers, as kernel maintainer Shuah Khan points out. She argues that the reasons for the popularity of drivers are easier acceptance compared to core kernel sections in addition to being an easier area to start with the kernel development. In these cases, the developers possibly add support for new drivers, add device IDs to the existing drivers or send fixes to the existing drivers [6].

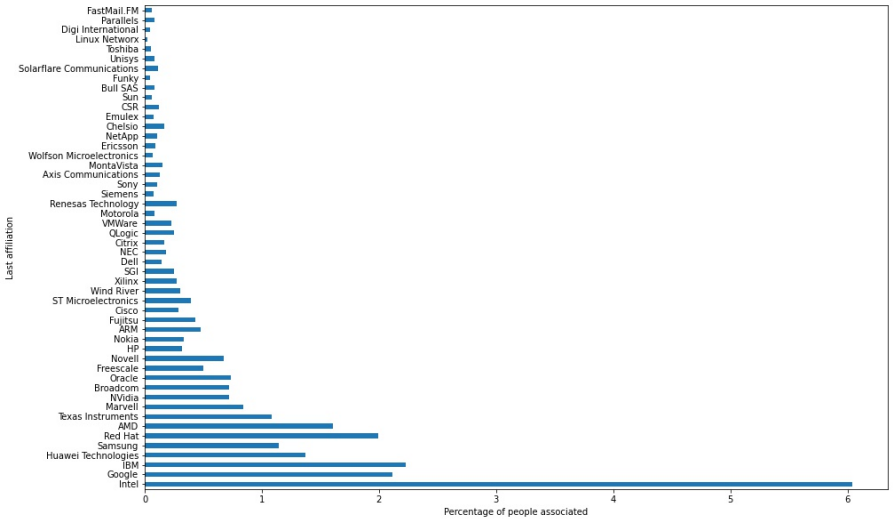

Figure 3.4.: Percentages of developers associated to selected companies

Figure 3.7 shows activity of shortly active committers, activity of one time committers and total activity by time. The activity described as shortly active contains authors who have less than 2 active months. This could either mean that an author newly joined the kernel development community or he/she has worked for a short period time and has not authored a commit afterwards. In figure 3.7, a gradual increase in total number of commits are observed in weekly chart through kernel releases of all time, both one time committers' commits and shortly active authors' commits remain relatively insignificant. We can observe larger numbers of shortly active author commits compared to one time committer commits throughout the time period investigated. However, this comparison is justified by the fact that the shortly active authors include one time committers, people who newly joined the community and people who have been active for a short period of time like one time committers but authored multiple commits. It is important to further note that the jump in both shortly active author commits and one time committer commits at the end of the graphs are caused by the authors whose future activity cannot be foreseen and might change in the future as they continue their activity.

3.3. Author Activity Area

Another aspect of the developer activity and a potential variable affecting the amount of review is the area of activity. To define the area, mailing lists that each author has submitted patches to is used. The motivation for defining this measure is to be able to draw characteristics of each developer based on what they have been doing in the kernel. We further attempt to associate where a developer belongs to based on this measure with the number of review emails they receive.

In order to define each person using mailing lists, vectors representing each person are

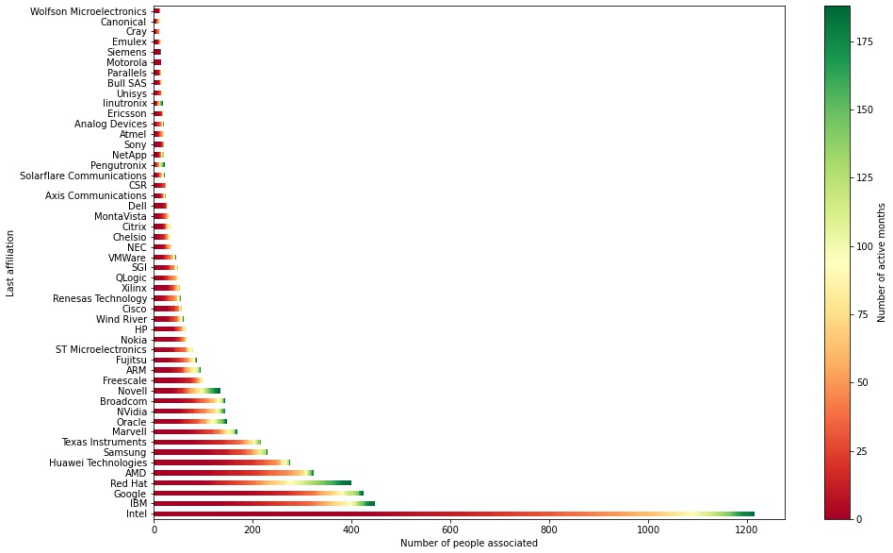

Figure 3.5.: Top companies with the larges number of developers associated to them

Table 3.1.: A sample vector representing a person by mailing lists

'alsa-devel@alsa-project.org'	'amd-gfx @lists.freedesktop.org'	'devicetree @vger.kernel.org'	'dri-devel @lists.freedesktop.org'
0.13	0	0.38	0.11

created. The vector dimensions are each of the mailing lists and the dimension values are the number of patches the developer has submitted to the corresponding mailing list.

Each developer vector is then re-scaled to have the norm of 1 or normalized. After the normalization, dimensions to be used in the following analysis are selected using a variance threshold of 0.01. The purpose of the normalization is to allow each developer to contribute to the variance threshold equally. Additionally, the normalization is intended to prevent imbalances in the following analysis. The variance threshold is used so that some of the less active mailing lists are eliminated and the mailing list that more distinctly define the developers are used. Table 3.1 shows a part of a sample vector representing an individual developer after the operations explained above are applied.

Next, we attempt to cluster developers in groups using the described vectors to represent each person. K-Means algorithm is used for clustering. The algorithm works by repeatedly assigning points to cluster centers and calculating new cluster centers using the points in the clusters [7].

Since K-Means algorithm requires to specify the number of clusters, a criteria to determine the number of clusters was needed. The sum of distances from each point to the corresponding cluster center is selected to be an error measure for this purpose. The number of clusters are then selected to be the point where the decrease of sum of distances plateaus. Figure 3.8 shows the changes in the errors as the number of clusters is increased. 26 is determined to be

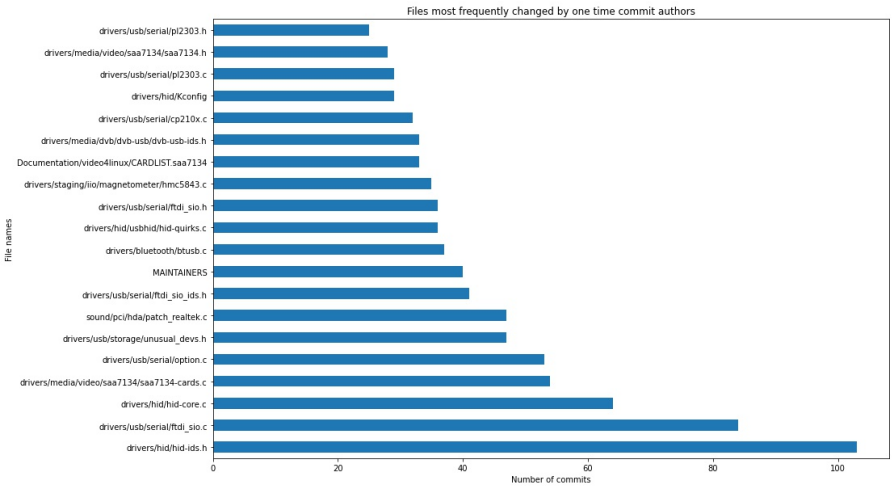

Figure 3.6.: Number of one timer commits to popular files

the appropriate number of clusters based on this measurements.

The dimension values of each cluster center is displayed in figure 3.9. The cluster centers can be interpreted as a representative of an average person that belongs to that cluster. We proceed to determine whether belonging to a certain cluster implies a differing amount of reviews from others.

Figure 3.10 shows the average number of responses received per patch authored by developers in each cluster. The red line marks the average number of responses received per patch regardless of the clusters. As it is seen in the graph, there are clusters both above and below the average. In order to statistically determine these deviations from the average and the deviations between the clusters, we proceed to apply a statistical test.

As the distributions of number of responses per patch are not normally distributed as displayed in appendix A, a non-parametric test had to be performed. Kruskal-Wallis H test is selected for this purpose. Kruskal-Wallis H test performs one-way analysis of variance using ranks. This test does not require normally distributed samples. The samples are only assumed to be independent and identically distributed [8]. Our null hypothesis is that all of the group means are equal while our alternative is that at least two group means differ from each other.

$$H_0 : \mu_i = \mu_j \; \forall i, j \in [0, 25] \quad H_A : \mu_i \neq \mu_j \; \exists i, j \in [0, 25] \tag{3.1}$$

Note that each group contains a series of numbers that are the number of responses received to a patch authored by a person in the corresponding cluster. Therefore, patch authors are clustered and the numbers of responses their patches received are statistically tested. The test resulted in a p-value of 0 in the measured precision (or rather a p-value < $2.2 * 10^{-16}$) and the null hypothesis is rejected with a 95% significance to conclude that at least two of the groups differ from each other.

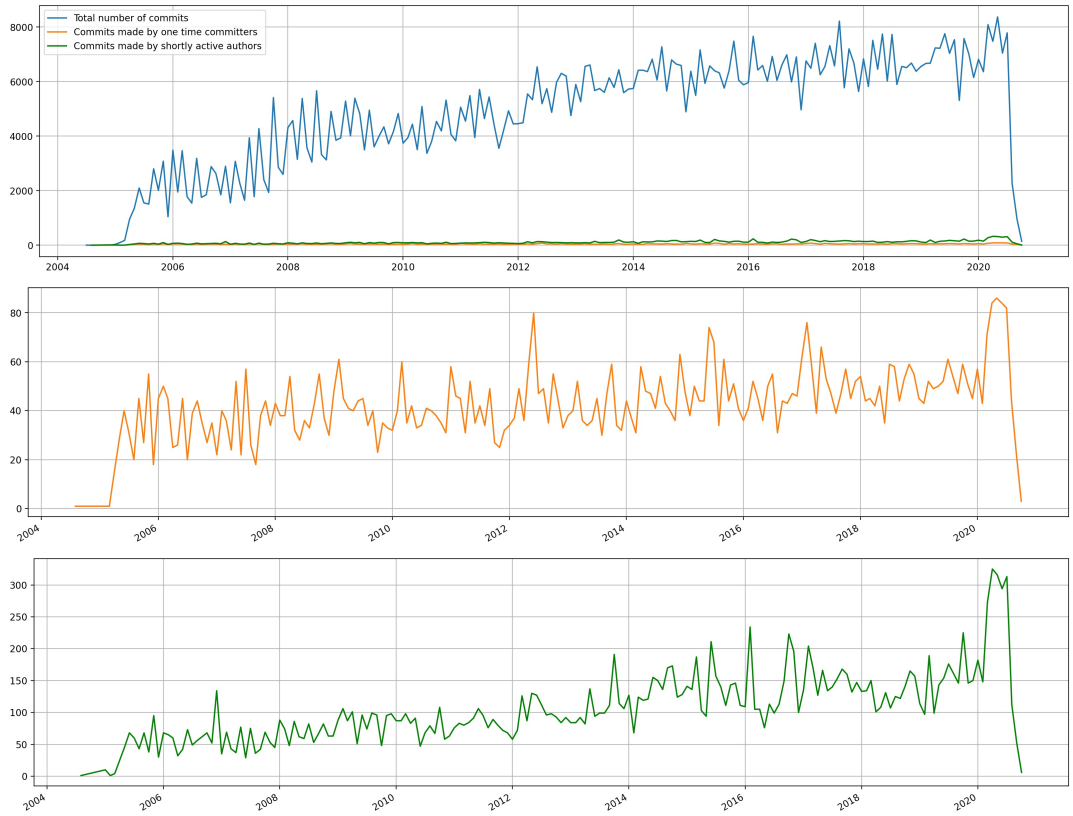

Figure 3.7.: One time committer and total activity through time

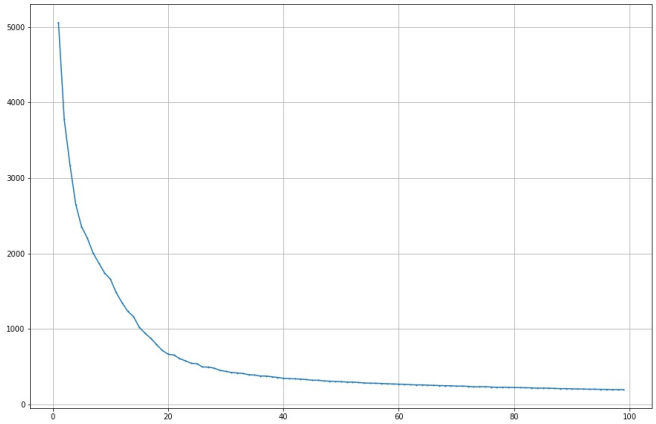

Figure 3.8.: Sums of errors for various numbers of clusters

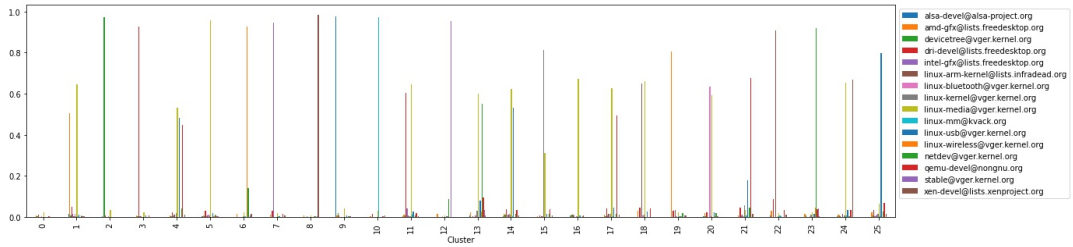

Figure 3.9.: Dimensions of each cluster center

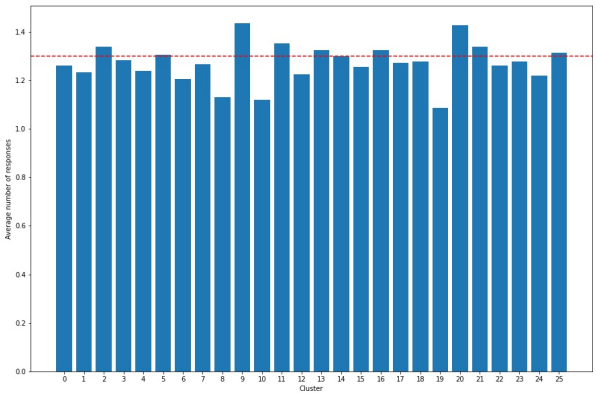

Figure 3.10.: Average number of responses per patch in each cluster

4. Maintainers

We continue the analysis of patch authors by differentiating between maintainers and other authors. In this chapter, a person is considered to be a maintainer only when he or she is contributing within their maintained area. More precisely, a patch contributing to a section out of a maintainer's area of responsibility will not be considered a patch authored by a maintainer in the following statistics. Patches between January 2018 and August 2020 and releases between v5.0 to v5.8 are considered. Self responses as well as responses sent by bots were filtered out from these statistics. Additionally, note that the patches which were eventually included in the kernel are considered here.

4.1. Authoring activity by maintainers

We start the analysis by hypothesizing that the patches authored by maintainers receive fewer responses since they are assumed to not need a review in the community. The reasoning behind this assumption is that the patches which are authored by the maintainers possibly include quick fixes in areas where the maintainer already has expertise.

Maintainers authored 17.27% of the patches with upstream commits associated and 18.62% of the upstream commits between kernel releases v5.0 to v5.8. Therefore, a significantly large portion of the patches and an even larger portion of the resulting upstream commits are authored by the maintainers.

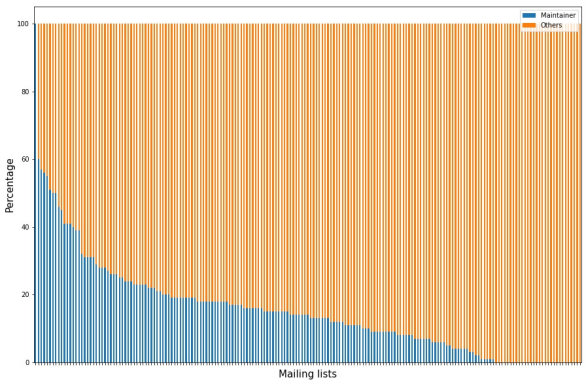

Figure 4.1.: Percentages of patches sent by maintainers

In the bar plot in figure 4.1, each bar represents a 100% of the patches submitted to a mailing list in the kernel development process. The bars are colored depending on the

authors of the patches being maintainers or not. As seen in figure 4.1, a range of 0% to 100% maintainer patches is seen across mailing lists.

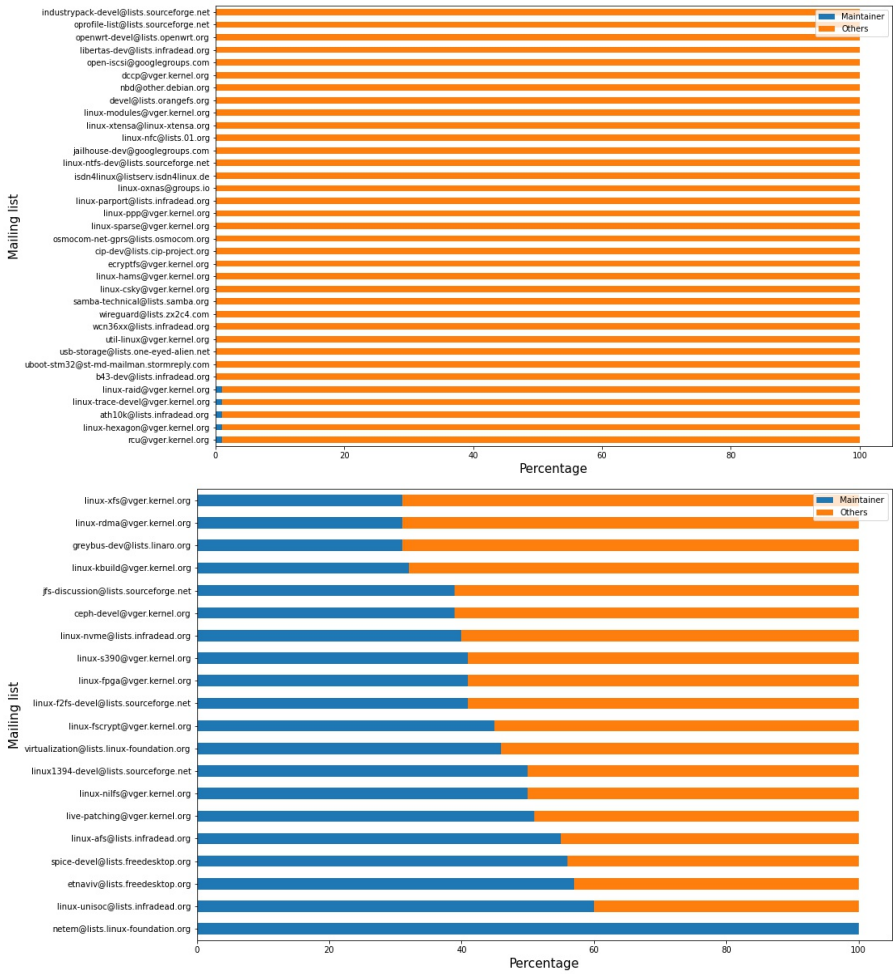

Figure 4.2.: Lists with lowest and highest percentage of maintainer activity

Figure 4.2 zooms in on figure 4.1 and shows the mailing lists which has the 20 highest and the 20 lowest percentages of the patches authored by maintainers. As it is seen in the graph, 100% of the patches on the mailing list netem@lists.linux-foundation.org were authored by maintainers. However, only two patches were authored in the investigated period of time. Similarly, the mailing list which is the second on the highest percentage ranking, linux-unisoc@lists.infradead.org, had only five patches submitted in this period. Furthermore, all of the twenty mailing lists on the highest percentage constitute a total of 7.05% of all of the patches that were submitted to any mailing list in the investigated time window. We move on to the reviewing activity in order to investigate the potential effects of the authoring activity.

Figure 4.3 shows the absolute numbers of patches authored by maintainers and non-maintainers for the top 20 mailing lists which have received the largest number of patches. In the most active list in this ranking, linux-kernel@vger.kernel.org, 13.81% of all patches

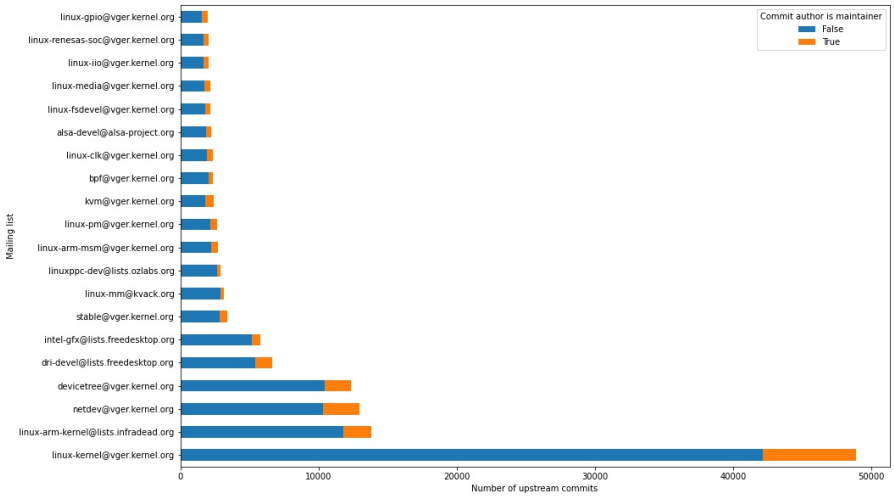

Figure 4.3.: Most active mailing lists

were authored by maintainers. 14.42% percent of patches in the second largest list linux-arm-kernel@lists.infradead.org were authored by maintainers. The largest percentage of patches authored my maintainer in the most active 20 lists belongs to kvm@vger.kernel.org. 25.98% of the patches on this mailing list were authored by maintainers. Among all of these top 20 most active lists, average percentage of maintainer patches is 16.69%.

4.2. Review activity

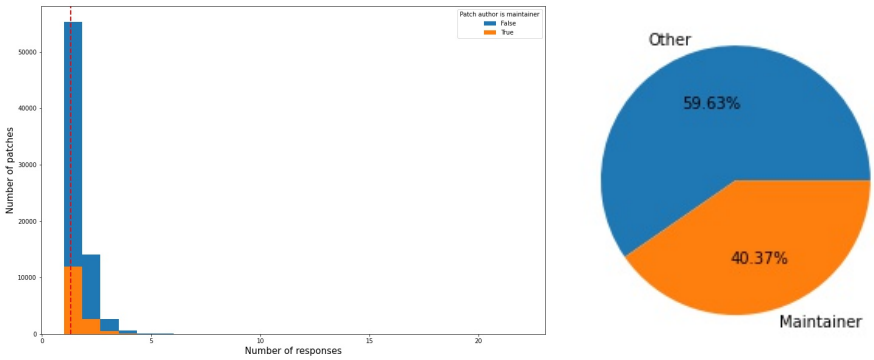

Figure 4.4.: Responses sent to maintainer patches

Even though maintainers authored 17.27% of patches with associated upstream commits, those patches received 40.37% of the total responses. The portion of the responses received by maintainers is significantly larger than the portion of the patches authored by maintainers, which suggests that maintainers are more likely to get a larger number of responses to their patches. In the histogram in figure 4.4 the distribution of the number of responses over

patches is shown. The plot is colored according to the patch author being maintainer or not. It is observed in the histogram that most of the maintainer patches received fewer responses, similar to other patches.

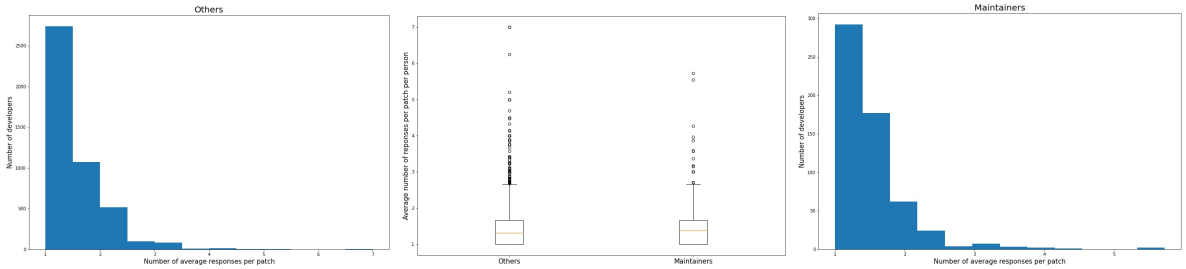

Figure 4.5.: Average number of responses received per patch

Figure 4.5 shows distributions of the average number of responses received per patch by maintainers and by others. In the box plots, similar distributions for maintainers and others are seen. However, quite a few more outliers in the non-maintainer cases are observed. Nevertheless, this can be associated with the volume of the two groups. Maintainers constitute only 11.22% of the population which is being investigated.

As it is clearly seen in both histograms in figure 4.5, the distributions for neither maintainers nor others are normal. Both histograms are skewed heavily to the left. We want to check the hypothesis that maintainers receive a larger average number of responses per patch. Since the two groups have long-tailed and skewed distributions, which cannot be assumed to be normal distributions, a non-parametric rank sum test needed to be conducted. Mann-Whitney test was selected for this purpose. This test compares two populations using ranks. It assumes that the distributions of the two populations are the same but does not require normality. Furthermore this test is robust to outliers [8].

$$H_0 : \mu_{maintainers} = \mu_{others} \quad H_A : \mu_{maintainers} > \mu_{others} \tag{4.1}$$

The test with null hypothesis that the average value of two groups are equal and the one-sided alternative hypothesis that maintainers have a larger mean response. The test resulted with a U statistic 1388087.5 with 574 person in the maintainers group and 4543 persons in the others group. The test rejected the null hypothesis with a 95% confidence. To summarize, there is indeed evidence that maintainers tend to get more responses per patch.

5. Patches

Each patch has certain components such as a patch author, changed files, number of lines and tags to help define it. This chapter focuses on the patches themselves, in order to determine the effects of various factors on the number of responses. For this chapter, patches between January 2018 and August 2020 and releases from v5.0 to v5.8 are considered.

5.1. Patch components

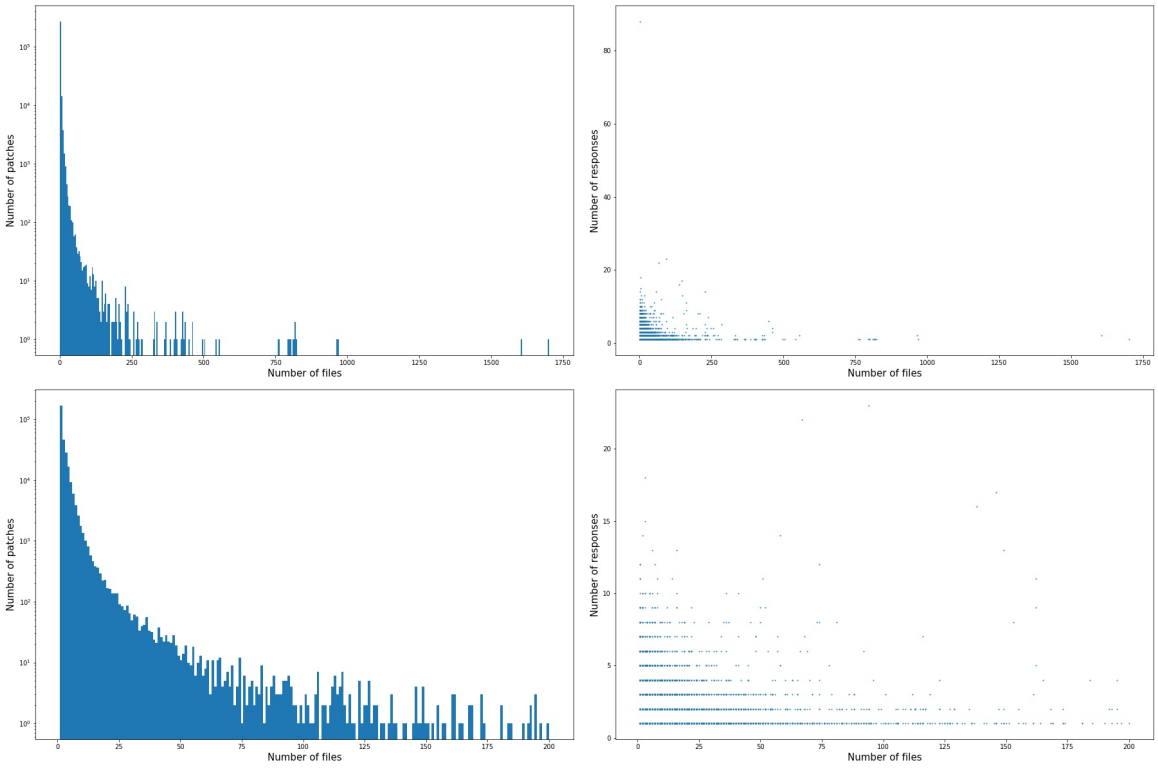

Figure 5.1.: Number of files in a patch

We start with investigating each patch is the number of files it proposes to change. Histograms in figure 5.1 shows the log-scaled distribution of the number of files over patches. A tendency to change fewer files in a single patch can be observed with a few exceptions. In order to determine the effect of solely number of changed files on the amount of review each receives, the relation between files and responses is considered further.

As seen in scatter plots in figure 5.1, no clear positive relationship between number of responses and the number of files can be seen. However when the MAINTAINERS file is considered, it can be seen that the number of files does not necessarily correspond to the number of people who will actively review the patch. Furthermore, there could possibly be intersecting sections in terms of file, meaning maintainers from various sections review the patch. For these reasons, the number of sections is considered next.

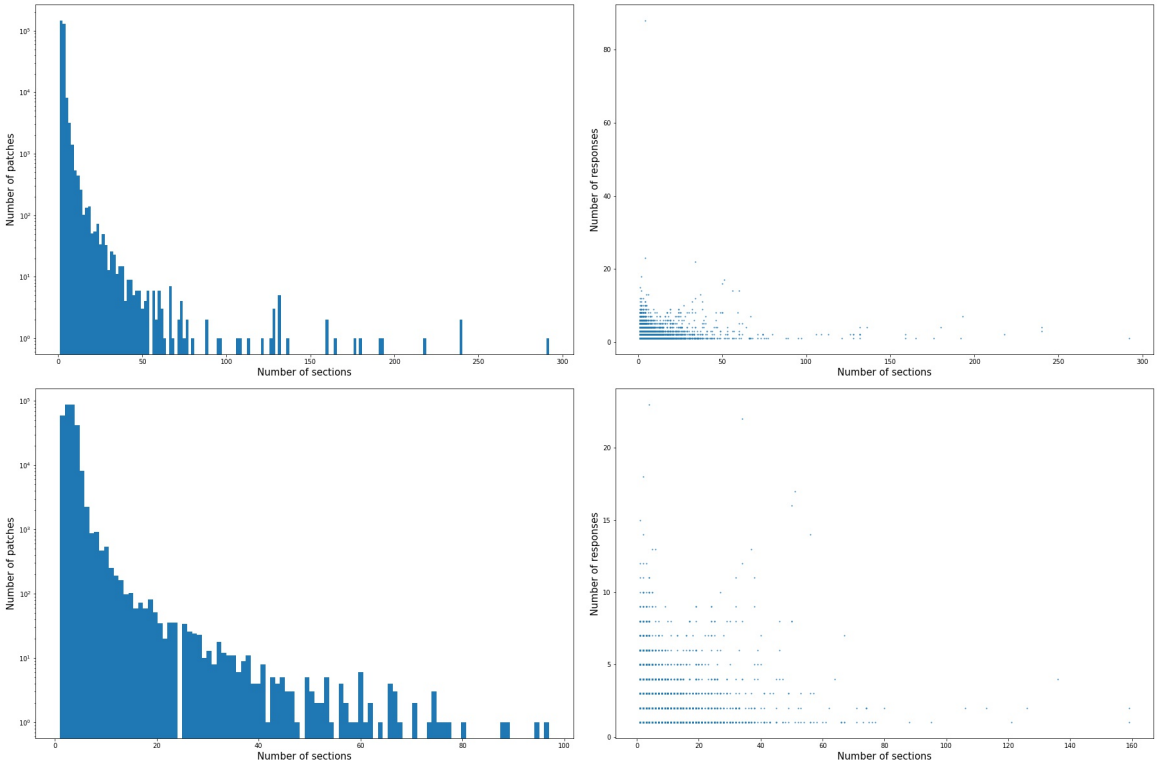

Figure 5.2.: Number of sections related to a patch

A patch is checked for maintainer section information and a combination of each such section is considered to be associated with that patch. Figure 5.2 shows a logarithm scaled distribution of maintainer sections associated with a patch. Similar to the number of files included, a tendency to have a smaller number of associated sections can be observed.

While no linear relationship between the number of responses each patch receive and number of sections can be seen in figure 5.2, outliers containing many sections or many responses are observed.

Histograms in figure 5.3 shows the distribution of the number of mailing lists a patch was sent to. The histogram is heavily skewed to the left. In fact, 51.34% of all the patches that were submitted to the kernel were sent to only one mailing list. This behavior resulted in the crowded area on the second graph, which shows the relation between the number of mailing list a patch was sent to and the number of responses it receives.

Since the number of files and the number of mailing lists seem to have a reverse relation

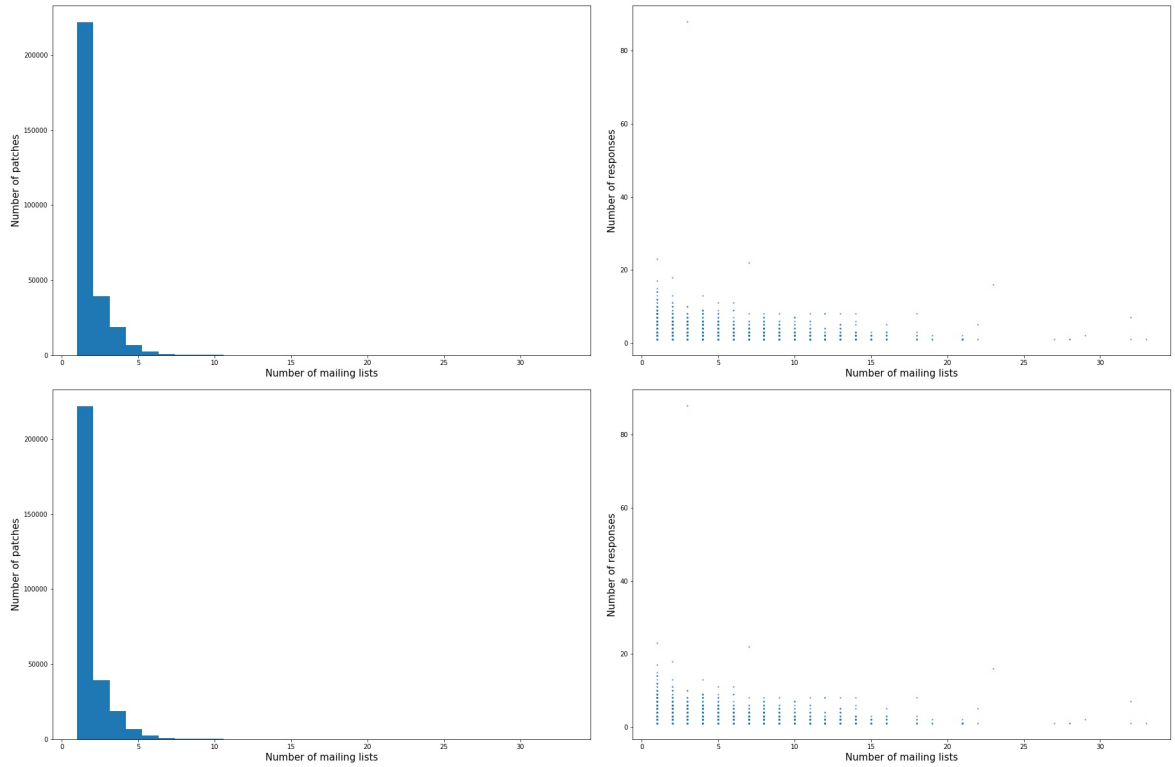

Figure 5.3.: Number of mailing list a patch was sent to

with the the number of responses, we test this relation by setting up a regression formula. For a possible correlation between the number of files and the number of mailing lists, we include an interaction term.

$$1.59 * \frac{1}{x_1} + 1.46 * \frac{1}{x_2} - 1.82 * (\frac{1}{x_1} * \frac{1}{x_2}) = y \tag{5.1}$$

In the above equation, x_1 represents the number of mailing lists and x_2 represents the number of files. y is the resulting dependent variable, number of responses. The regression analysis resulted an R-Squared value of 0.76, which means the selected independent variables only explain the 76% of the variability in the dependent variable y.

$$-2.01 * \frac{1}{x_1^2} - 0.8 * \frac{1}{x_2^2} + 3.22 * \frac{1}{x_1} + 1.41 * \frac{1}{x_2} - 0.61 * (\frac{1}{x_1} * \frac{1}{x_2}) = y \tag{5.2}$$

We further attempt to extend this model by including reverse quadratic terms $\frac{1}{x_1^2}$ and $\frac{1}{x_1^2}$. However, R-Square value is no larger than 0.78 with increased standard errors for each variable. Therefore, we do not have enough evidence to represent y in such a formula.

5.2. Individual Mailing Lists

We would like to investigate whether where a patch is sent to has an effect on the amount of review it receives. For this purpose, we look at the individual mailing lists rather than looking at the number of mailing lists. It is important to consider that self responses and bot responses are filtered out when calculating the following averages while the patch email itself is counted.

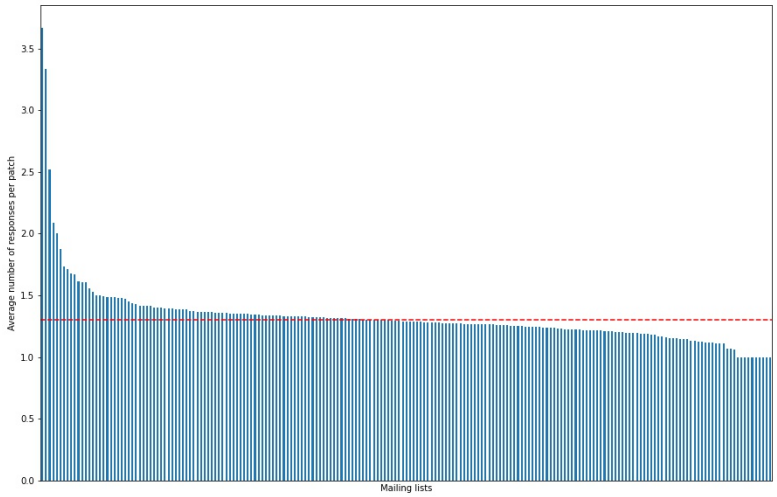

Figure 5.4.: Average number of responses per patch in each mailing list

The bar plot in figure 5.4 shows average number of responses per patch in each mailing list. The graph is not annotated with mailing list names for the sake of readability and simplicity. Nevertheless, it can be seen that there are a wide range of averages from 3.67 on the list workflows@vger.kernel.org to 1 on multiple mailing lists. The average number of responses for any patch, as marked by the red line, is 1.3. Note that a patch may be sent to multiple mailing lists. Thus, one patch may be affecting more than one of the said average. However, from figure 5.3 we already know that many of the patches were sent to a single mailing list. This case is investigated in more detail in the following analysis.

Fig 5.5 lists the top 30 mailing lists to which were isolated in the sense that many patches were only sent to these mailing lists alone. 96.11% of all the emails that were sent to the leading list qemu-devel@nongnu.org were only sent to qemu-devel@nongnu.org. However this list does not suffice to explain the skew to left on figure 5.3 while only 11.26% of all the patches submitted were submitted to this particular list. Second mailing list on the most isolated lists linux-kernel@vger.kernel.org appears to be the most popular list on the kernel. 45.48% of all the patches submitted in the investigated window, were submitted to linux-kernel@vger.kernel.org. Furthermore 15.16% of the patches that were submitted to linux-kernel@vger.kernel.org, were submitted solely here.

In many cases, patches were sent to more than one mailing list. In order to consider such cases, the correlations between the mailing list is investigated. Figure 5.6 shows the

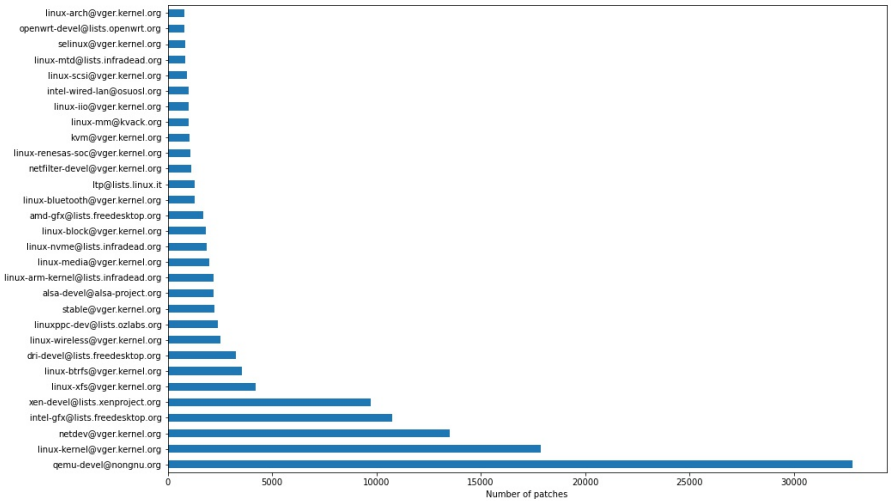

Figure 5.5.: Top most isolated mailing lists

positive correlations between the mailing lists. The first figure shows the entire set of mailing lists while the second figure only shows a set of the selected mailing lists. It is observed in the heat-maps that patches tend to be submitted to certain mailing lists together. For example, linux-arm-kernel@lists.infradead.org and devicetree@vger.kernel.org lists have 0.37 correlation.

For the purpose of considering the individual mailing lists without overlaps and correlations between variables, we move on to cluster patches. The aim of this approach is to construct a better defining model with mailing lists as variables and to take the mailing lists that frequently seen together into account. Therefore, we will be able to associate groups of individual mailing lists to a resulting number of responses per patch. The cluster assignments will bring the patches which were sent to the same mailing lists together.

For clustering, patches are represented as follows. Each patch is defined as a vector of mailing lists. That is, each mailing list is represented as a dimension. Each dimension takes the value of 1 if the patch is submitted to that mailing list and 0 of the patch is not. This vector construction allows to define each patch solely as a combination of mailing lists. Similar to the analysis about individual developers in chapter 3, the vectors which represent each patch are normalized in order to contribute to the variance threshold equally and to enable better clustering.

As the dimensions of such vectors would be high, the following computations would be lengthy. In addition, certain more extensive mailing lists are related to many patches while some other mailing lists are very specific to a set of patches. In order to eliminate such cases, a variance threshold of 0.01 across all patches is applied to each dimension which is representative of a mailing list in this vectorization.

K-Means Algorithm was used for clustering the patches. Since the algorithm requires to specify the number of clusters, the change in total distances from cluster member points to cluster centers are used as a measure of error, similar to described in chapter 3. Figure 5.7

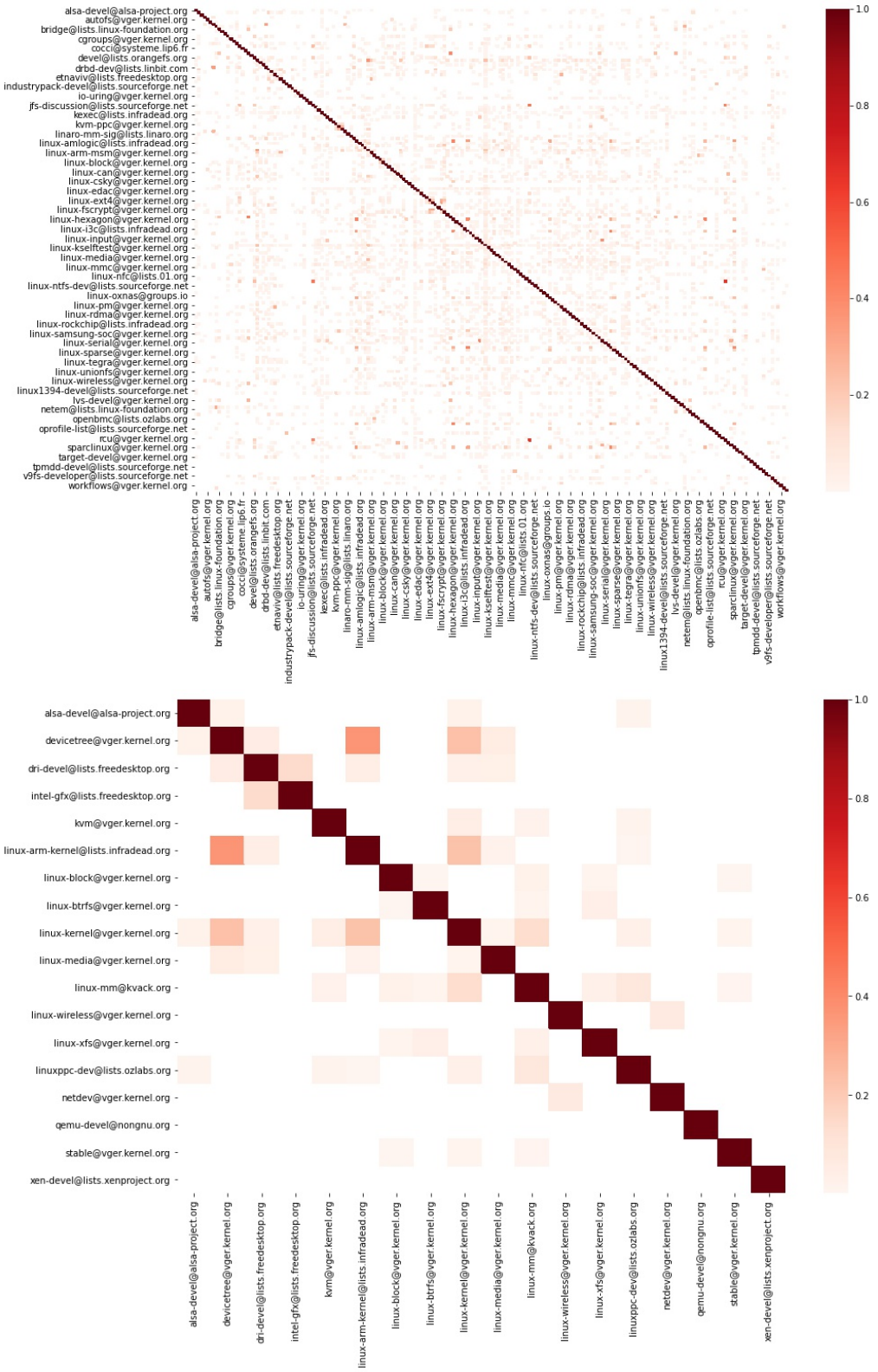

Figure 5.6.: Positive correlations between mailing lists

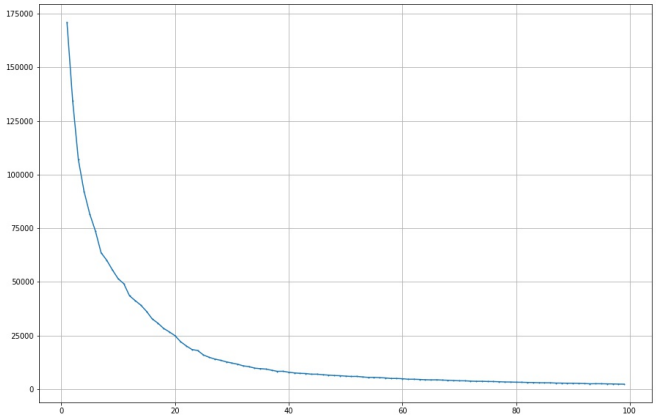

Figure 5.7.: Sums of errors for various numbers of clusters

shows the changes in total error as the number of clusters increase. Taking these errors as a reference, the number of clusters is determined to be 32.

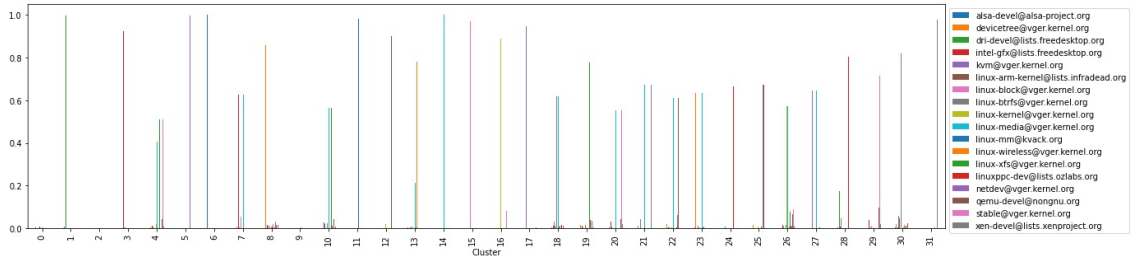

Figure 5.8.: Cluster centroids

Each cluster is represented by a centroid vector which has the same dimension names as the patch vectors. Values of these dimensions is the average of each patch in the cluster. Therefore the centroids may be considered to illustrate the mailing lists that represent the patches in each cluster most frequently belong to. Figure 5.8 displays values of each dimension of the cluster center vectors.

Figure 5.9 shows the average number of responses per patch for each of the clusters. Overall average number of responses a patch receives is marked with the red line. Clusters with both above and below average responses are observed. However the averages across clusters is not sufficient to significantly determine the effects of being in a cluster in terms of human responses. Therefore a statistical test is conducted. Since the distribution of the number of responses is not normal within clusters while having a similar distribution across clusters, the non-parametric Kruskal-Wallis test is conducted. For the distributions across clusters, refer to appendix B.

$$H_0 : \mu_i = \mu_j \; \forall i, j \in [0, 25] \quad H_A : \mu_i \neq \mu_j \; \exists i, j \in [0, 25] \tag{5.3}$$

The null hypothesis and the alternative hypothesis are stated in the above formula. With

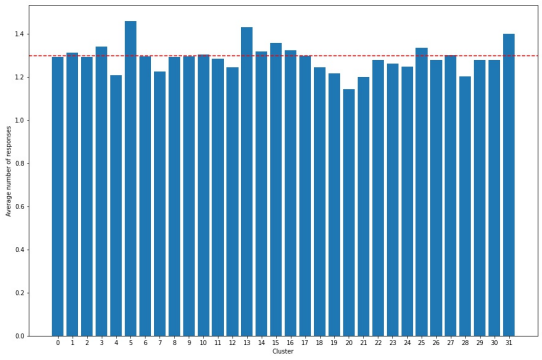

Figure 5.9.: Average number of responses

the test, we are trying to disprove or fail to disprove the assumption of equal means across clusters. The test rejected the null hypothesis with 95% significance. Therefore, there is evidence to reject that patches receive the same amount of average responses regardless of which cluster they were included in based on the mailing lists they were submitted to.

5.3. Individual Sections

Alternatively, when maintainer sections are used as a measure of area in the kernel instead of mailing lists similar results are observed. A union of the maintainer sections corresponding to each file a patch changes is considered to be the sections a patch belongs to. More precisely, if a patch proposes to change multiple files, all of the sections corresponding to all of files is used.

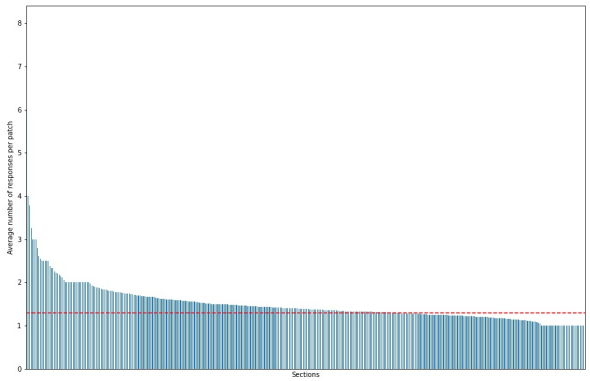

Figure 5.10.: Average number of responses per patch in each section

Figure 5.10 shows the average number of responses within each section. The section names are not annotated for readability. A range of averages is observed. To illustrate, the patches which belong to the section ARM/LG1K ARCHITECTURE has an average of 8 responses. However, only 3 patches were associated to this section. On the other hand, a number of

sections such as ANALOG DEVICES INC AD7292 DRIVER, ATUSB IEEE 802.15.4 RADIO DRIVER and TEXAS INSTRUMENTS' DAC7612 DAC DRIVER has an average of 1 response, which would correspond to the patch email itself. Nevertheless, only 0.24% of the patches were associated with the 168 sections with an average 1 response. Note that a patch may belong to multiple sections and therefore contribute to multiple of means in figure 5.10.

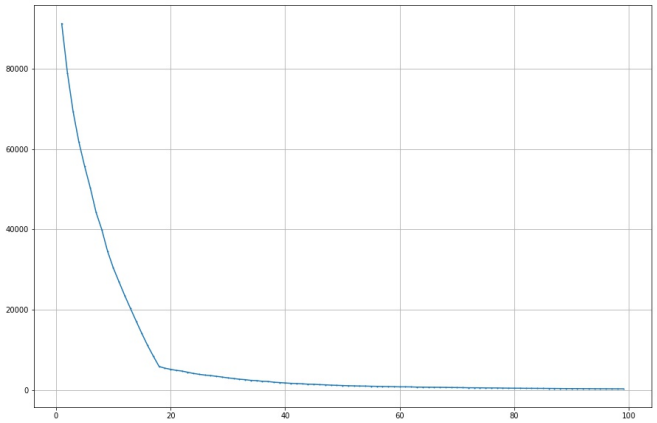

Figure 5.11.: The distribution of average number of responses per patch in each cluster

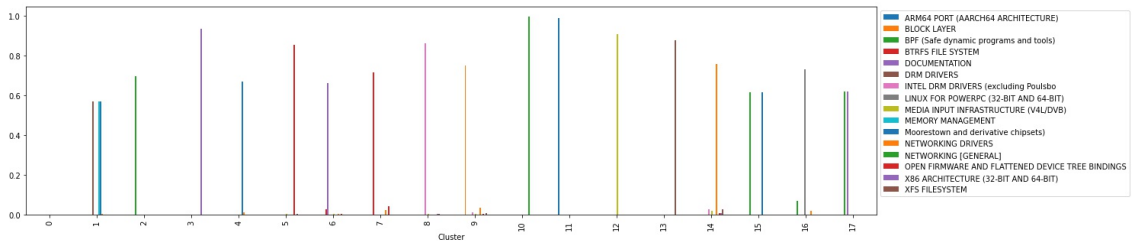

Figure 5.12.: Cluster centroids

Figure 5.7 shows the decrease of errors as the number of clusters is increased. Number of clusters are determined to be 18 in this case. Refer to appendix C for more details on this clustering. Similar to mailing list clustering, the numbers of responses within clusters is not normally distributed. Therefore, the non-parametric Kruskal-Wallis test is used once again.

$$H_0 : \mu_i = \mu_j \; \forall i, j \in [0, 25] \quad H_A : \mu_i \neq \mu_j \; \exists i, j \in [0, 25] \tag{5.4}$$

The null hypothesis is that each of the clusters has the same average number of responses while the alternative hypothesis is that at least two cluster means differ. Similar to the case with mailing lists, the null hypothesis is rejected at the 95% significance. Therefore, there is enough evidence to reject that each cluster has the same average number of responses.

6. Bots

7.94% of the response traffic is classified as being authored by bots through PaStA tool. As the authors of response emails across mailing lists were studied earlier, this activity was excluded in order to focus on the factors determining human activity. This chapter investigates bot activity in detail. Patches between January 2018 and August 2020 and releases from v5.0 to v5.8 are considered.

6.1. Comparing Bot Activity with Human Activity

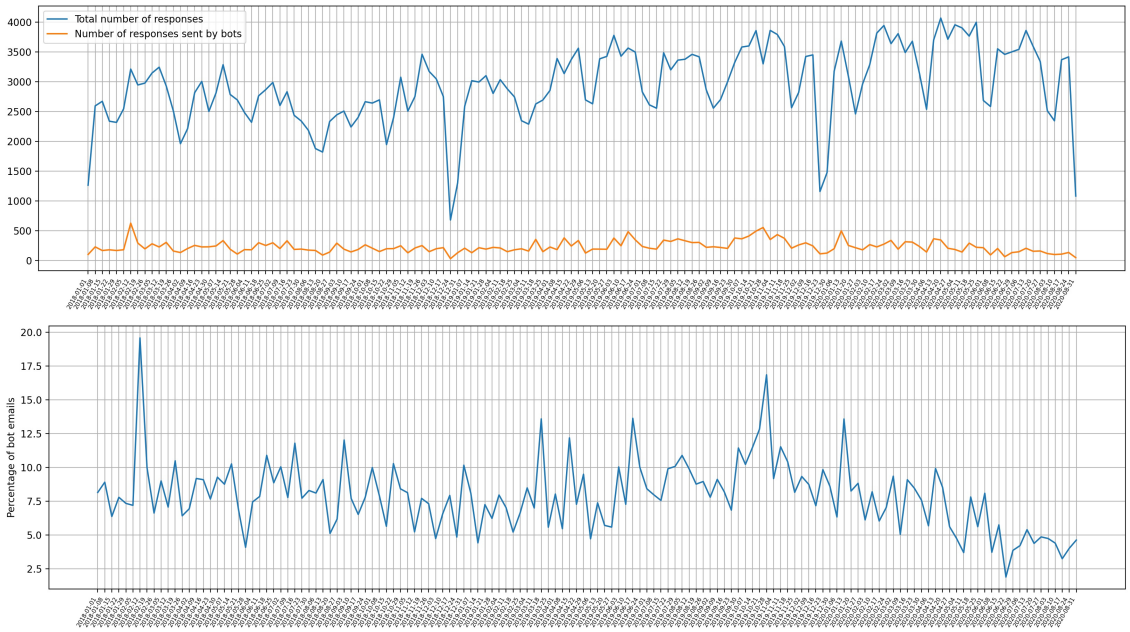

Figure 6.1.: Bot and human response email activity through time

In figure 6.1, weekly total number of emails sent by bots and the weekly total number of emails are shown. The second graph in figure 6.1 shows the weekly percentages of emails sent by bots.

A drop in the amount of activity on holiday seasons in December is observed for both overall and bot cases. Furthermore, increases in the overall activity appears to reflect on increases in the bot activity in many cases. Distinctive examples of such parallel jumps are the weeks of 19-02-2019 and 13-02-2020. However on the percentage graph, no conclusions

can be drawn about the activity by time other than the slight decrease on the relative bot activity starting from the beginning of 2020. In addition, in the previous chapter about the one time committer activity, an increase on the newcomer activity was also observed.

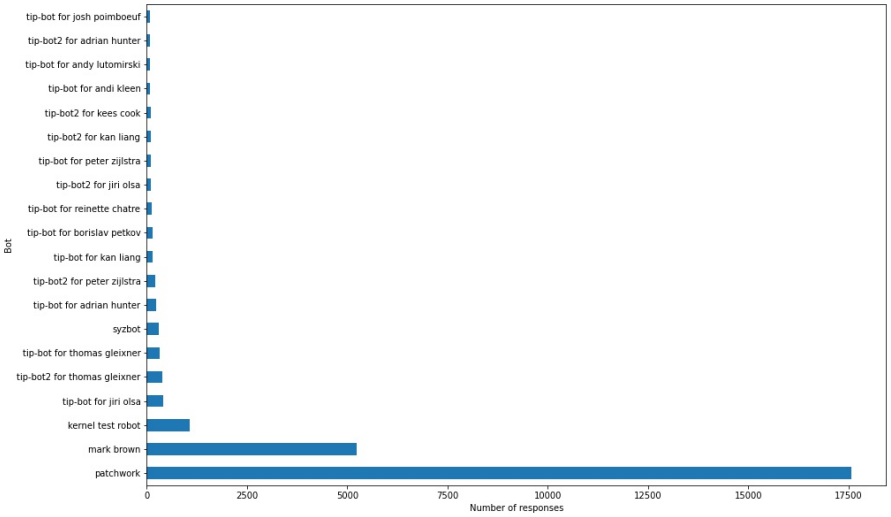

Figure 6.2.: Bots with 20 largest number of emails authored

As shown on figure 6.2, fewer bots such as Patchwork authored significantly larger number of patch responses than the other bots. However, this does not necessarily mean increased bot activity across all areas. For example the patchwork bot has sent all of its large number of emails to the mailing list intel-gfx@lists.freedesktop.org. Similarly, the bot for Mark Brown is significantly active on alsa-devel@alsa-project.org and linux-kernel@vger.kernel.org lists. Nevertheless, it is important to consider the overall email activity on these mailing lists take the bot email activity into account accordingly.

Figure 6.3 shows the logarithm-scaled distribution of the number of responses sent by a single bot. While there are exceptions such as the Patchwork bot and the bot for Mark Brown seen in figure 6.2, individual bots have sent very few emails. Therefore, the amount of bots existing in a mailing lists may not correspond to increased bot activity in the mailing list.

6.2. Bot Activity Across Mailing Lists

In this section, the relative and absolute bot activities are further investigated across individual mailing lists.

The first graph in figure 6.4 displays the percentages of emails sent by bots across different mailing lists. In the second graph, top 20 of the mailing lists which has the largest percentage of bot activity are seen. As seen in figure 6.4, only the top three of the mailing lists has larger than 30% of their email activity coming from bots. 11.94% of all of the emails sent to kernel development community were sent to these top three mailing lists linux-tip-commits@vger.kernel.org, intel-gfx@lists.freedesktop.org and alsa-devel@alsa-

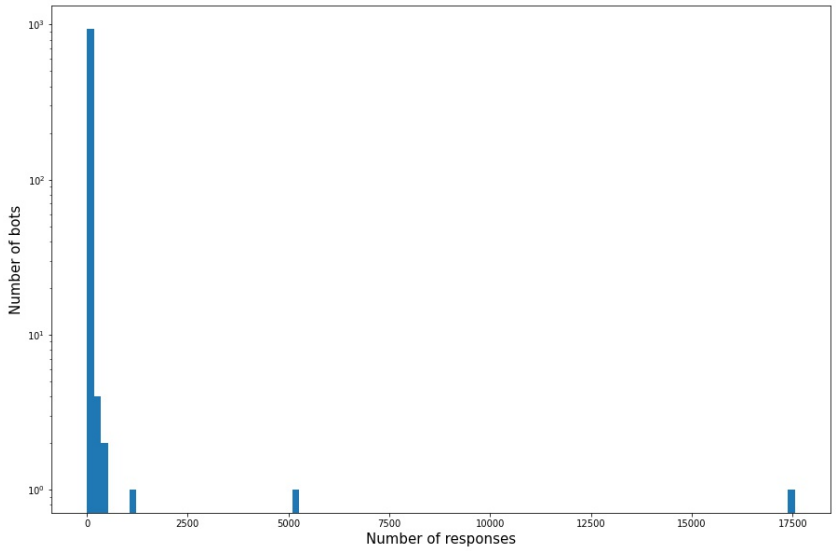

Figure 6.3.: Distribution of the number of responses sent by a bot

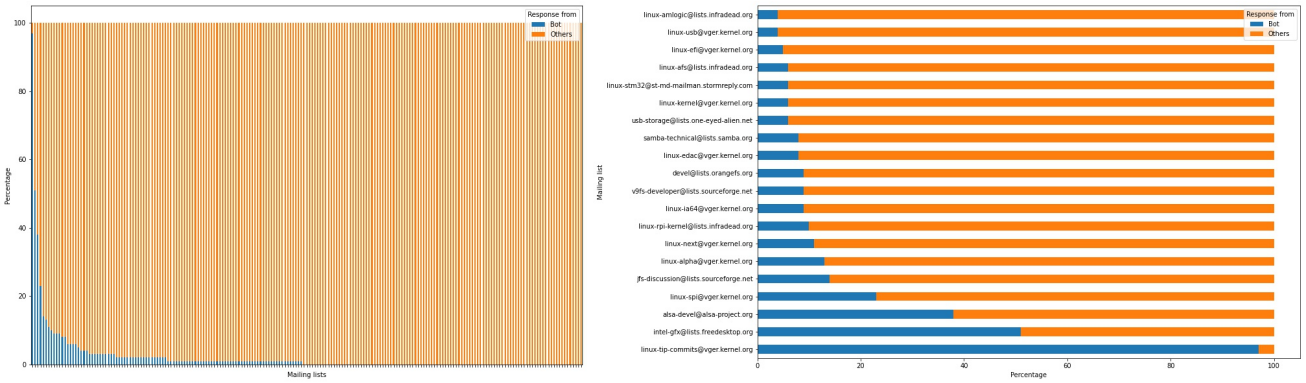

Figure 6.4.: Percentage of bot emails across mailing lists

project.org. However, the mailing list with the largest percentage of bot activity, linux-tip-commits@vger.kernel.org, has a dramatically smaller area of activity. Only 0.83% of all of the emails were sent to this particular mailing list, many of which coming from tip bots with the exception of a small amount of human activity.

8.42% of all emails were sent to the second mailing list in the ranking, which is intel-gfx@lists.freedesktop.org. As stated in the previous section, this is the mailing list to which the most active bot on the kernel development process, Patchwork, sends all of its emails. In point of fact, 50.76% of all of the emails sent to this mailing list were sent by Patchwork bot. Other bots which have sent emails to this list are Kernel Test Robot, tip-bot2 for alexey budankov and tip-bot2 for qian cai.

The third mailing list on the bot activity percentage ranking, alsa-devel@alsa-project.org, has a significant portion of its email activity, precisely 48.01%, coming from a single user, Mark

Brown. Nonetheless, 77.84% of this users emails which are sent to alsa-devel@alsa-project.org were sent by bots. The bot for Mark Brown was indeed the second most active bot on all of the mailing lists.

7. Conclusion

Review process in the Linux kernel development has a diverse set of variables which define it. Noteworthy activity from both bots and humans are observed and investigated throughout this thesis. There was no evidence found suggesting that developer experience has a positive effect on the amount of review. However, maintainers are proved to receive larger numbers of responses per patch when compared to others.

Enough evidence to reject that any area in the kernel is subjected to equal amount of human review is found both through patch authors' activity areas and the areas to which patches are submitted to. Remarkable bot activity from bots such as Kernel Test Robot and Patchwork is also observed in some areas in the development process.

A. Clustering Developers

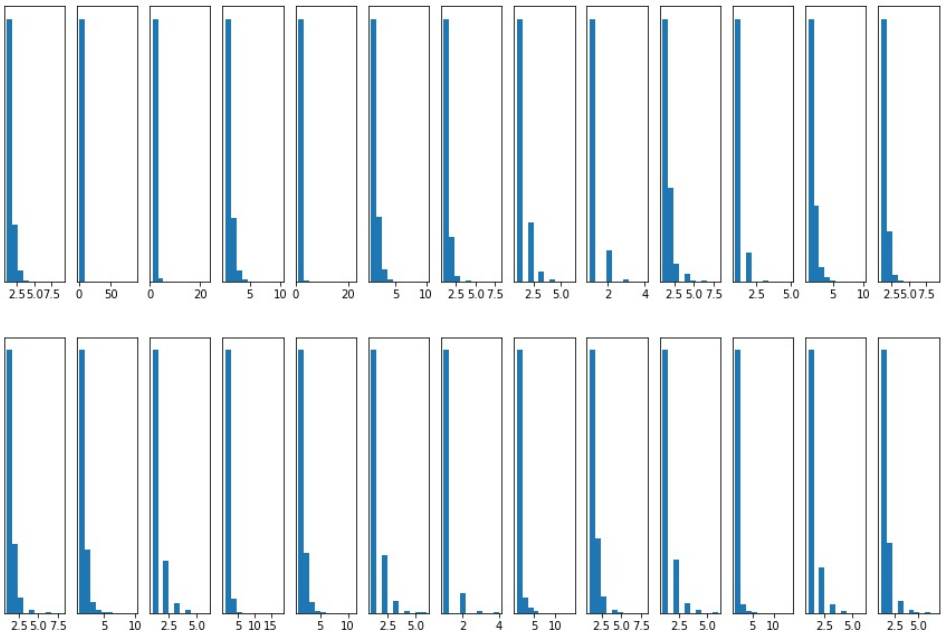

Figure A.1.: The distribution of average number of responses per patch in each cluster

B. Clustering Patches Using Mailing Lists

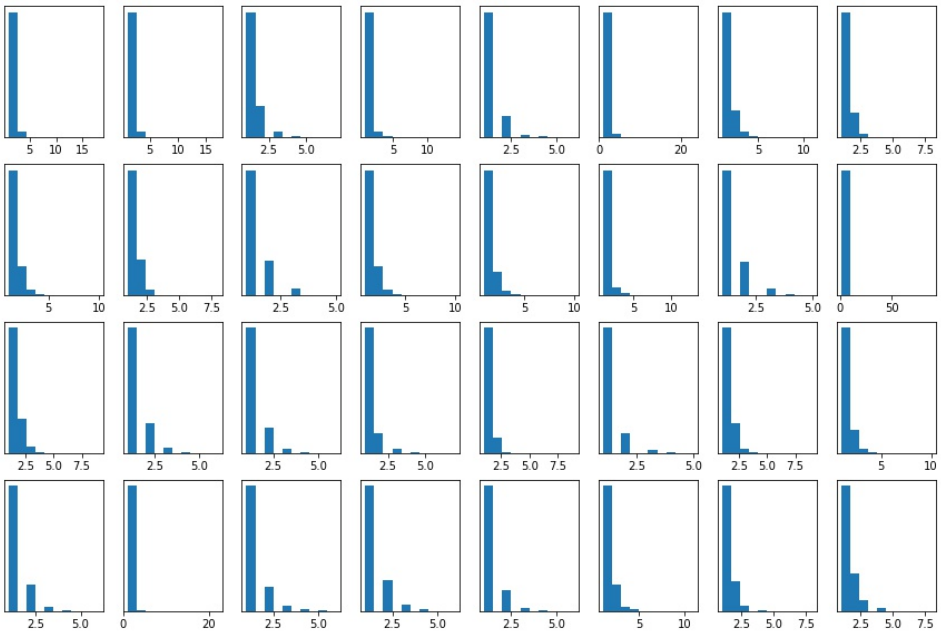

Figure B.1.: The distribution of average number of responses per patch in each cluster

C. Clustering Patches Using Maintainer Sections

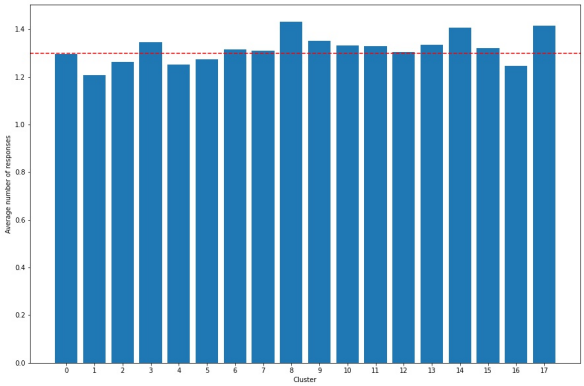

Figure C.1.: The average number of responses per patch in each cluster

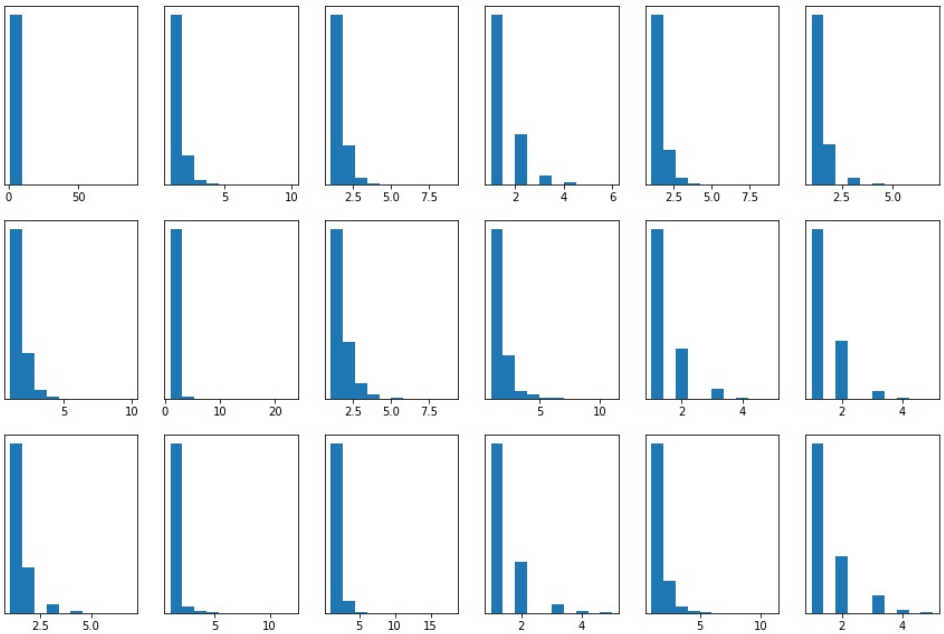

Figure C.2.: The distribution of average number of responses per patch in each cluster

List of Figures

List of Tables

Bibliography

[1] *Submitting patches: the essential guide to getting your code into the kernel*. 2021. URL: https://www.kernel.org/doc/html/latest/process/submitting-patches.html (visited on 02/23/2021).

[2] *How the development process works*. 2021. URL: https://www.kernel.org/doc/html/latest/process/2.Process.html#the-lifecycle-of-a-patch (visited on 03/08/2021).

[3] J. Corbet. *Gitdm (the "git data miner")*. git://git.lwn.net/gitdm.git. 2020.

[4] J. Corbet. *Gitdm (the "git data miner")*. https://lwn.net/Articles/290957/. 2008.

[5] *HOWTO do Linux kernel development*. 2021. URL: https://www.kernel.org/doc/html/latest/process/howto.html (visited on 02/23/2021).

[6] S. Khan. personal communication. Nov. 3, 2020.

[7] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2016.

[8] T. Lumley. *Biostatistics: a methodology for the health sciences*. Wiley-Interscience, 2004.