Preliminary exploratory analysis: PMD Alerts + Commits

Bruno Crotman

20/10/2019

Contents

1 Introduction

This document shows the how to collect PMD alerts related to each commit of a project and introduces an analysis about this information.

In the Section 2, this document explains briefly how PMD Source Code Analyzer works and which kind of alerts it generates. In the Section 3 the data collection and a brief analysis using the project Twitter4J is shown. The section 4 discusses the conclusions and potential next steps for the work aiming an article or a PhD thesis.

2 PMD Alerts

PMD is a source code analyzer that finds common flaws, like unused variables, empty catch blocks, unnecessary object creation and so forth (Dangel 2019).

In Java, the programing language in which Twittter4J is written, there are 8 groups of rules that generate alerts:

- Best Practices: enforce generally accepted best practices, e.g. the usage of foreach instead of for when applicable, avoidance of hard coded IP addresses, avoidance of the reassignment of loop variables.
- Code Style: enforce specific ccode style, e.g, the non static classes must declare at leas one constructor, control statement must have braces.
- Design: e.g. abstract classses must have methods, classes must not have too many fields or methods
- Documentation: e.g. the comments' size must be within certain limits
- Error prone: detect constructs that are either broken, extremely confusing or prone to runtime errors, e.g. assigment to operands, if statements whose conditionals are always true or always false.

PMD lets you choose which types of alerts to use and even create new rules. In this work a “quick start” ruleset, that comes as a basic configuration, is used.

3 Preliminary analysis

In this preliminary analysis, the GitHub repository related to Twitter4J project (Twitter4J 2019) is cloned and each commited code is analyzed by PMD (). The information is

3.1 Collecting Data

The following code collects data from every commit made to Twitter4J project.

Each commit is checked out and the PMD alerts are generated.

In the end, all the PMD alerts are stored.

```
unlink("repository/Twitter4J", recursive = TRUE, force = TRUE)

clone("https://github.com/Twitter4J/Twitter4J.git","repository/Twitter4J")

repository <- repository("repository/Twitter4j")

iteracao <- 0

check_out_and_pmd <- function(commit){

    iteracao <<- iteracao + 1
    print(iteracao)
    temp <- (paste0(UUIDgenerate(),".csv"))
    print(commit$sha)
    checkout(commit, force = TRUE)
    shell(paste0("pmd/bin/pmd.bat -d repository/Twitter4J -f csv -R rulesets/java/quickstart.xml -cache
    conteudo <- read_csv(temp) %>%
        mutate(sha = commit$sha)
    file.remove(temp)
    conteudo

}


line_count <- function (commit){

    iteracao <<- iteracao + 1
    print(iteracao)
    print(commit$sha)
    checkout(commit, force = TRUE)

    result <- shell("git -C repository/Twitter4J diff --stat 4b825dc642cb6eb9a060e54bf8d69288fbee4904",
        enframe(value = "word") %>%
        separate(col = word, into = c("file","number"), sep = "\\|") %>%
        filter(str_detect(file, ".java")) %>%
        mutate(lines = str_extract(number, "[0-9]+")) %>%
        filter(!is.na(lines)) %>%
        mutate(lines = as.integer(lines)) %>%
        summarise(lines = sum(lines)) %>%
        mutate(sha = commit$sha)


    result

}


check_out_and_pmd_possibly <- possibly(check_out_and_pmd, otherwise = tibble(sha = "erro"))

commits <-  commits(repository) %>%
    map_df(as_tibble)

saveRDS(commits, file =  "commits.rds")

commits_alerts <-  commits(repository) %>%
    map_dfr(check_out_and_pmd_possibly)

saveRDS(commits_alerts, file =  "commits_alerts.rds")

line_count_possibly <- possibly(line_count, otherwise = tibble(sha = "erro"))

line_counts <-  commits(repository) %>%
    map_df(line_count_possibly)

saveRDS(line_counts, file =  "line_counts.rds")

tags_hist_author <- tags(repository) %>%
  map(function(x) x$author ) %>%
  map(function(x) as.character(x$when) ) %>%
  enframe() %>%
  filter(str_detect(value,"[2]"))

tags_hist_tagger <- tags(repository) %>%
  map(function(x) x$tagger ) %>%
  map(function(x) as.character(x$when) ) %>%
  enframe() %>%
  filter(str_detect(value,"[2]"))

tags_hist <- bind_rows(tags_hist_author,tags_hist_tagger) %>%
  rename(
    version = name,
    date = value
  ) %>%
  separate(col = version, into = c("major", "minor", "patch"), sep = "\\.") %>%
  mutate(
    date = ymd_hms(date)
  ) %>%
  arrange(date) %>%
  mutate(
    lag_major = lag(major),
    lag_minor = lag(minor)
  ) %>%
  mutate(
    major_changed = if_else(lag_major != major, T, F),
    minor_changed = if_else(lag_minor != minor | major_changed , T, F)
  )



saveRDS(tags_hist, file =  "tags_hist.rds")
```

Data can be loaded from the files that serve as cache

```
commits_from_cache <- read_rds("commits.rds")

alerts_from_cache <- read_rds("commits_alerts.rds")

lines_from_cache <- read_rds("line_counts.rds")

tags_from_cache <- read_rds("tags_hist.rds")


commits_com_tag <- commits_from_cache %>%
  arrange(when) %>%
  mutate(
    next_commit = lead(when)
  ) %>%
  crossing(tags_from_cache) %>%
  filter(date >= when, date < next_commit  ) %>%
  select(
    sha,
    major,
    minor,
    patch,
    major_changed,
    minor_changed
  ) %>%
  mutate(
    version = paste(major, minor, patch, sep = ".")
  )




all_data <-  commits_from_cache %>%
    left_join(commits_com_tag, by = c("sha")) %>%
    inner_join(alerts_from_cache, by = c("sha")) %>%
    inner_join(lines_from_cache, by = c("sha"))
```

3.2 Evolution of the code size

After an instability, it seems that the project evolves from the commit 300 until aproximately commit 1450, until the release of version 3.0.0. After that the number of lines of code ceases to increase, and so do the number of alerts and the number of alerts per line of code.

Figure 1 shows this evolution, in terms of PMD alerts per line of code, total PMD alerts and total lines of code.

```r
alerts_total <- all_data %>%
    group_by(sha, when) %>%
    summarise(
      alerts_per_line = n()/mean(lines),
      lines = mean(lines),
      alerts = n() ,
      version = first(version),
      major_changed = first(major_changed),
      minor_changed = first(minor_changed)
    ) %>%
    ungroup() %>%
    arrange(when) %>%
    mutate(commit = row_number())

alerts_evolving <-  alerts_total %>%
    filter(between(commit,300, 1450))


p1 <- ggplot(alerts_total) +
    geom_area(aes(x = commit, y = alerts_per_line ), fill = "gray") +
    geom_text(
      data = alerts_total %>%  filter(minor_changed),
      aes(x = commit, y = alerts_per_line + 0.006, label = version ),
      color = "red",
      size = 2,
      family = "CMU Serif"
    ) +
    geom_area(data = alerts_evolving, aes(x = commit, y = alerts_per_line ), fill = "darkred") +
    geom_point(
      data = alerts_total %>% filter(minor_changed),
      aes(x = commit, y = alerts_per_line ),
      color = "red",
      size = 2
    ) +
    geom_point(
      data = alerts_total %>% filter(!is.na(version)),
      aes(x = commit, y = alerts_per_line ),
      color = "red",
      size = 1
    ) +
    theme_minimal() +
    scale_x_continuous(breaks = seq(0,to = 2500, by = 200)) +
    ggtitle("Alerts per line x Commit") +
    labs(x = "", y = "Alerts per line") +
    theme(text =  element_text(size = 8, family = "CMU Serif"))


p2 <- ggplot(alerts_total) +
    geom_area(aes(x = commit, y = alerts ), fill = "gray") +
    geom_area(data = alerts_evolving, aes(x = commit, y = alerts ), fill = "darkred") +
    geom_text(
      data = alerts_total %>%  filter(minor_changed),
      aes(x = commit, y = alerts + 300, label = version ),
      size = 2,
      color = "red",
      family = "CMU Serif"
    ) +
    geom_point(
      data = alerts_total %>% filter(minor_changed),
      aes(x = commit, y = alerts ),
      color = "red",
      size = 2
    ) +
    geom_point(
      data = alerts_total %>% filter(!is.na(version)),
      aes(x = commit, y = alerts ),
      color = "red",
      size = 1
    ) +
    theme_minimal() +
    scale_x_continuous(breaks = seq(0,to = 2500, by = 200)) +
    ggtitle("Alerts x Commit") +
    labs(x = "", y = "Alerts") +
    theme(text =  element_text(size = 8, family = "CMU Serif"))


p3 <- ggplot(alerts_total) +
    geom_area(aes(x = commit, y = lines ), fill = "gray") +
    geom_area(data = alerts_evolving, aes(x = commit, y = lines ), fill = "darkred") +
    geom_text(
      data = alerts_total %>%  filter(minor_changed),
      aes(x = commit, y = lines + 13000, label = version ),
      size = 2,
      color = "red",
      family = "CMU Serif"
    ) +
    geom_point(
      data = alerts_total %>% filter(minor_changed),
      aes(x = commit, y = lines ),
      color = "red",
      size = 2
    ) +
    geom_point(
      data = alerts_total %>% filter(!is.na(version)),
      aes(x = commit, y = lines ),
      color = "red",
      size = 1
    ) +
    theme_minimal() +
    scale_x_continuous(breaks = seq(0,to = 2500, by = 200))+
    ggtitle("Lines x Commit") +
    labs(x = "Commit", y = "Lines") +
    theme(text =  element_text(size = 8, family = "CMU Serif"))
```

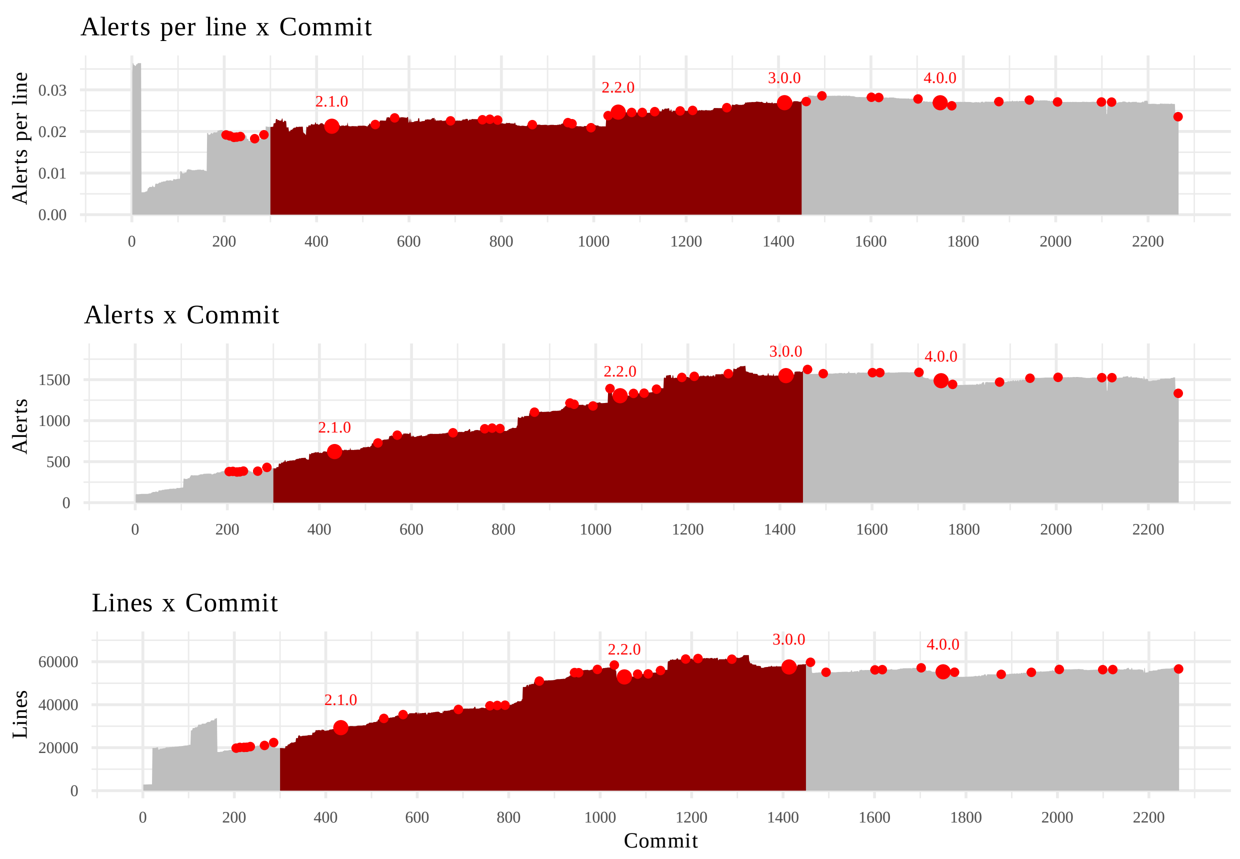


Figure 1: Project evolution

```
grid.arrange(p1, p2, p3, nrow = 3)
```

Figure 2 shows the number of commits made in each quarter.

```
commits_per_week <- alerts_total %>%
    mutate(week = as.Date(round_date(when, "quarter")) ) %>%
    group_by(week) %>%
    count(n = n())

commits_per_week_evolving <- alerts_evolving %>%
    mutate(week = as.Date(round_date(when, "quarter") )) %>%
    group_by(week) %>%
    count(n = n())

ggplot(commits_per_week) +
    geom_col(aes(x = week, y = n)) +
    geom_col(data = commits_per_week_evolving, aes(x = week, y = n), fill = "darkred") +
    scale_x_date(
      date_breaks = "year",
      date_labels = "%Y"
    ) +
    ggtitle("Commits per quarter") +
```

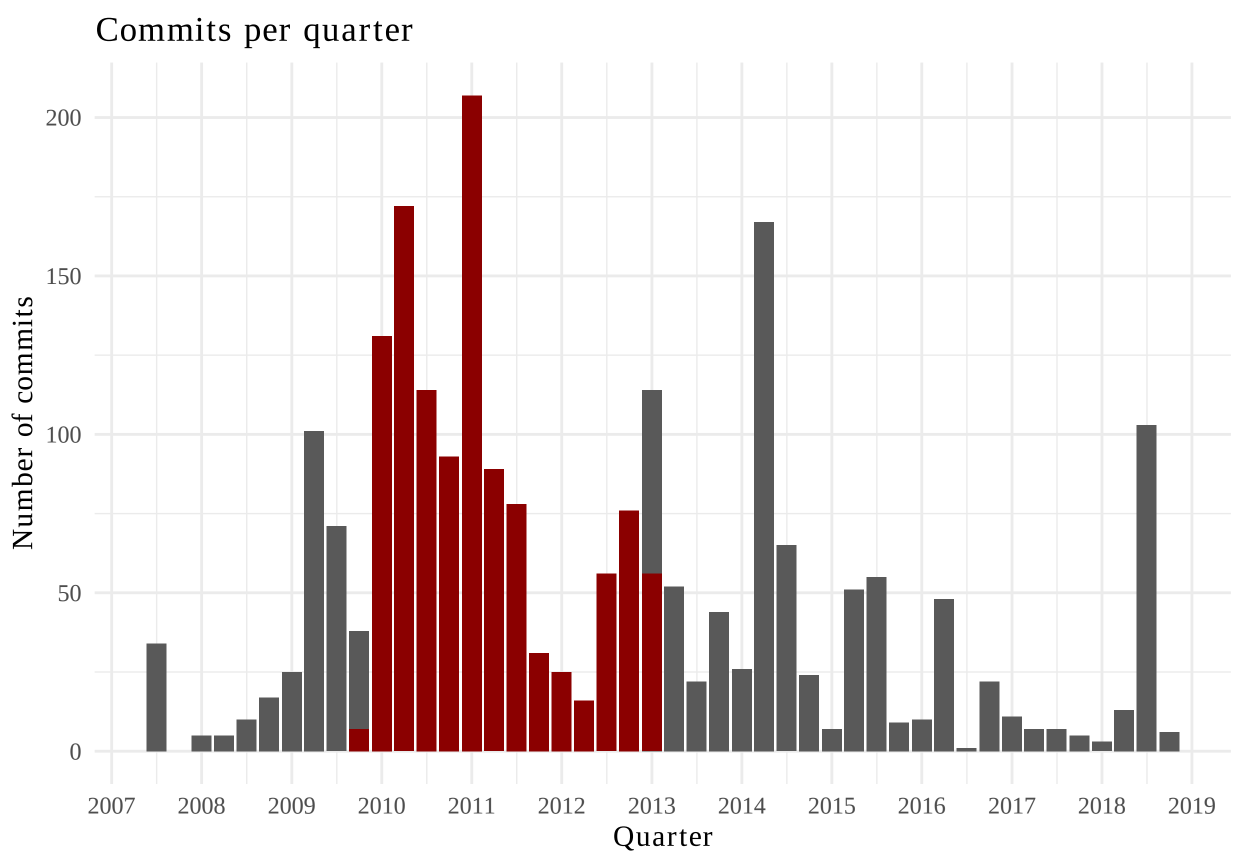


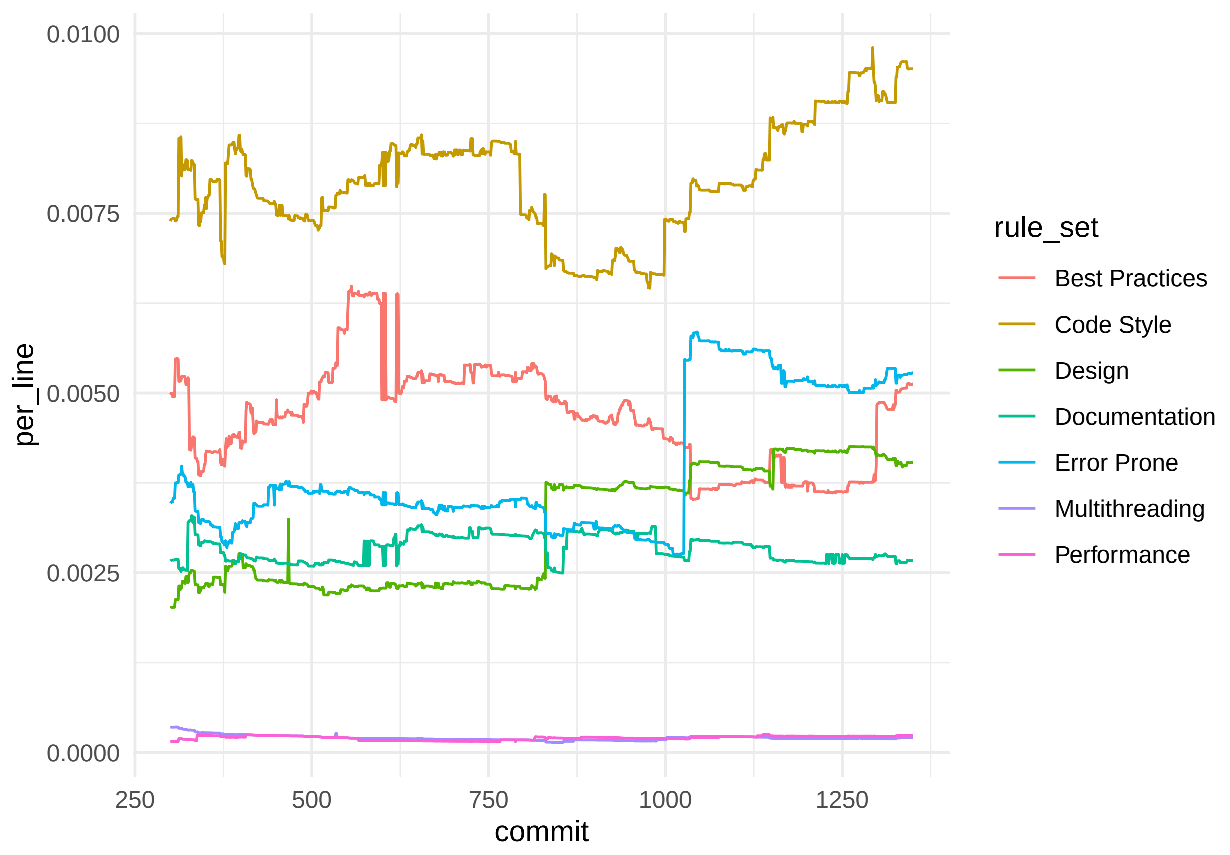
Figure 2: Project evolution

```
    theme_minimal() +
    theme(text = element_text(family = "CMU Serif")) +
    labs(y = "Number of commits", x = "Quarter")
```

3.3 Types of alert

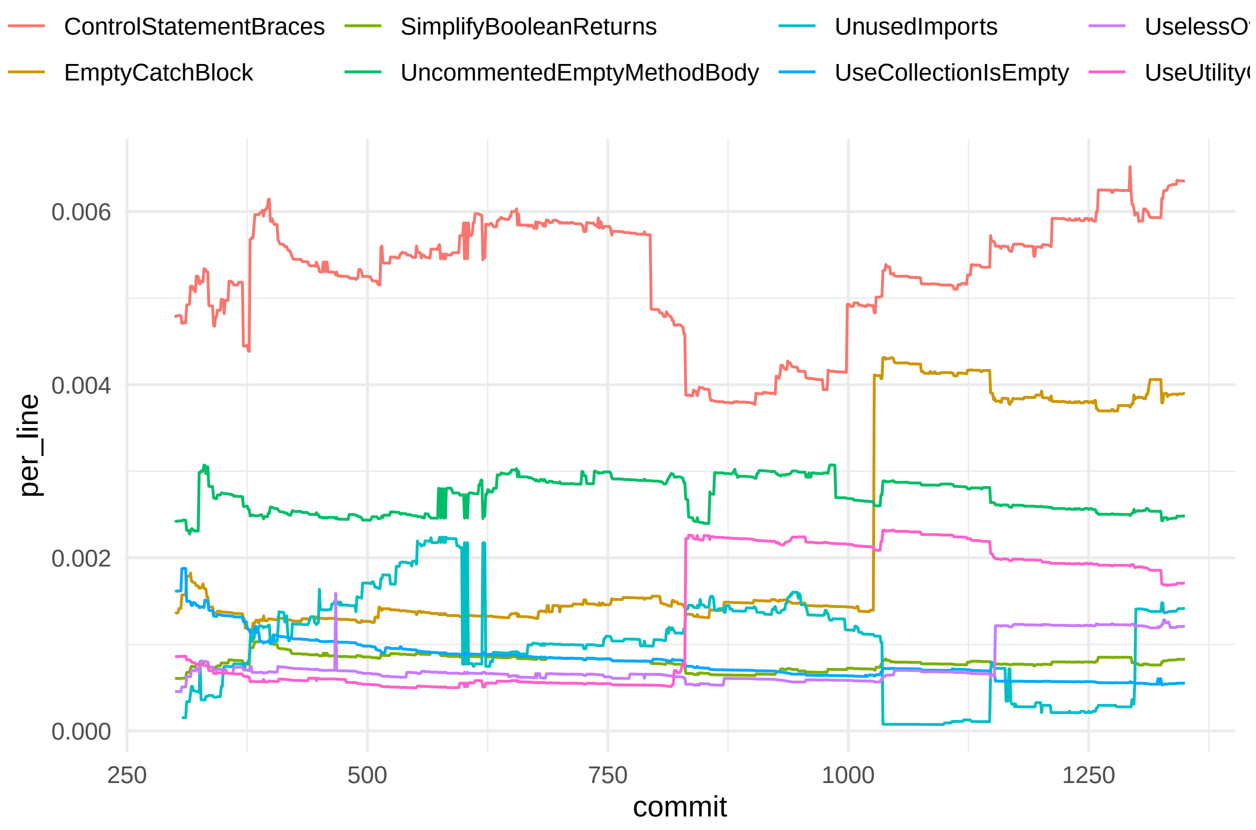
```
rule_set <- all_data %>%
    rename(
        rule_set = `Rule set`
    ) %>%
    select(sha, when, author, Package, File, Priority, rule_set, Rule, lines) %>%
    inner_join(alerts_total, by = c("sha")) %>%
    mutate(per_line = 1/lines.x) %>%
    group_by(commit, rule_set) %>%
    summarise(per_line = sum(per_line)) %>%
    filter(between(commit, 300, 1350 ))


ggplot(rule_set) +
    geom_line(aes(color = rule_set, x = commit, y = per_line)) +
    theme_minimal()
```



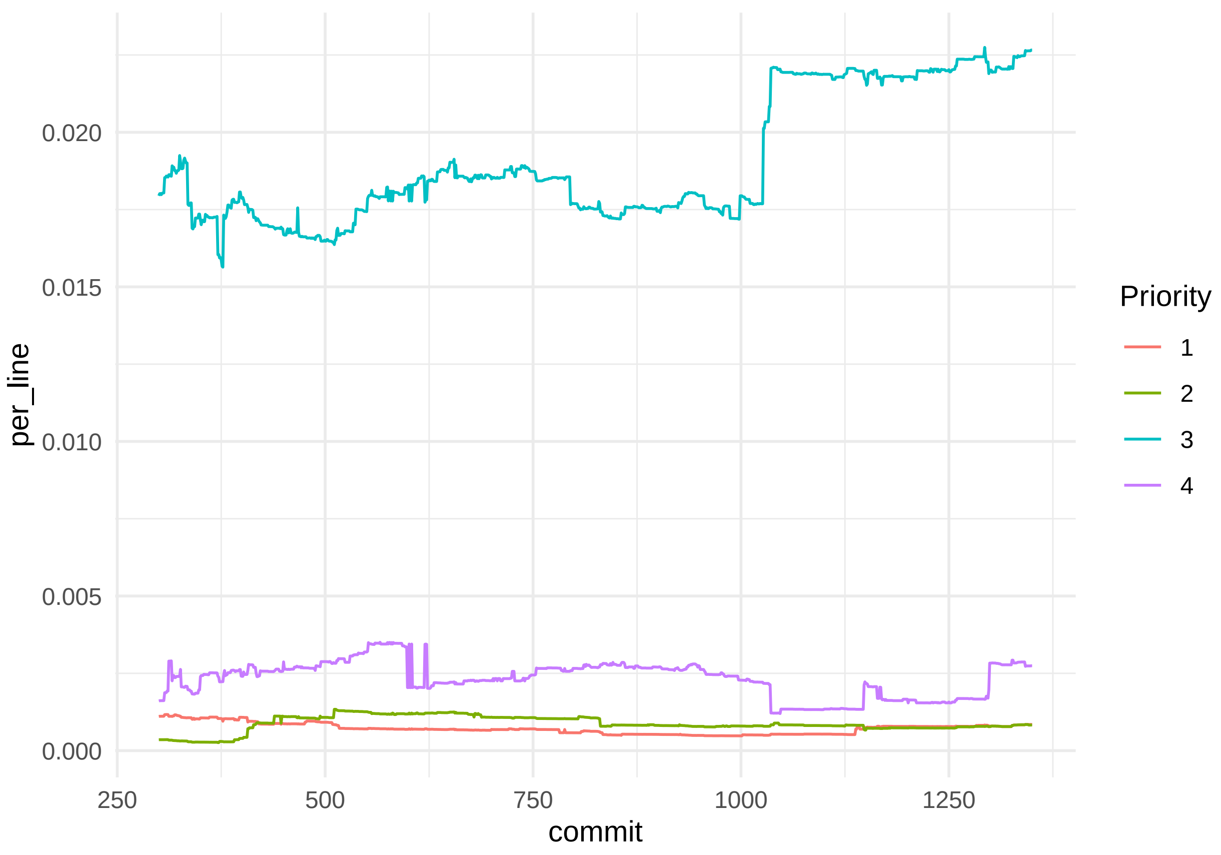
```
rule <- all_data %>%
    rename(
        rule_set = `Rule set`,
        rule = Rule
    ) %>%
    select(sha, when, author, Package, File, Priority, rule_set, rule, lines) %>%
    inner_join(alerts_total, by = c("sha")) %>%
    mutate(per_line = 1/lines.x) %>%
    group_by(commit, rule) %>%
    summarise(per_line = sum(per_line)) %>%
    ungroup() %>%
    filter(between(commit, 300, 1350 )) %>%
    mutate(rule = fct_lump(rule, n = 8, w = per_line, other_level = "Other"))


ggplot() +
    geom_line(data = rule %>% filter(rule != "Other"), aes(color = rule, x = commit, y = per_line)) +
    theme_minimal() +
    theme(legend.position = "top")
```



```
rule_set <- all_data %>%
    rename(
        rule_set = `Rule set`
    ) %>%
    select(sha, when, author, Package, File, Priority, rule_set, Rule, lines) %>%
    inner_join(alerts_total, by = c("sha")) %>%
    mutate(per_line = 1/lines.x) %>%
    group_by(commit, Priority) %>%
    summarise(per_line = sum(per_line)) %>%
    ungroup() %>%
    mutate(Priority = as.factor(Priority)) %>%
    filter(between(commit, 300, 1350 ))


ggplot(rule_set) +
    geom_line(aes(color = Priority, x = commit, y = per_line)) +
    theme_minimal()
```



4 Conclusions and next steps

5 References

Dangel, Andreas et. al. 2019. "PMD Source Code Analyzer." 2019. https://pmd.github.io/.

Twitter4J. 2019. "Twitter4J." 2019. http://twitter4j.org/en/.