TravisCI Identification

Overview

This study pulls a sample of 400 repos using the GitHub Archive Events data. Previous analysis has shown that samples taken from Watch events result in more established repositories with a higher number of contributors. Initial CI usage identification resulted in a high proportion of repositories using TravisCI over other other CI tools. It is possible that TravisCI is simply the easiest to identify, however this finding is consistent with other research. (TODO: cite sources)

This study is a vital first step in the next phases of research that examine how repositories are impacted when they start using CI and what factors would indicate a high probability of a repository adopting CI that currently doesn't use it.

The key questions explored in this study are:

- How can we best identify whether a repository is using TravisCI?
- Based on the TravisCI identification, can we hypothesize on how to identify whether a repository that is not using TravisCI is using another CI tool?

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(bigrquery)
library(ggplot2)
library(reshape2)

project <- "bonnyci-github-archive"
```

CI Identification

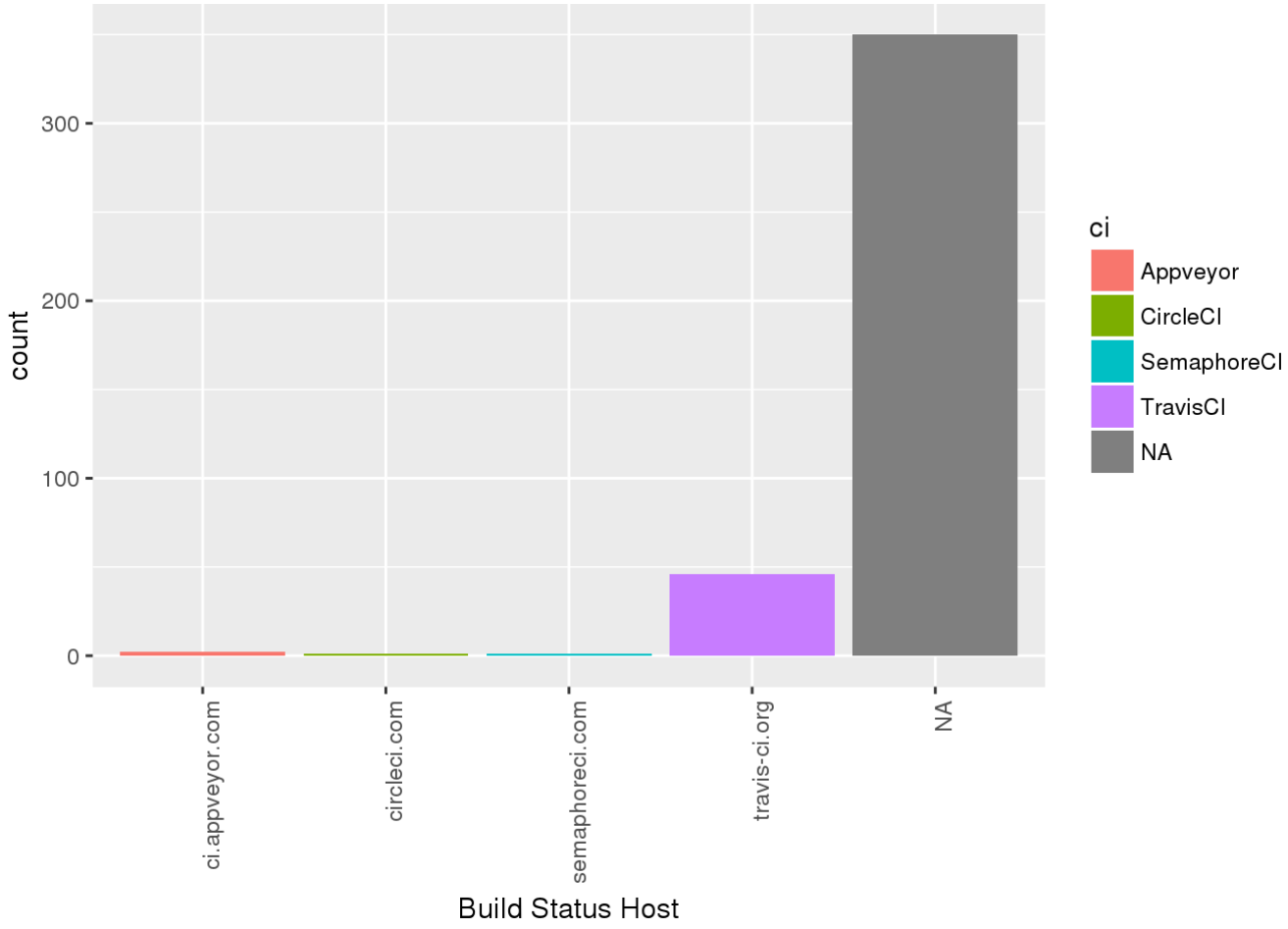
Build Status tag in README

An initial attempt to identify CI usage was done by evaluating the existance of a “build status” tag in the README and extracting the host. Because this method only resulted in a small number of repositories being identified as using CI, additional methods are explored here.

The key advantage of this method is the ability to identify what CI systems are possibly in use. The disadvantage is it isn’t the most accurate. Depending on how the build status tag is structured, this method may pick up the wrong hostname and require manual correction. Additionally, a repo may not have a README or may not be using a Build Status tag at all and they would then not be identified.

```
# plot of repositories with build status tags and hostnames therein

# repo_readme_sql <-
#   'select * from [bonnyci-github-archive:ci_plunder_travisci.repo_readme_buildstatu
s]'
#
# repo_readme <- query_exec(repo_readme_sql, project = project)

# fix fields that were wrongly identified
# repo_readme <- repo_readme %>%
#   mutate(
#     build_status_host = ifelse(repo_slug == "ruyadorno/dom-i18n", "travis-ci.org", b
uild_status_host)
#   )

# repo_readme <- repo_readme %>% mutate(
#   ci=
#     ifelse(build_status_host=="travis-ci.org", "TravisCI",
#                     ifelse(build_status_host=="ci.appveyor.com", "Appveyor",
#                            ifelse(build_status_host=="circleci.com", "CircleCI",
#                                   ifelse(build_status_host=="semaphoreci.com", "Semap
horeCI", NA))
#                     )
#            )
# )
#
# saveRDS(repo_readme, "repo_readme.rds")
```

```
repo_readme <- readRDS("repo_readme.rds")

# repos with a build status in the readme
ggplot(data = repo_readme,
       aes(x = build_status_host, fill=ci)) +
  geom_bar() +
  xlab("Build Status Host") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

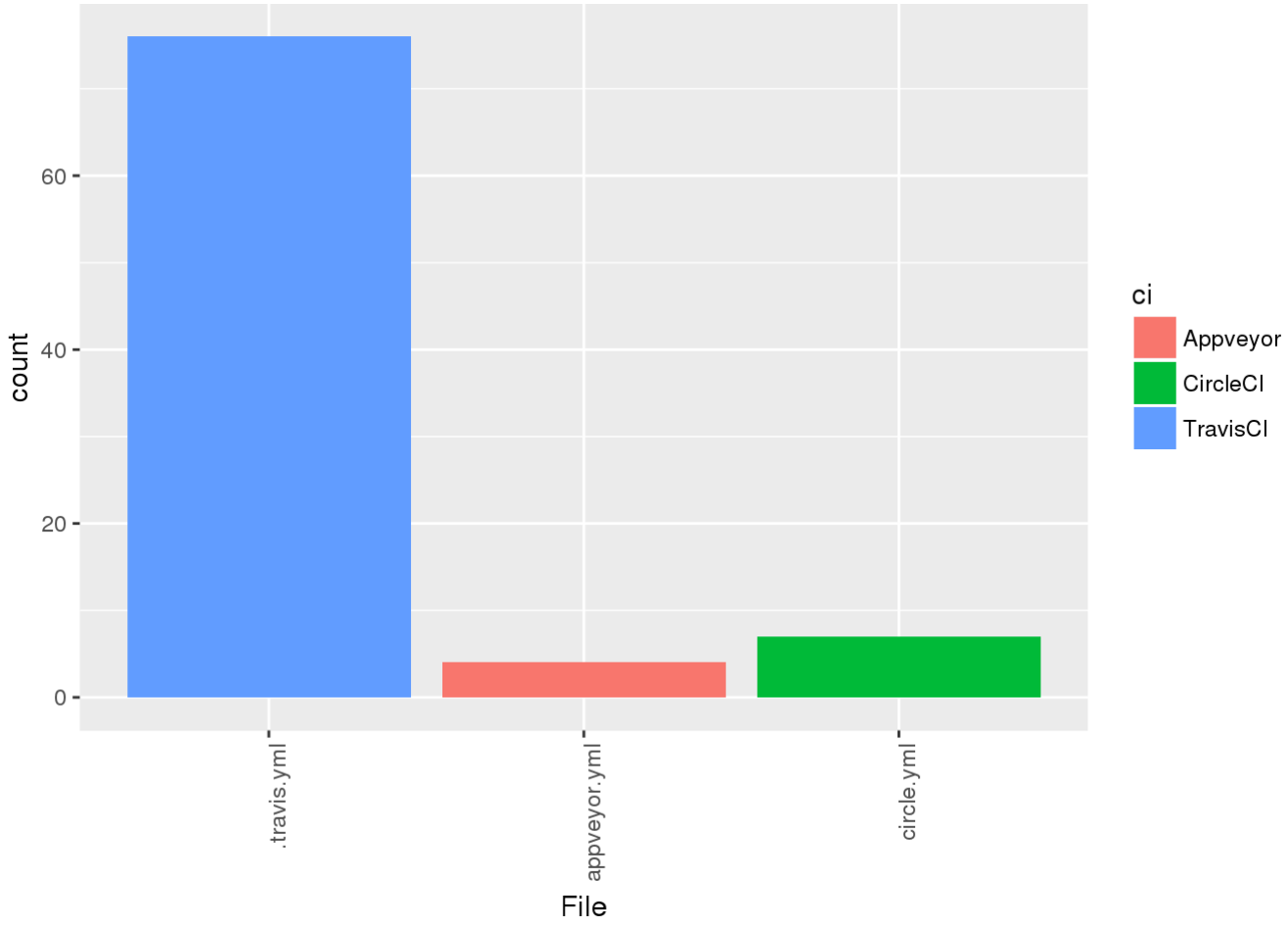


In-repo Configuration Files

The second method explored is searching the repository contents for a specific configuration file. This method is fairly accurate in that if the file exists, there is a strong possiblity the repo is either using the CI, has used it in the past, or intends to in the future. The file contents are retrieved from the Github API. The shortcomings of this method are that the CI must use in-repo configuration, the filename must be consistent and easily searchable across many repos, and the file must be in the same place in each repository (typically at the root level). This method had the best results in terms of numbers of repos identified as using CI, but the breadth of the identification is limited.

```
# plot of repositories with .travis.yml in file contents

# yml_ci_sql <-
#   'select * from [bonnyci-github-archive:ci_plunder_travisci.yml_ci]'
#
# yml_ci <- query_exec(yml_ci_sql, project = project)

# yml_ci <- yml_ci %>% mutate(
#   ci=
#     ifelse(name == ".travis.yml", "TravisCI",
#                   ifelse(name == "appveyor.yml", "Appveyor",
#                          ifelse(name =="circle.yml", "CircleCI", NA)
#                   )
#            )
# )

#saveRDS(yml_ci, "yml_ci.rds")
```

```
yml_ci <- readRDS("yml_ci.rds")

ggplot(data = yml_ci,
       aes(x=name, fill=ci)) +
  geom_bar() +
  xlab("File") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



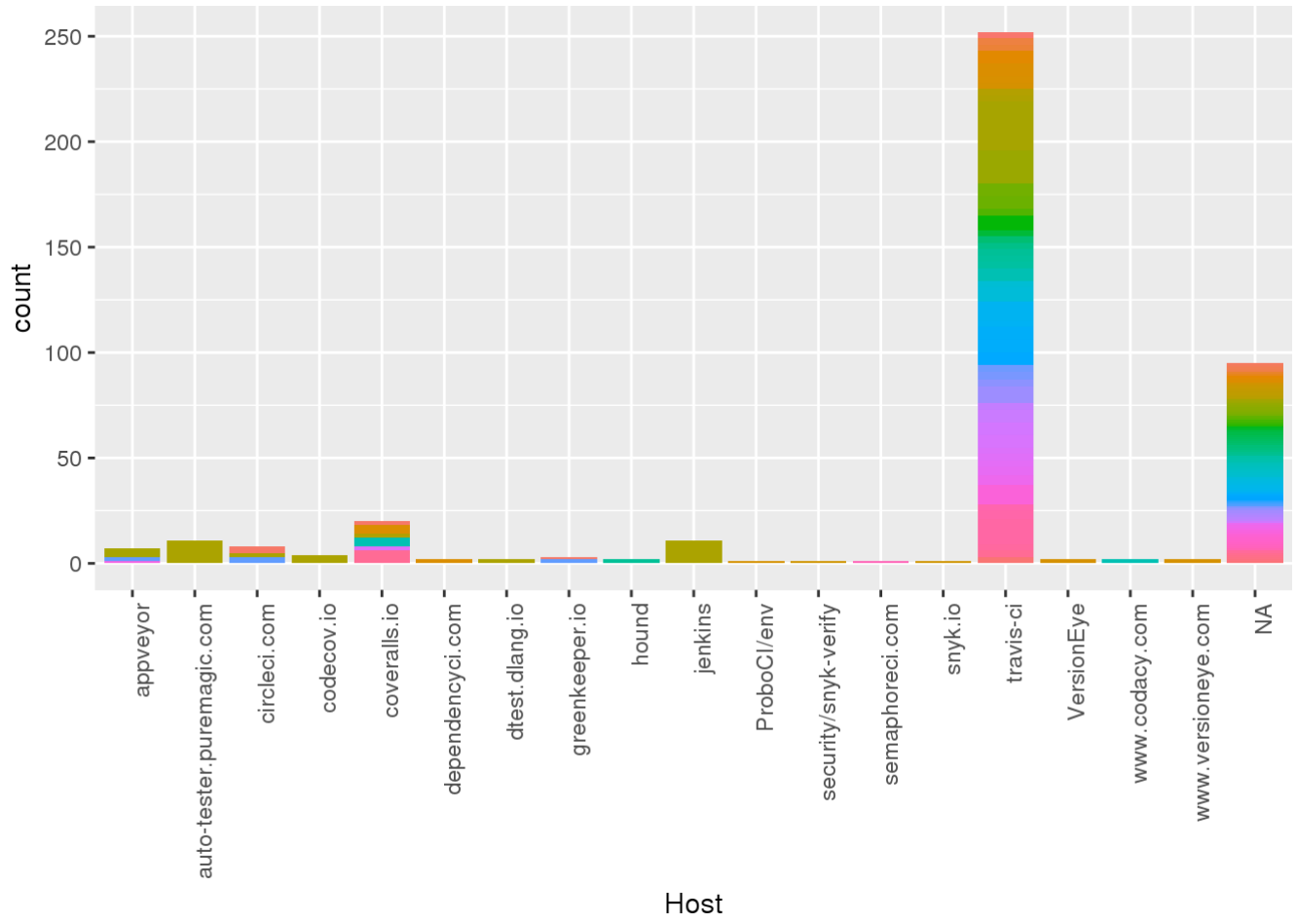
Statuses from Latest Pull Request

If a Github repository is integrated with an external service, that service will typically publish a status message when a new pull request is made. These statuses can be retrieved for public user-owned repositories and organization-owned repositories with relaxed pull access. For more information about Github statuses see https://developer.github.com/v3/repos/statuses/ (https://developer.github.com/v3/repos/statuses/)

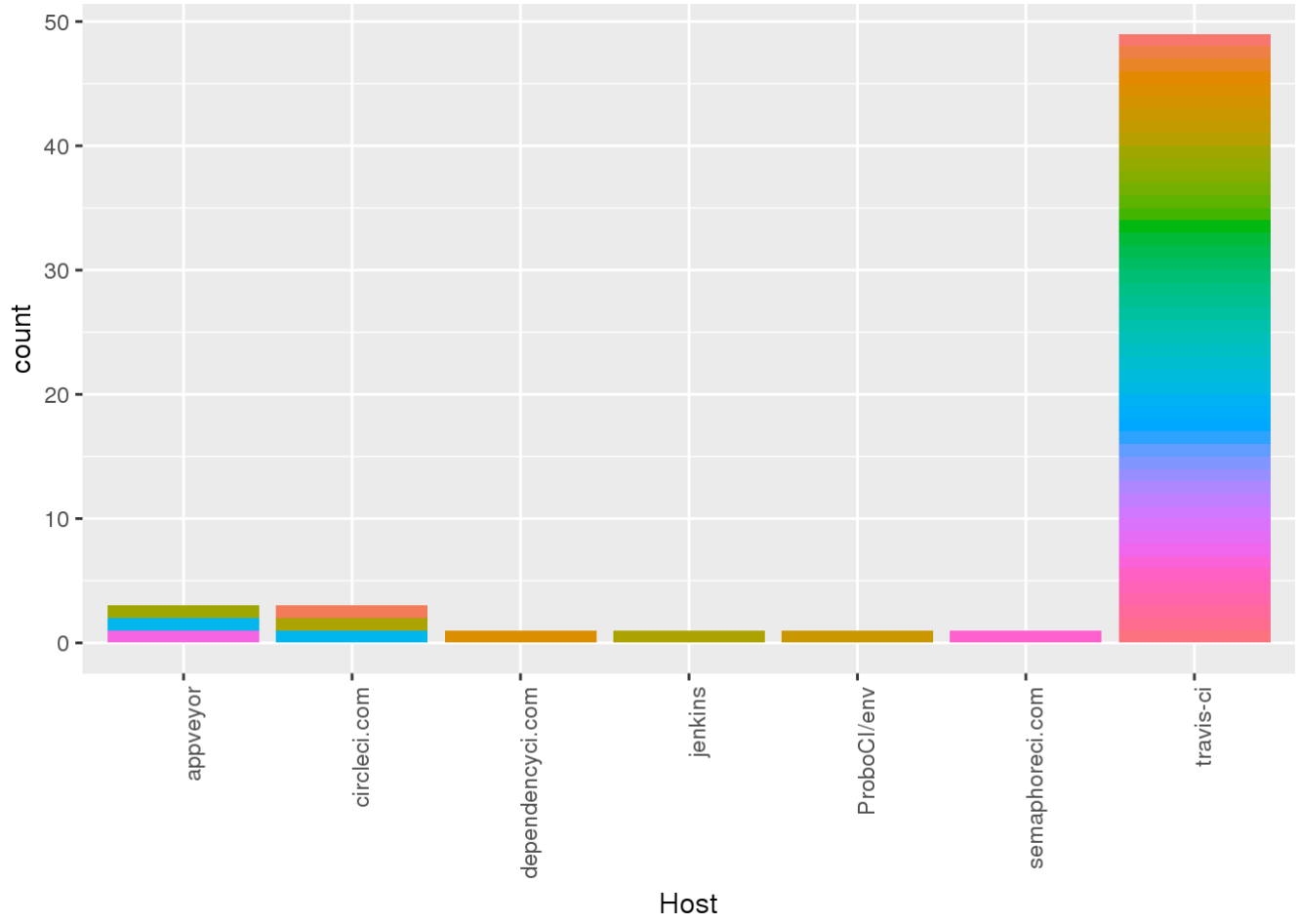
Status messages will clearly indicate if a repository is configured to use an external CI pipeline because the host information is provided in the data fields. Similar to the build status tag approach, the hosts have been extracted from these fields.

```
# statuses_sql <-
# 'select * from [bonnyci-github-archive:ci_plunder_travisci.pull_requests_statuses_ho
sts]'
#
# statuses <- query_exec(statuses_sql, project = project)
#

# is.na(statuses$context) <- statuses$context == ""
#
# statuses <- statuses %>% mutate(
#   host=ifelse(!is.na(context_host), context_host,
#               ifelse(!is.na(status_host), status_host, context))
# )
#
# saveRDS(statuses, "statuses.rds")
```

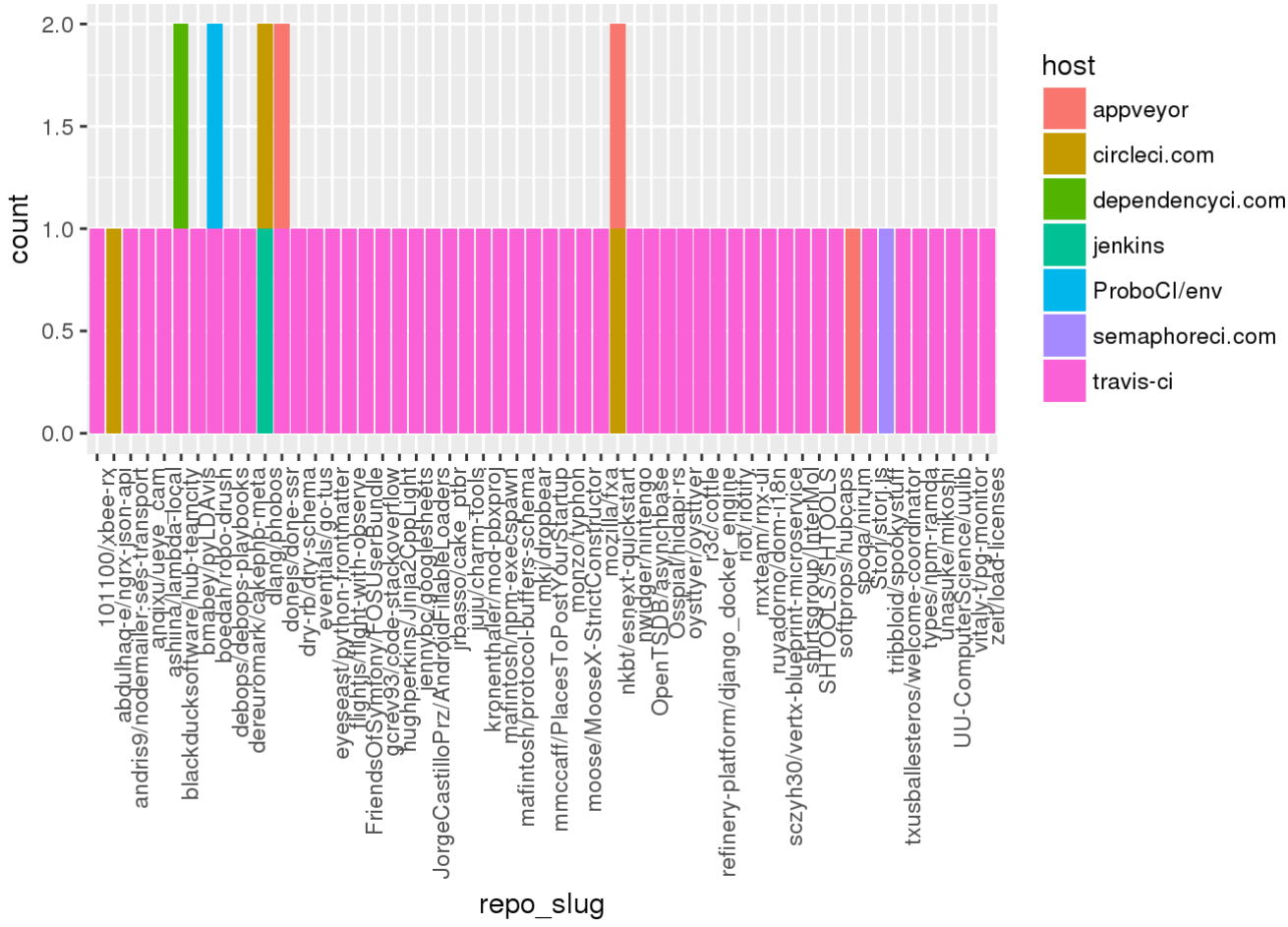
```
# all hosts identified through statuses
statuses <- readRDS("statuses.rds")

# repos with a build status in the readme
ggplot(data = statuses,
       aes(x = host, fill=repo_slug)) +
  geom_bar() +
  xlab("Host") +
  theme(legend.position="none") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
statuses_context_ci <- statuses %>% mutate(
  context_ci = grepl("ci|continuous-integration", context, ignore.case = TRUE)
) %>%
  filter(context_ci == TRUE)

statuses_context_ci_by_repo <- statuses_context_ci %>%
  group_by(repo_slug, host) %>%
  summarise(num_ci_statuses=n())

ggplot(data = statuses_context_ci_by_repo,
       aes(x = host, fill=repo_slug)) +
  geom_bar() +
  xlab("Host") +
  theme(legend.position="none") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
ggplot(data = statuses_context_ci_by_repo,
       aes(x = repo_slug, fill=host)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
# TODO: i really need to learn how to write R, this is just awful

statuses_ci <- statuses_context_ci_by_repo %>%
  mutate(
    ci= ifelse(host=="travis-ci", "TravisCI",
                       ifelse(host=="appveyor", "Appveyor",
                              ifelse(host=="circleci.com", "CircleCI",
                                     ifelse(host=="jenkins", "Jenkins",
                                            ifelse(host=="ProboCI/env", "ProboCI",
                                                   ifelse(host=="semaphoreci.com", "Se
maphoreCI",
                                                          ifelse(host=="dependencyci.c
om", "DependencyCI",
                                                                 NA)))))))
    )
```

A shortcoming of this method is that statuses cannot identify if a repository is using CI if the repository is not integrated with the external CI pipeline. This can happen if the repository isn't actually hosted on Github and is just mirrored or if the project has a custom CI pipeline that is external to Github (again, if the repository is mirrored).

We will look at this in the next section to see how many repositories that failed to be identified through this method were identified through the other methods.

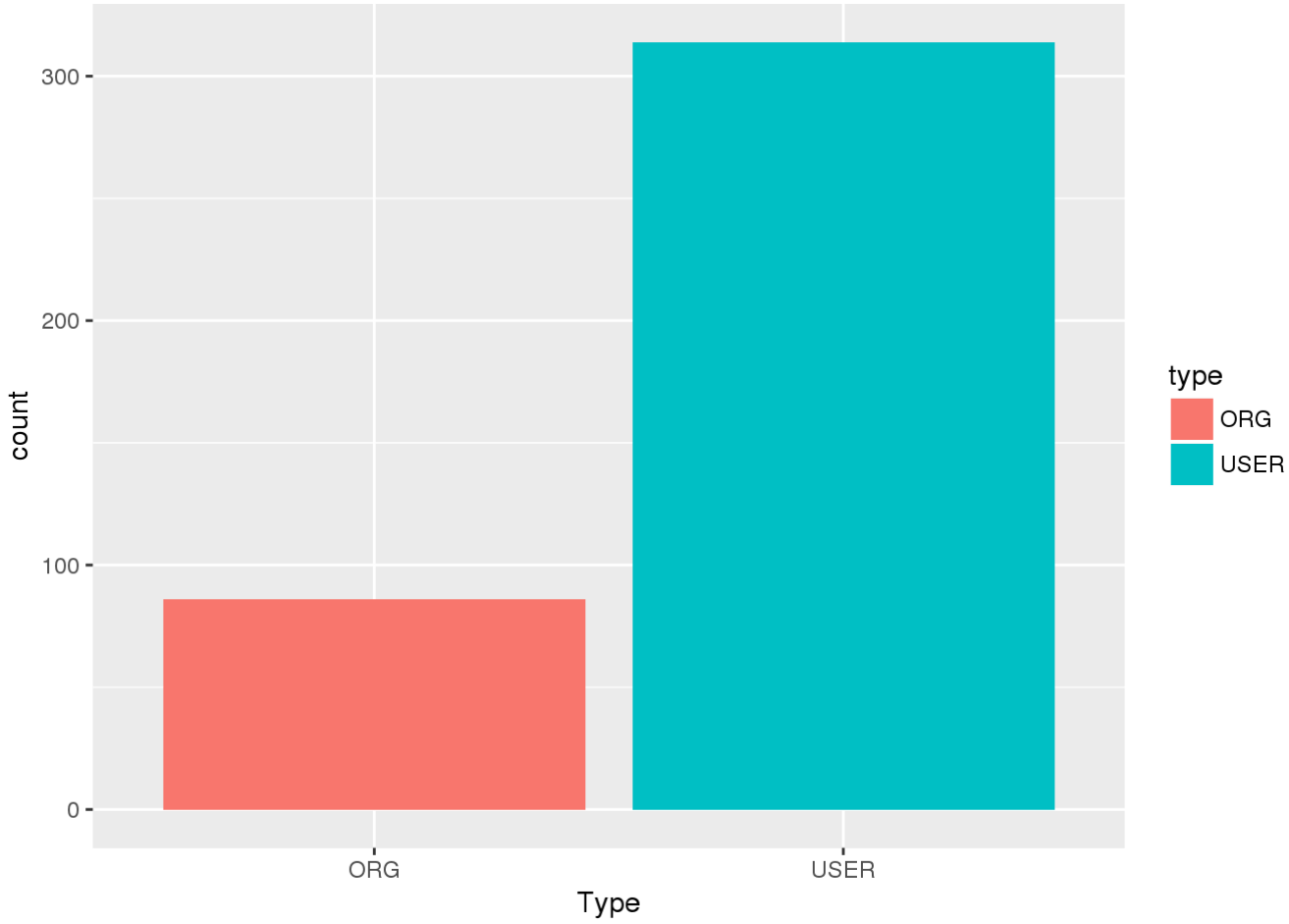
The other problem with status messages is that if an organization-owned repository is not configured to allow public read/pull access, then the only users who can get status information on pull requests have to have those permissions. This means organization-owned repositories may be misrepresented as not using CI through this method.

What proportion of the repositories in the sample are organization-owned?

Around 22% of the repositories were owned by organizations, just under 1/4 of the sample.

```
# orgs_sql <-
# 'select * from [bonnyci-github-archive:ci_plunder_travisci.ghevents_repo_names_org]'

# orgs <- query_exec(orgs_sql, project = project)
#
# orgs <- orgs %>% mutate(
#   type = ifelse(is.na(org), "USER", "ORG")
# )
#
# saveRDS(orgs, "orgs.rds")
```

```
orgs <- readRDS("orgs.rds")

ggplot(data = orgs,
       aes(x=type, fill=type)) +
  geom_bar() +
  xlab("Type")
```

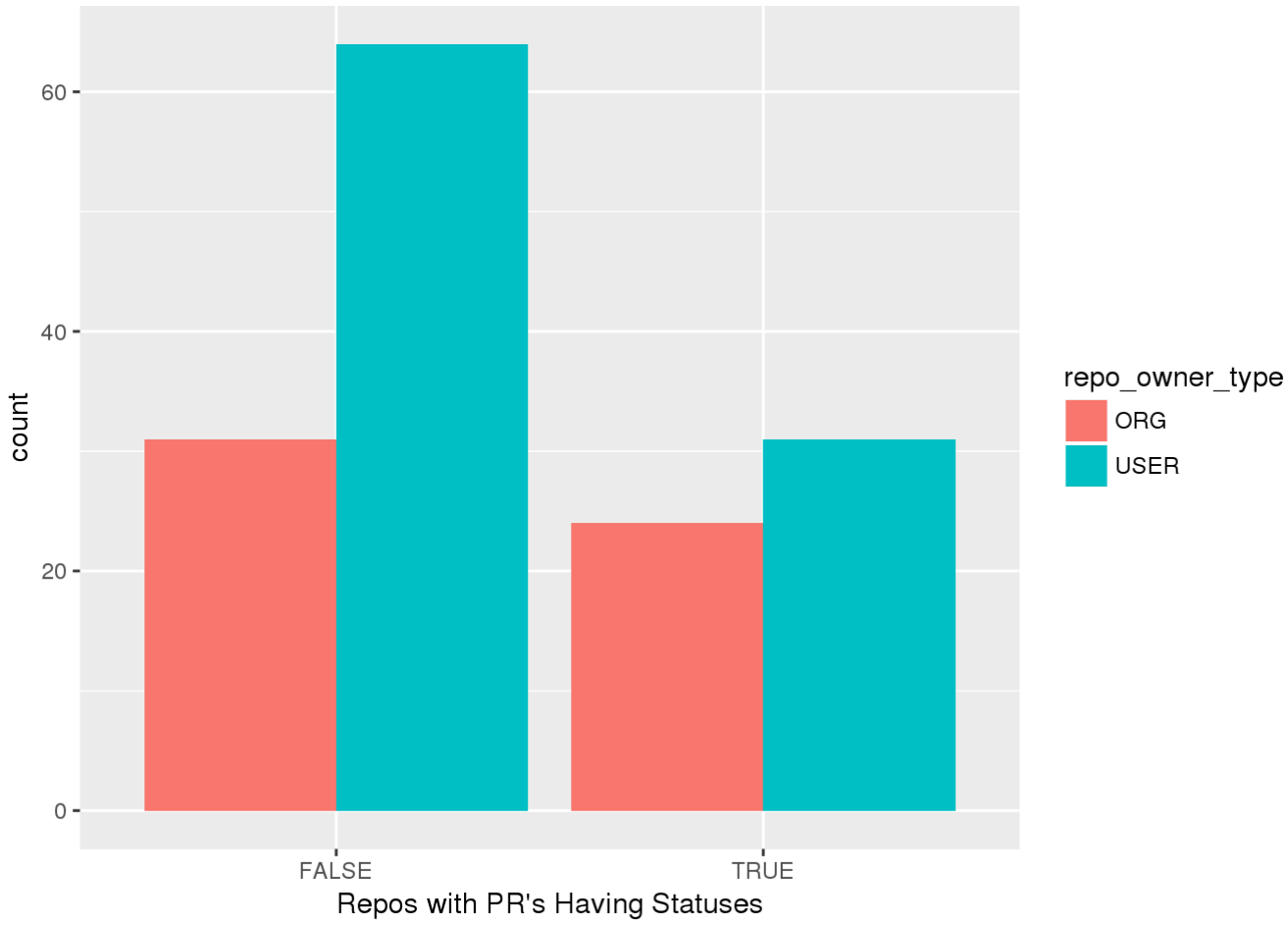


Of the repositories that had pull requests, how many had statuses and were they user- or organization-owned?

24 out of the 86 org-owned repos in the sample had both pull requests and statuses. 31 of the 316 user-owned repos in the sample had both pull requests and statuses. 55 out of the 400, or just under 14% of repositories had both pull requests and statuses.

```
# pull_requests_statuses_vs_owner_type
# status_vs_org_sql <-
#   'select * from [bonnyci-github-archive:ci_plunder_travisci.pull_requests_statuses_
vs_owner_type]'
#
# status_vs_org <- query_exec(status_vs_org_sql, project = project)
#
# saveRDS(status_vs_org, "status_vs_org.rds")
```

```
status_vs_org <- readRDS("status_vs_org.rds")

ggplot(data = status_vs_org,
       aes(x=has_statuses, fill=repo_owner_type)) +
  geom_bar(position="dodge") +
  xlab("Repos with PR's Having Statuses")
```

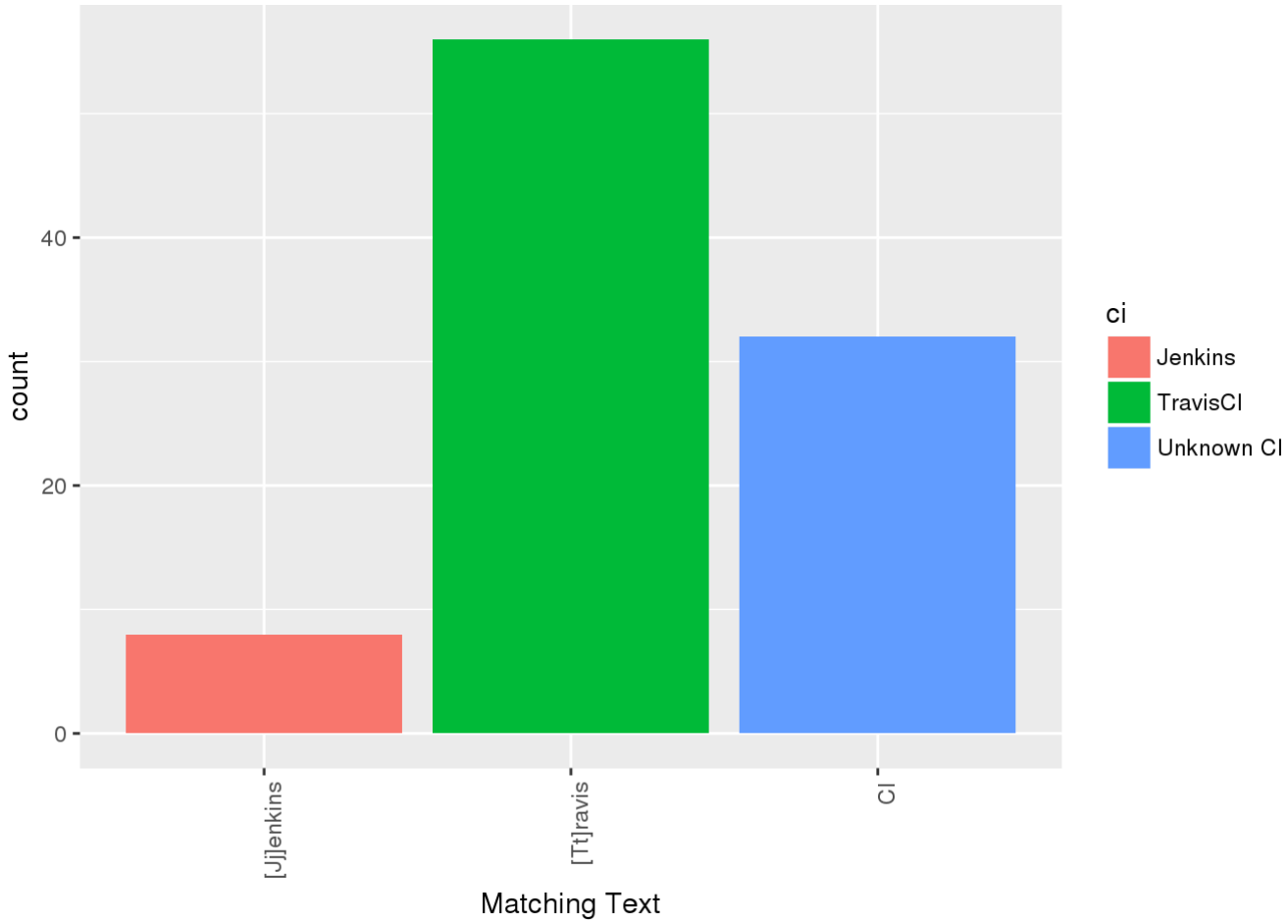


Text Search

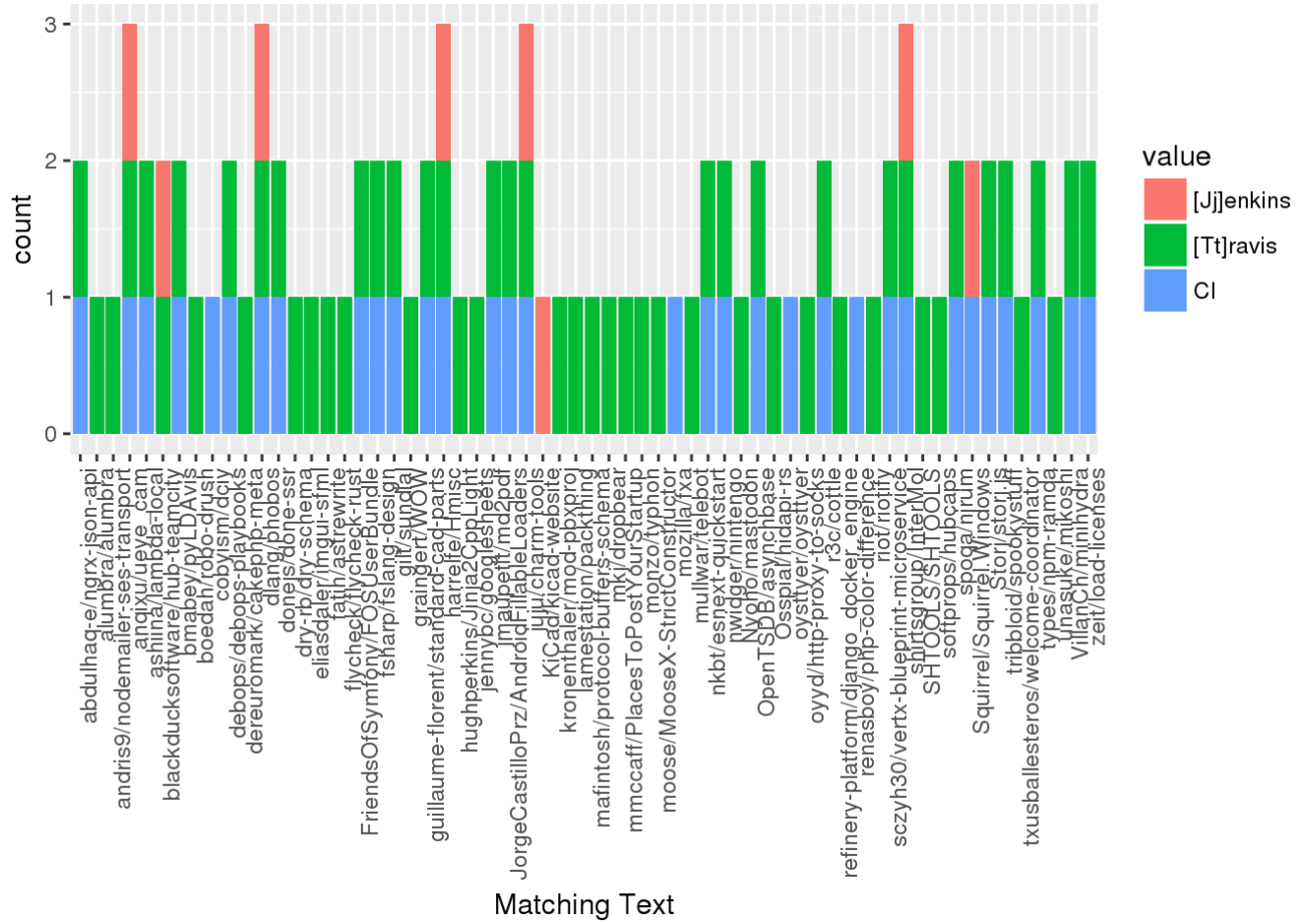
Event text searches for “CI” didn’t produce any identifications that could be identified through other methods and the ones that were not identified by other methods did not use CI. Additionally, at this stage the context of the values searched had to be manually evaluated.

```
# payload_sql <-
#   'select * from [bonnyci-github-archive:ci_plunder_travisci.payload_summary]'
#
# payload <- query_exec(payload_sql, project = project)
#
# payload <- payload %>% mutate(
#   ci = ifelse(payload == '[Tt]ravis', "TravisCI",
#               ifelse(payload == '[Jj]enkins', "Jenkins",
#                      "Unknown CI"))
# )
#
# saveRDS(payload, "payload.rds")
```

```
payload <- readRDS("payload.rds")

ggplot(data = payload,
       aes(x=payload, fill=ci)) +
  geom_bar() +
  xlab("Matching Text") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
payload_melt <- melt(payload %>% select(repo_slug, ci, payload), id=c("repo_slug", "c
i"))

ggplot(data = payload_melt,
       aes(x=repo_slug, fill=value)) +
  geom_bar() +
  xlab("Matching Text") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

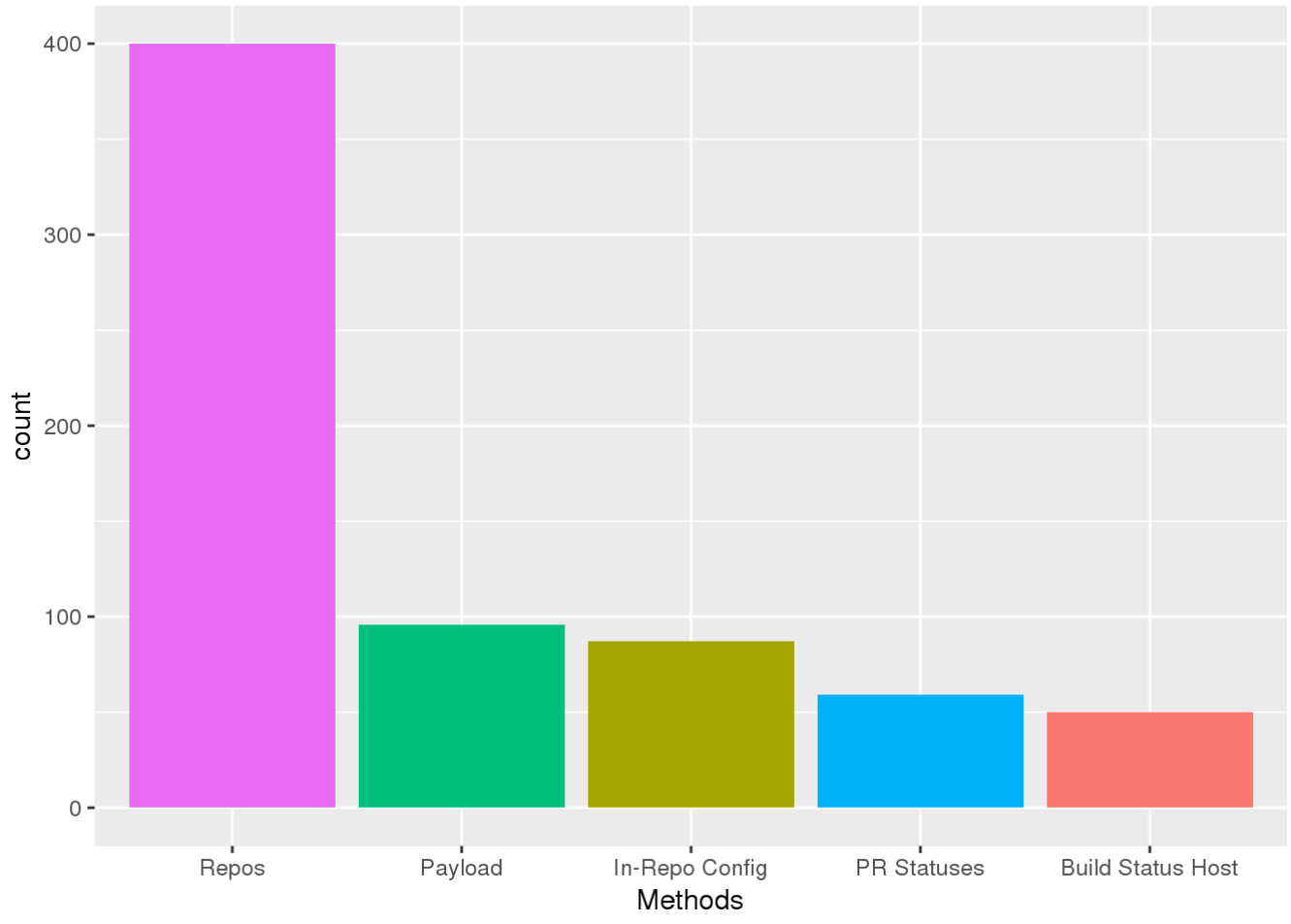


Alternatively, searches for specific CI were more successful. For TravisCI, all repos were identified this way that were also identified with the in-repo config file plus 3 extra. The remaining three did not currently use Travis but 2 had in the past and the remaining one was trying to. Overall pretty good accuracy. This means we could potentially try building a repo list based on TravisCI users from searching event text.

Comparison

How many repos overall did each method identify?

```
statuses_count <-nrow(statuses_ci %>% filter(!is.na(ci)) %>% select(repo_slug))
yml_ci_count <- nrow(yml_ci %>% filter(!is.na(ci)) %>% select(repo_slug))
build_status_host_count <- nrow(repo_readme %>% filter(!is.na(ci)) %>% select(repo_slu
g))
payload_count <- nrow(payload %>% filter(!is.na(ci)) %>% select(repo_slug))
all_repos <- 400

ci_compare <- data.frame(count=c(build_status_host_count,
                                 yml_ci_count,
                                 statuses_count,
                                 payload_count,
                                 all_repos),
                         value=c("Build Status Host",
                                 "In-Repo Config",
                                 "PR Statuses",
                                 "Payload",
                                 "Repos"))

ggplot(data = ci_compare,
       aes(x=reorder(value, -count), y=count, fill=value)) +
  geom_bar(stat="identity") +
  xlab("Methods") +
  #theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  theme(legend.position="none")
```

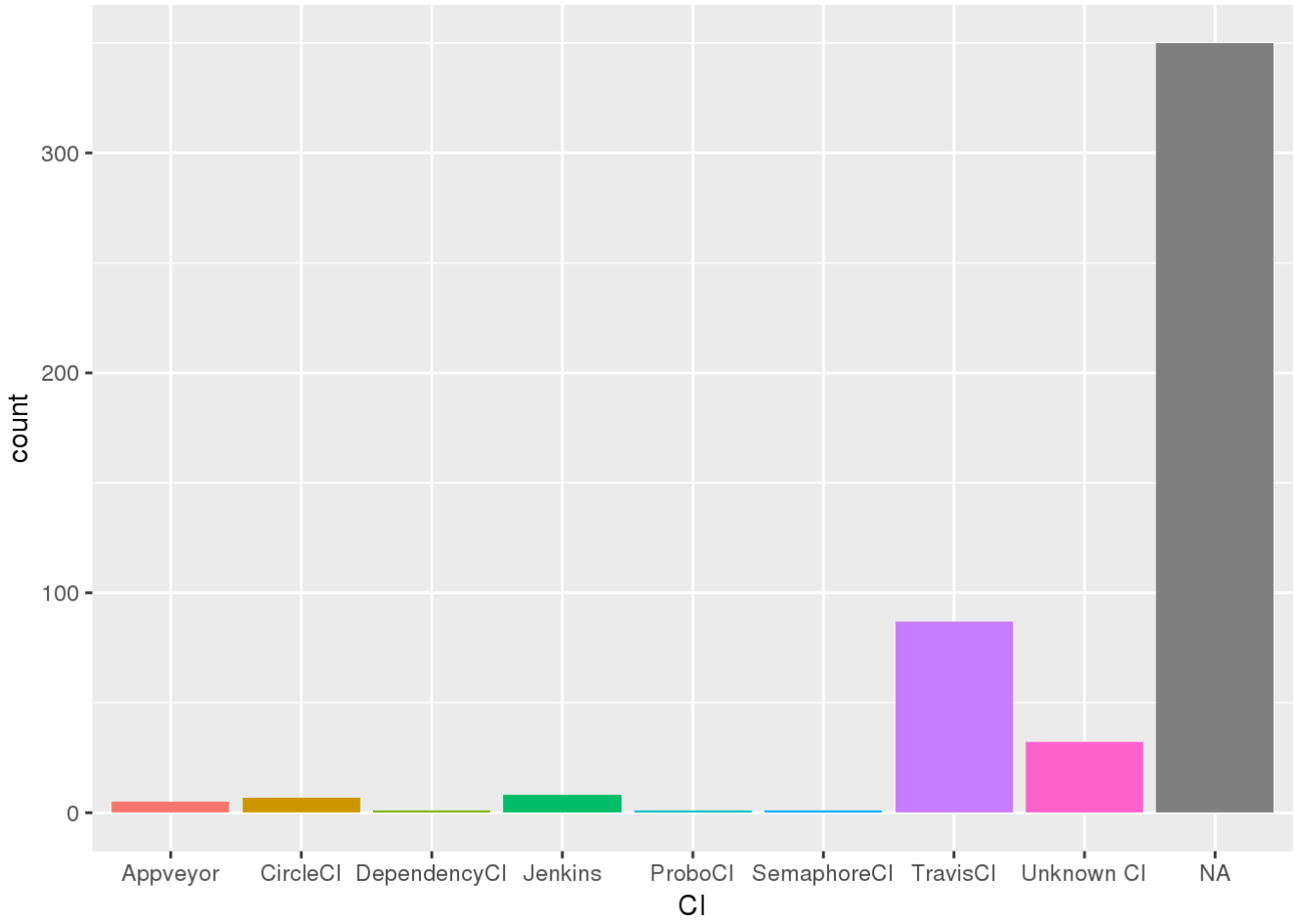


```
ggsave("ci_identification_methods.png")
```

```
## Saving 7 x 5 in image
```

What was the overlap? Did some methods identify repos that the others missed?

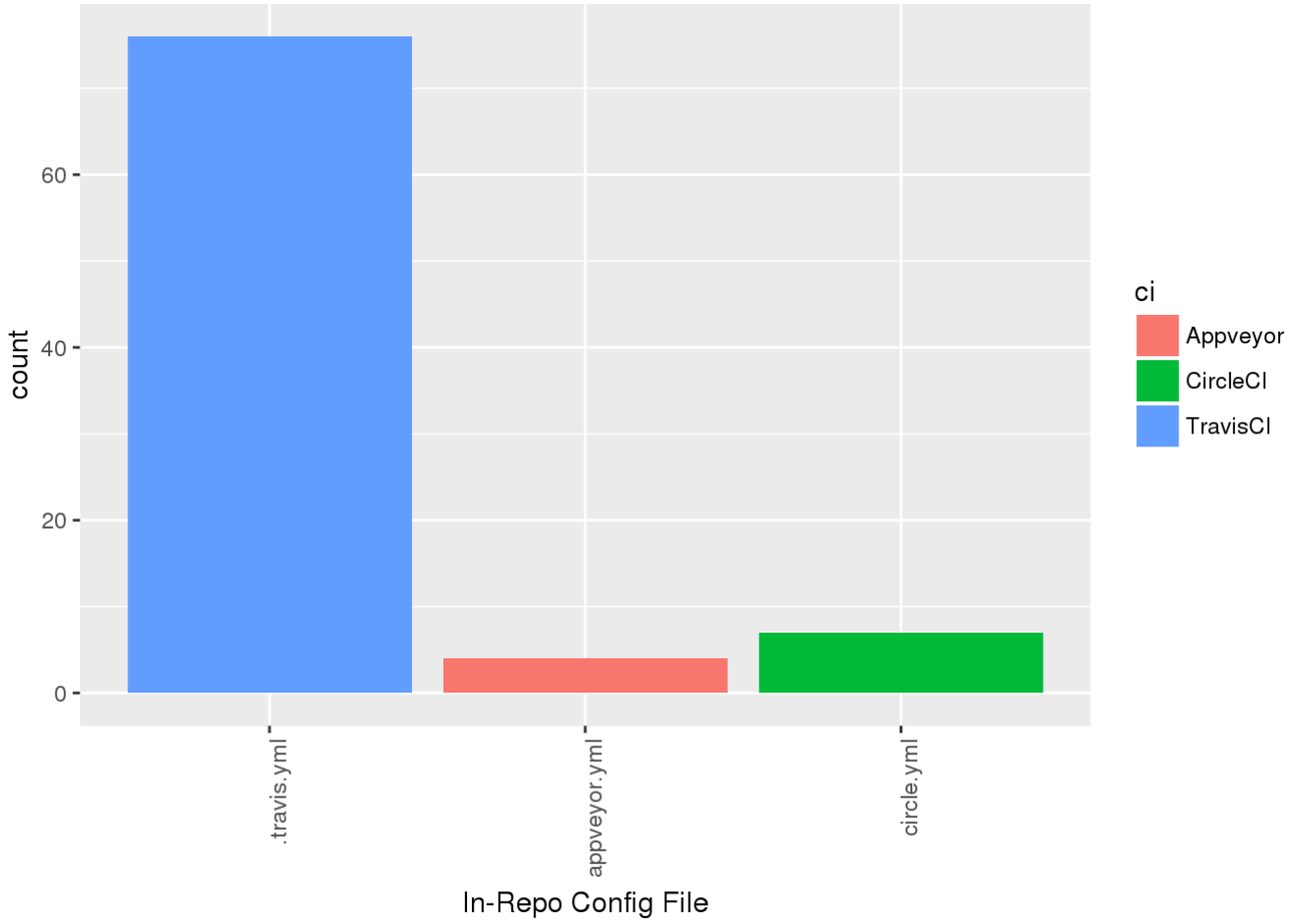
```
overall_ci <- bind_rows(yml_ci %>% select(repo_slug, name, ci),
          repo_readme %>% select(repo_slug, build_status_host, ci),
          statuses_ci %>% select(repo_slug, host, ci),
          payload %>% select(repo_slug, payload, ci))

overall_ci_melt <- melt(overall_ci, id=c("repo_slug", "ci"))

overall_ci_melt_summary <- overall_ci_melt %>%
  group_by(ci, repo_slug) %>%
  summarise(freq=n())

ggplot(data = overall_ci_melt_summary,
       aes(x=ci, fill=ci)) +
  geom_bar() +
  theme(legend.position="none") +
  xlab("CI")
```



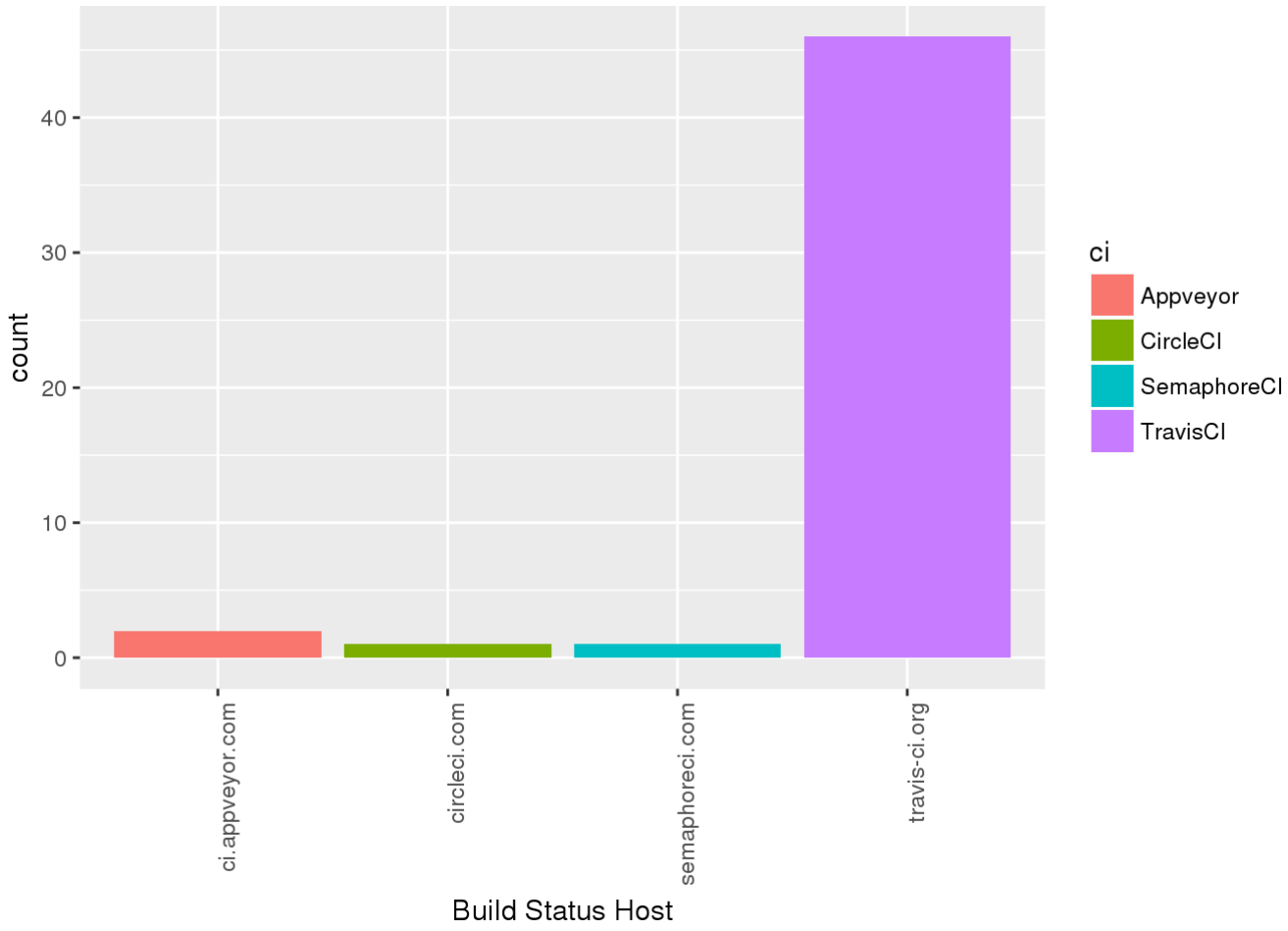

```
ggsave("ci_identification.png")
```

```
## Saving 7 x 5 in image
```

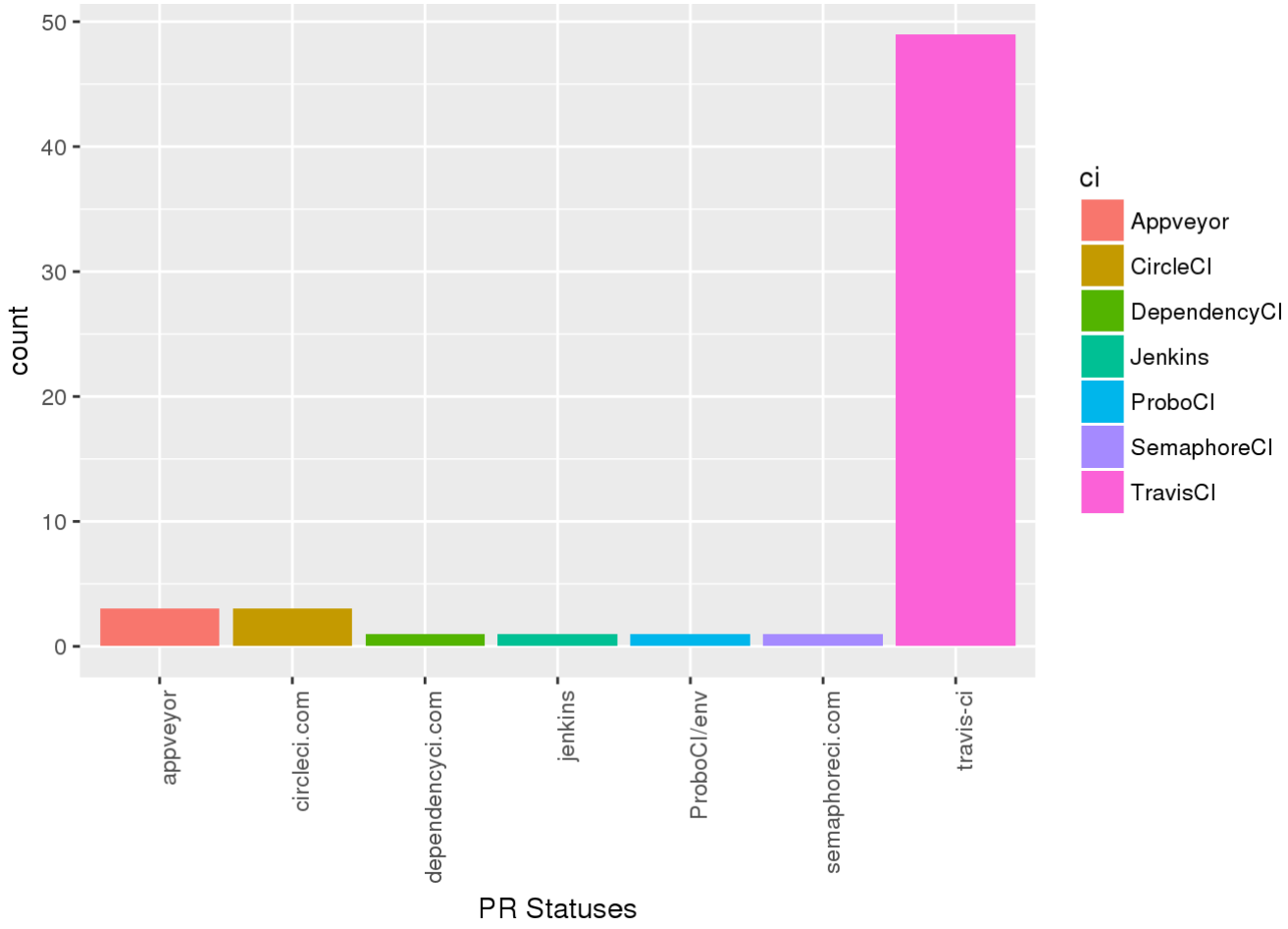
```
overall_ci_melt <- overall_ci_melt %>% filter(!is.na(value))

ggplot(data = overall_ci_melt %>% filter(variable == "name"),
       aes(x=value, fill=ci)) +
  geom_bar() +
  xlab("In-Repo Config File") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



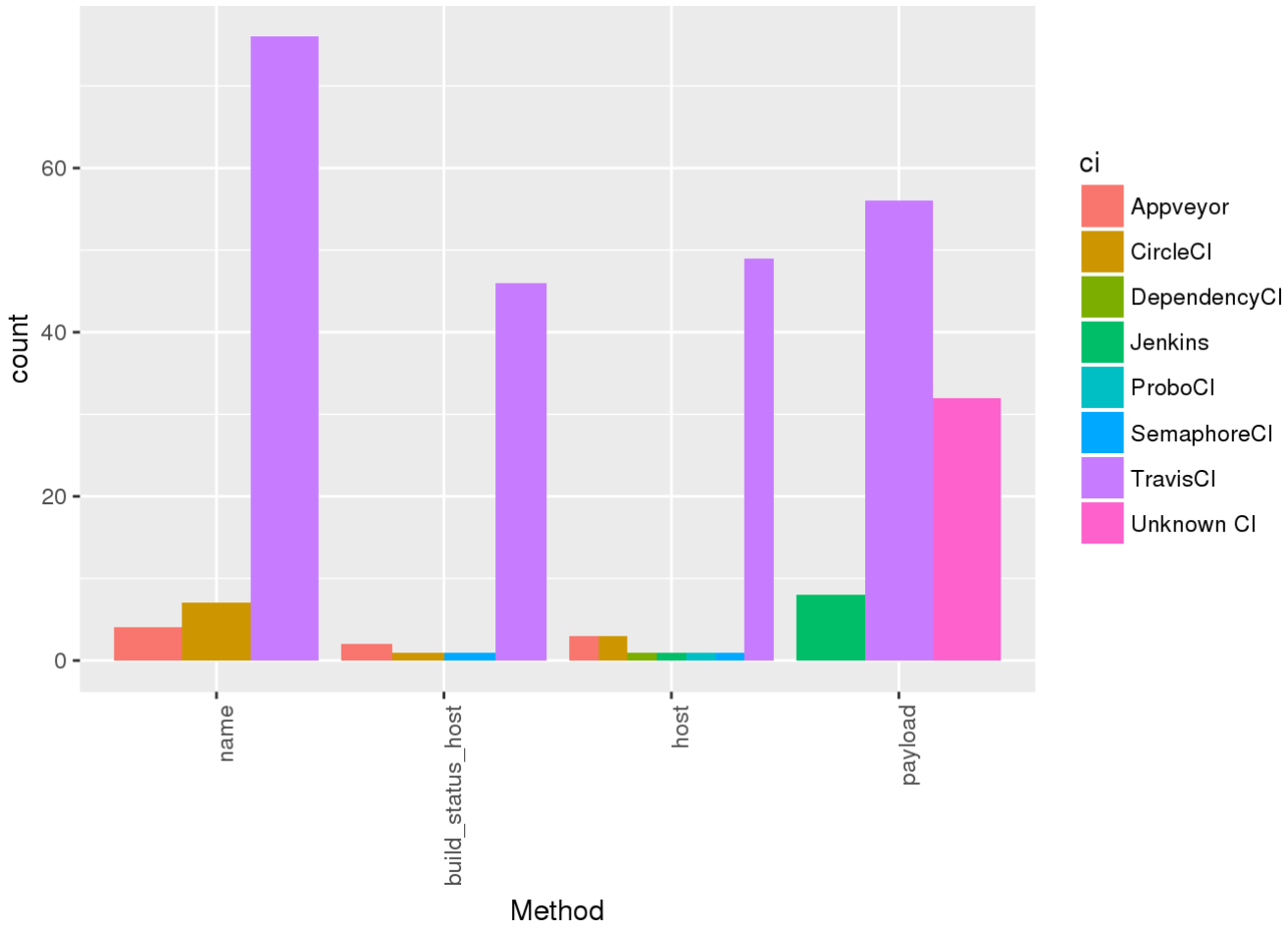

```
ggplot(data = overall_ci_melt %>% filter(variable == "build_status_host"),
       aes(x=value, fill=ci)) +
  geom_bar() +
  xlab("Build Status Host") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
ggplot(data = overall_ci_melt %>% filter(variable == "host"),
       aes(x=value, fill=ci)) +
  geom_bar() +
  xlab("PR Statuses") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
ggplot(data = overall_ci_melt,
       aes(x=variable, fill=ci)) +
  geom_bar(position="dodge") +
  xlab("Method") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```




```
ggsave("ci_identification_methods_detail.png")
```

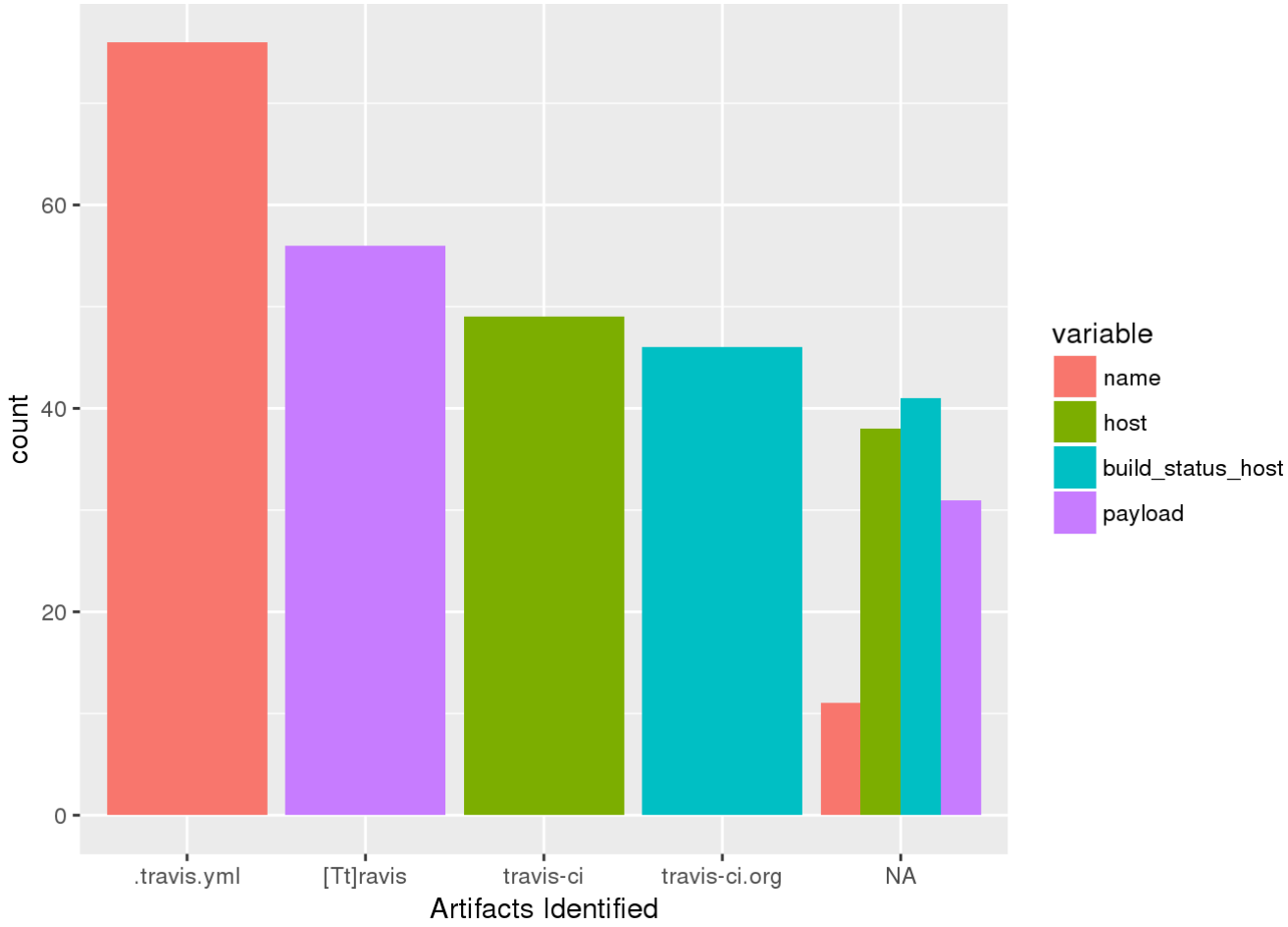
```
## Saving 7 x 5 in image
```

```
ci_methods <- overall_ci %>%
  mutate(
    has_name = ifelse(is.na(name), 0, 1),
    has_build_status_host = ifelse(is.na(build_status_host), 0, 1),
    has_host =  ifelse(is.na(host), 0, 1),
    has_payload =  ifelse(is.na(payload), 0, 1),
    is_travis =  ifelse(ci == "TravisCI", 1, 0),
    is_jenkins =  ifelse(ci == "Jenkins", 1, 0),
    is_unknown =  ifelse(ci == "Unknown CI", 1, 0)
  )

# summarize by repo
ci_methods_by_repo <- ci_methods %>%
  group_by(repo_slug, ci) %>%
  summarise(
    name = sum(has_name),
    build_status_host = sum(has_build_status_host),
    host = sum(has_host),
    payload = sum(has_payload),
    TravisCI = sum(is_travis),
    Jenkins = sum(is_jenkins),
    Unknown = sum(is_unknown)
  )
```

Which methods successfully identified TravisCI?

In-repo config identified 100% of the TravisCI repos. The ones that were not identified by this method were identified through the payload text search and that is explained further below. The other methods were split down the middle indicating that if in-repo config is not available, these methods may increase identification.

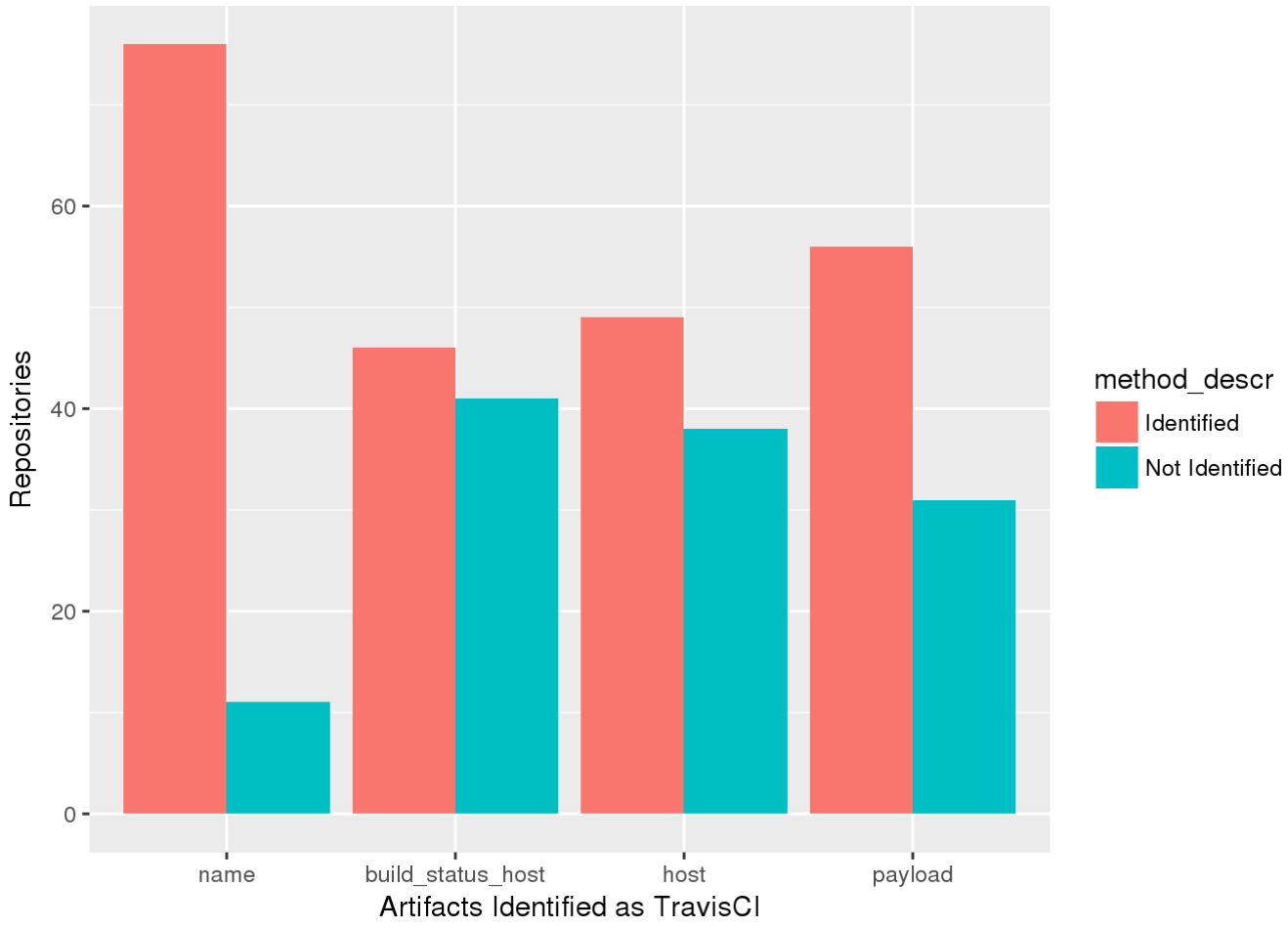
```
overall_ci_travis <- overall_ci %>% filter(ci == "TravisCI") %>% select(repo_slug, nam
e, host, build_status_host, payload)

overall_ci_travis_melt <- melt(overall_ci_travis, id="repo_slug")

overall_ci_travis_melt_summary <- overall_ci_travis_melt %>%
  arrange(repo_slug, variable, value) %>%
  group_by(repo_slug, variable) %>%
  summarise(
    value = first(value)
  )

ggplot(data = overall_ci_travis_melt_summary,
       aes(x=value, fill=variable)) +
  geom_bar(position="dodge") +
  xlab("Artifacts Identified")
```




```
ggsave("overall_travis.png")
```

```
## Saving 7 x 5 in image
```

```
ci_methods_travis <- ci_methods_by_repo %>% filter(ci == "TravisCI")

ci_methods_travis_melt <- melt(ci_methods_travis %>%
                                 select(repo_slug, name, build_status_host, host, payl
oad),
                               id="repo_slug", value.name = "method_value")

ci_methods_travis_melt <- ci_methods_travis_melt %>%
  mutate(
    method=variable,
    method_descr=ifelse(method_value==0, "Not Identified", "Identified")
         ) %>%
  select(repo_slug, method, method_value, method_descr)

ci_methods_travis_melt_by_method <- ci_methods_travis_melt %>%
  group_by(method, method_descr) %>%
  summarise(
    num_repos = n()
  )

ggplot(data = ci_methods_travis_melt_by_method,
       aes(x=reorder(method, -num_repos), y=num_repos, fill=method_descr)) +
  geom_bar(stat="identity", position="dodge") +
  xlab("Artifacts Identified as TravisCI") +
  ylab("Repositories")
```



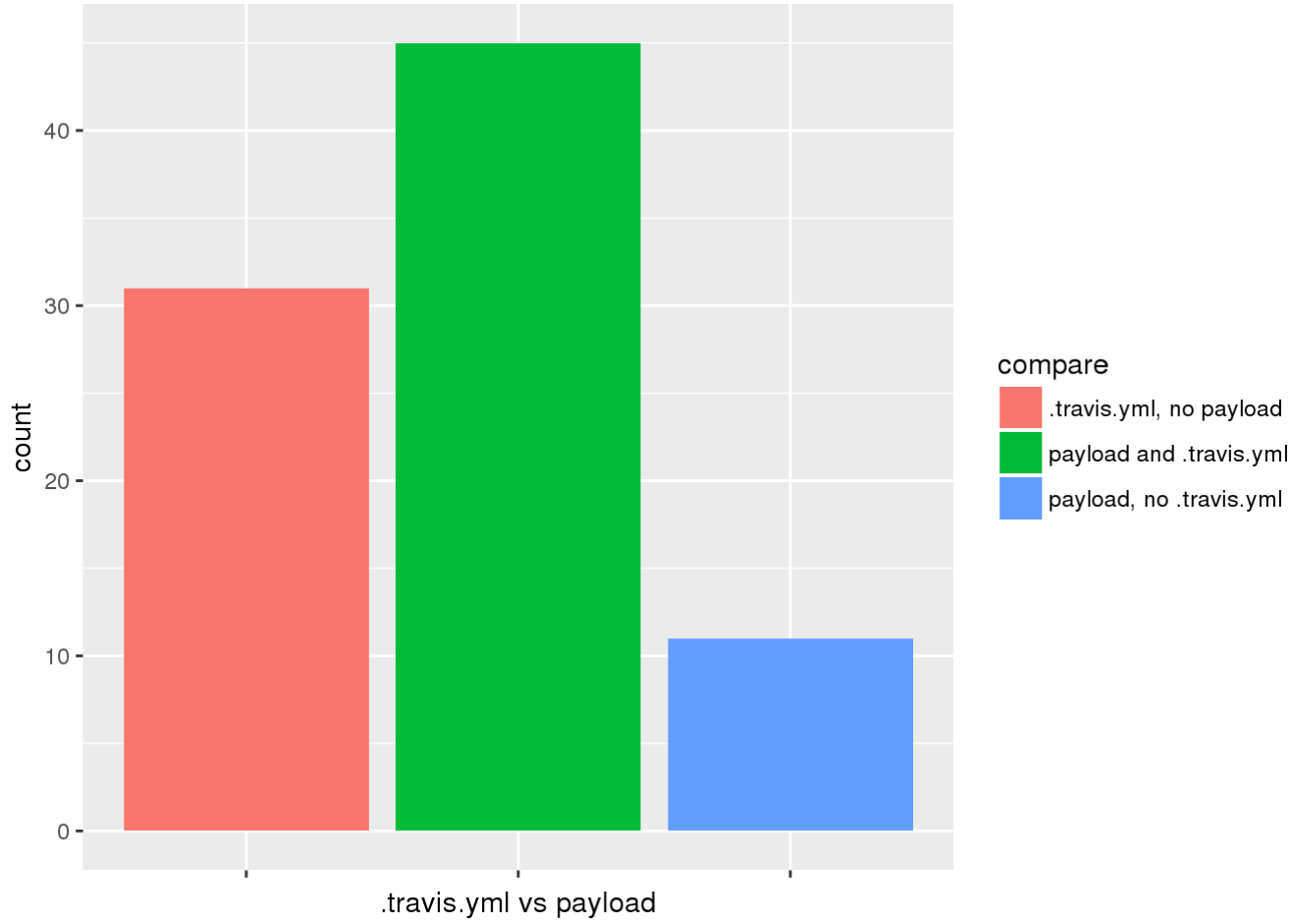
```
ggsave("overall_travis_by_method.png")
```

```
## Saving 7 x 5 in image
```

All of the repos that were not identified by the in-repo configuration method were identified through the payload text search, which suggests that none of these repos were currently using TravisCI. In-repo configuration identified 76 repositories using TravisCI. The payload text search identified 56 repositories in total and 11 of those repositories did not have an in-repo config. Additionally, the payload text search for “Travis” failed to identify 62 TravisCI repos.

The advantage to the payload search is it can be done in Google BigQuery and isn’t capped by the Github API’s rate limiting. The disadvantage is that it’s not accurate and repositories will be missed if this method is used alone. Additional analysis is needed to determine what factors contributed to failed identification. For instance, is repo age a factor? Older repos might have less CI churn which would explain why a payload wouldn’t hit specifically for those terms.

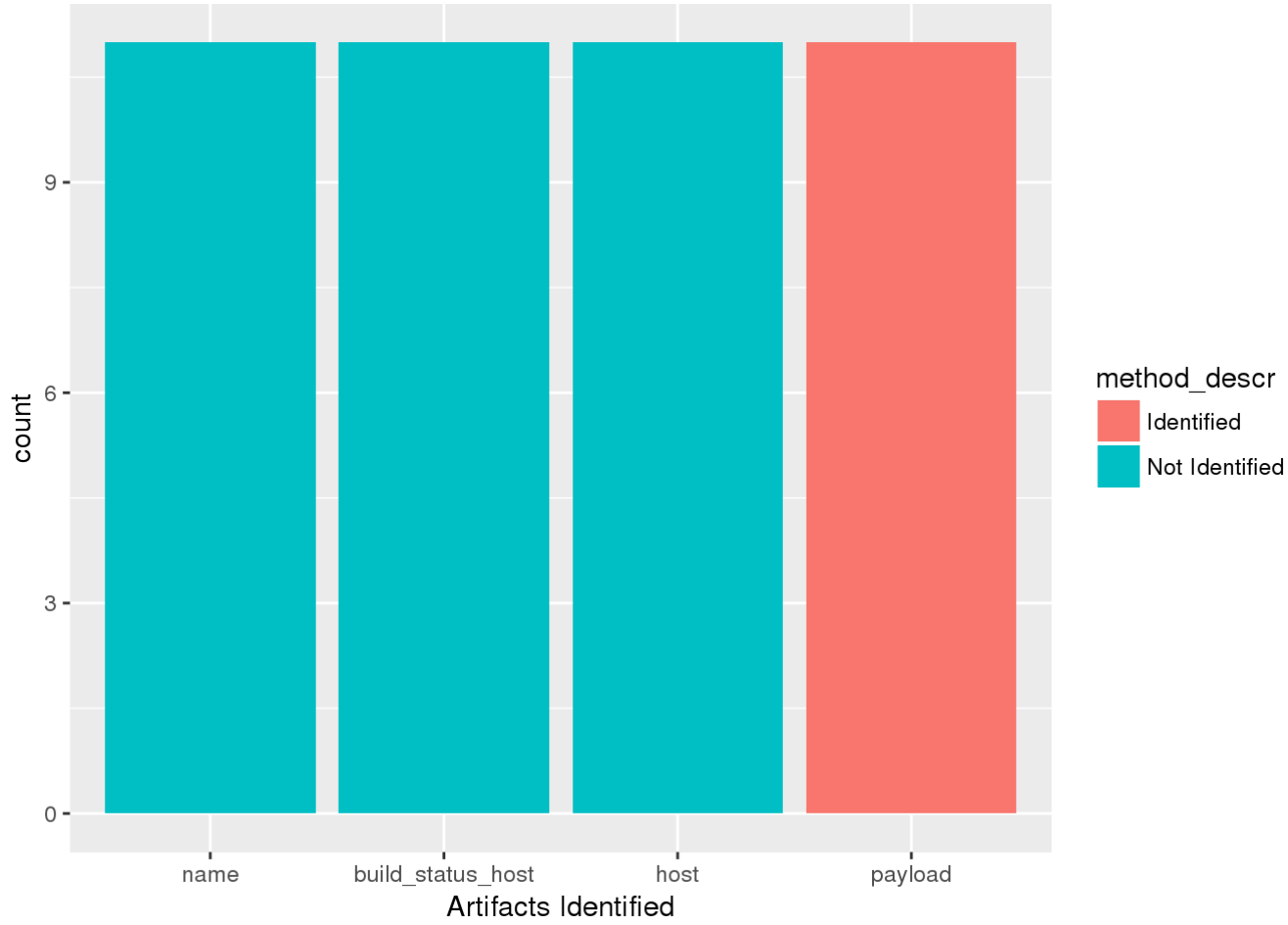
```
ci_methods_travis_compare <- ci_methods_travis %>%
  mutate(
    compare = ifelse(
      name==1 & payload == 0, ".travis.yml, no payload", ifelse(
        name==0 & payload == 1, "payload, no .travis.yml", ifelse(
          name==1 & payload == 1, "payload and .travis.yml", ifelse(
            name==0 & payload==0, "no payload or .travis.yml", NA
          )
        )
      )
    )
  )

ggplot(data = ci_methods_travis_compare,
       aes(x=compare, fill=compare)) +
  geom_bar(position="dodge") +
  xlab(".travis.yml vs payload") +
  theme(axis.text.x = element_blank())
```




```
ggsave("travis_yml_vs_payload.png")
```

```
## Saving 7 x 5 in image
```

```
travis_payload_only <- ci_methods_travis %>% filter(name==0 & payload == 1) %>%
select(repo_slug)
travis_payload_only_melt <- merge(ci_methods_travis_melt, travis_payload_only, id="rep
o_slug")

ggplot(data = travis_payload_only_melt,
       aes(x=method, fill=method_descr)) +
  geom_bar(position="dodge") +
  xlab("Artifacts Identified")
```



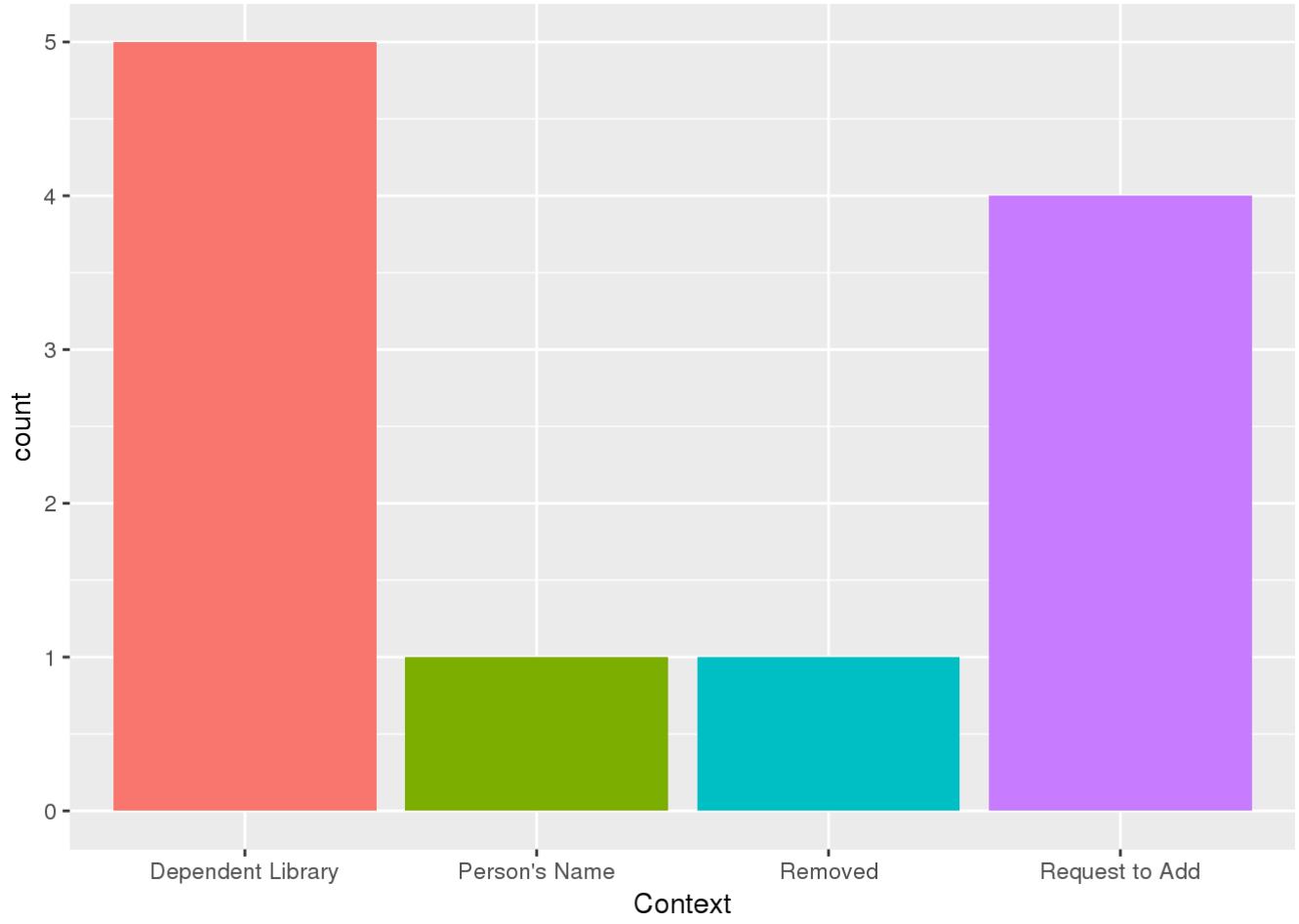

```
# repos_travis_payload_only_sql <- paste(
#   "SELECT * FROM [bonnyci-github-archive:ci_plunder_travisci.payload_travis] WHERE r
epo_slug IN('",
#     paste(as.character(travis_payload_only$repo_slug), collapse="','"),
#   "')",
#   sep="")
#
# repos_travis_payload_only <- query_exec(repos_travis_payload_only_sql, project = pro
ject)
# saveRDS(repos_travis_payload_only, "repos_travis_payload_only.rds")
```

```
repos_travis_payload_only <- readRDS("repos_travis_payload_only.rds")

travis_payload_only_event_types <- repos_travis_payload_only %>%
  group_by(repo_slug, event_type) %>%
  summarise(payload_hits=n())
```

```
repos_travis_payload_only_verification <- read.csv("repos_travis_payload_only_verifica
tion.csv")
repos_travis_payload_only_verification
```

```
##                   repo_slug uses_travisci   payload_context
## 1  abdulhaq-e/ngrx-json-api            No Dependent Library
## 2              dlang/phobos            No    Request to Add
## 3     eliasdaler/imgui-sfml            No Dependent Library
## 4    flycheck/flycheck-rust            No Dependent Library
## 5      fsharp/fslang-design            No     Person's Name
## 6              gilt/sundial            No    Request to Add
## 7            harrelfe/Hmisc            No Dependent Library
## 8          jmaupetit/md2pdf            No    Request to Add
## 9     lamestation/packthing            No    Request to Add
## 10   nkbt/esnext-quickstart            No Dependent Library
## 11    tribbloid/spookystuff            No           Removed
```

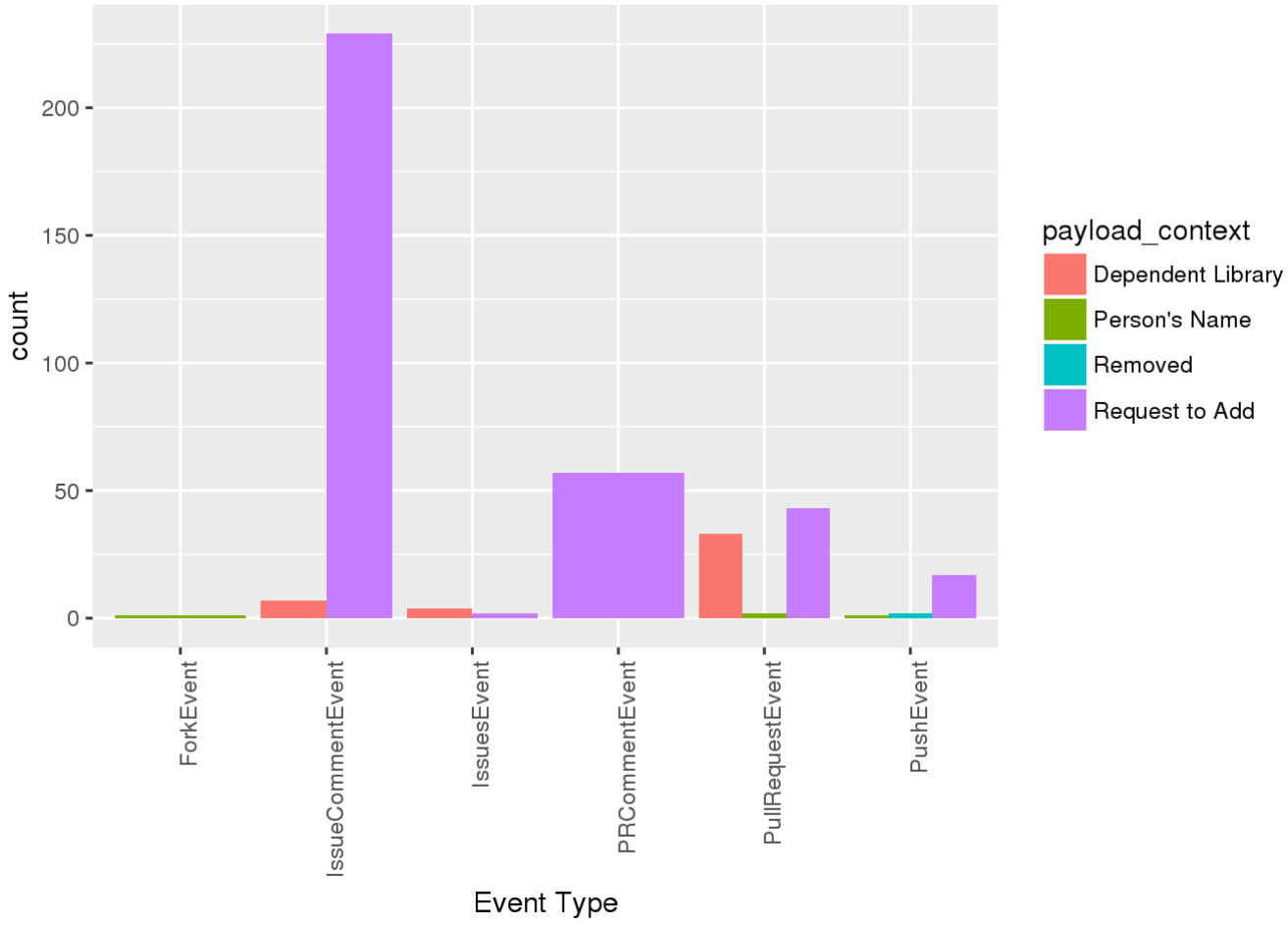
```
ggplot(data = repos_travis_payload_only_verification,
       aes(x=payload_context, fill=payload_context)) +
  geom_bar(position="dodge") +
  xlab("Context") +
  theme(legend.position="none")
```




```
ggsave("travis_payload_context.png")
```

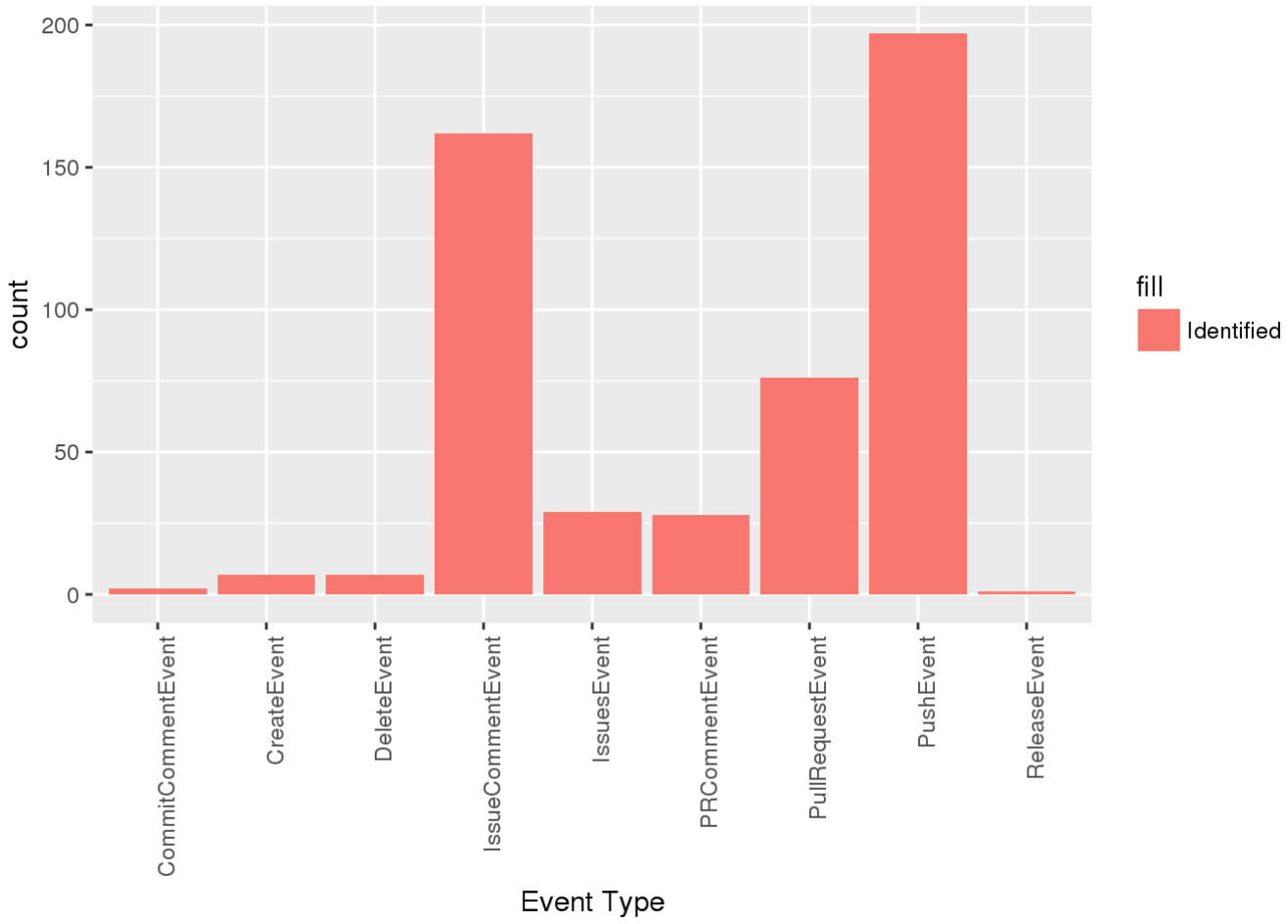
```
## Saving 7 x 5 in image
```

```
repos_travis_payload_all <- merge(repos_travis_payload_only_verification, repos_travis
_payload_only,
                                  by="repo_slug", all=TRUE)

repos_travis_payload_all <- repos_travis_payload_all %>% mutate(event_type=ifelse(even
t_type=="PullRequestReviewCommentEvent", "PRCommentEvent", event_type))

ggplot(data = repos_travis_payload_all,
       aes(x=event_type, fill=payload_context)) +
  geom_bar(position="dodge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("Event Type")
```

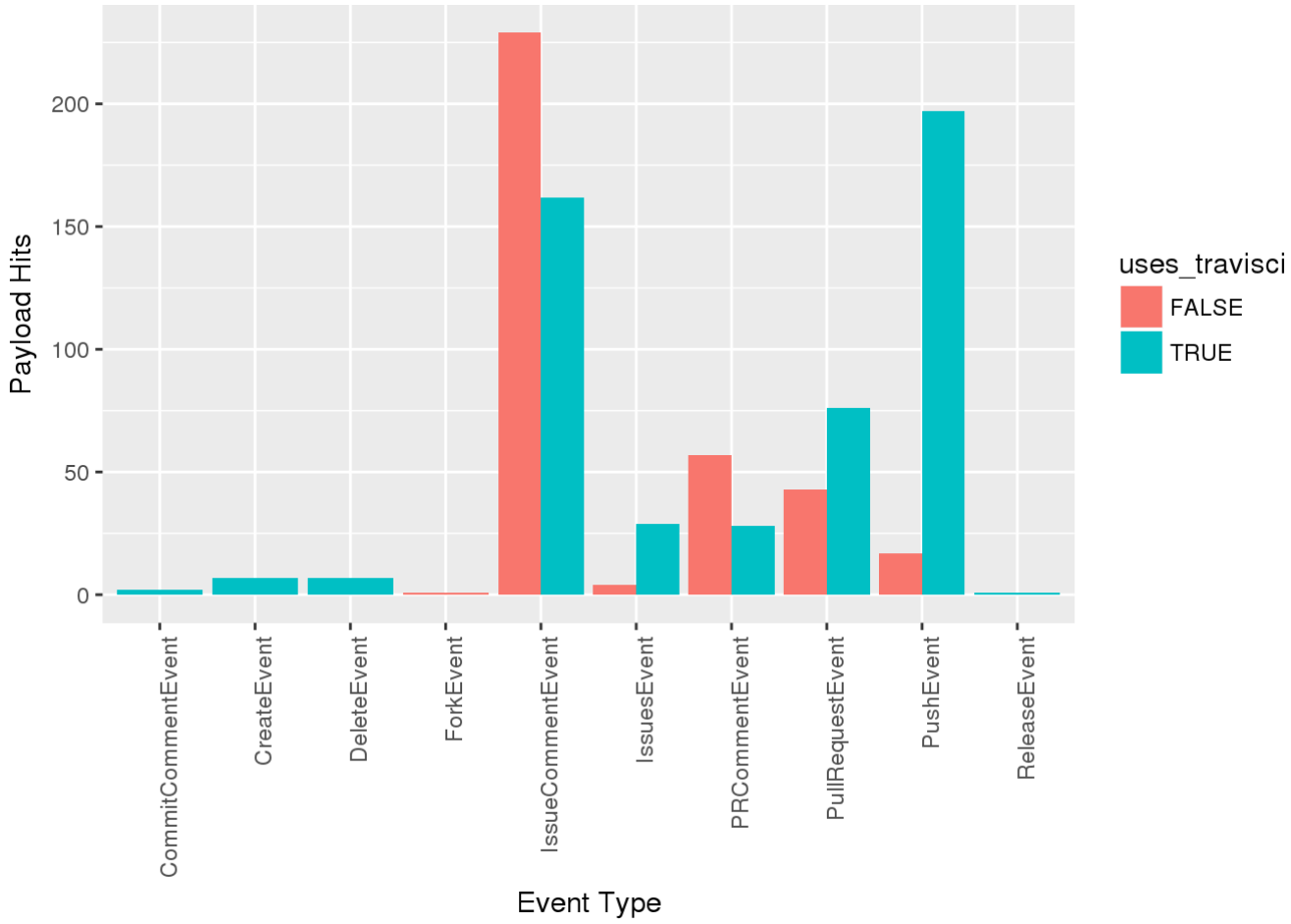


```
repos_travis_name_payload <- ci_methods_travis %>% filter(name==1 & payload == 1) %>%
select(repo_slug)

# travis_name_payload_sql <- paste(
#   "SELECT * FROM [bonnyci-github-archive:ci_plunder_travisci.payload_travis] WHERE r
epo_slug IN('",
#     paste(as.character(repos_travis_name_payload$repo_slug), collapse="','"),
#   "')",
#   sep="")
#
# travis_name_payload <- query_exec(travis_name_payload_sql, project = project)
#
# saveRDS(travis_name_payload, "travis_name_payload.rds")
```

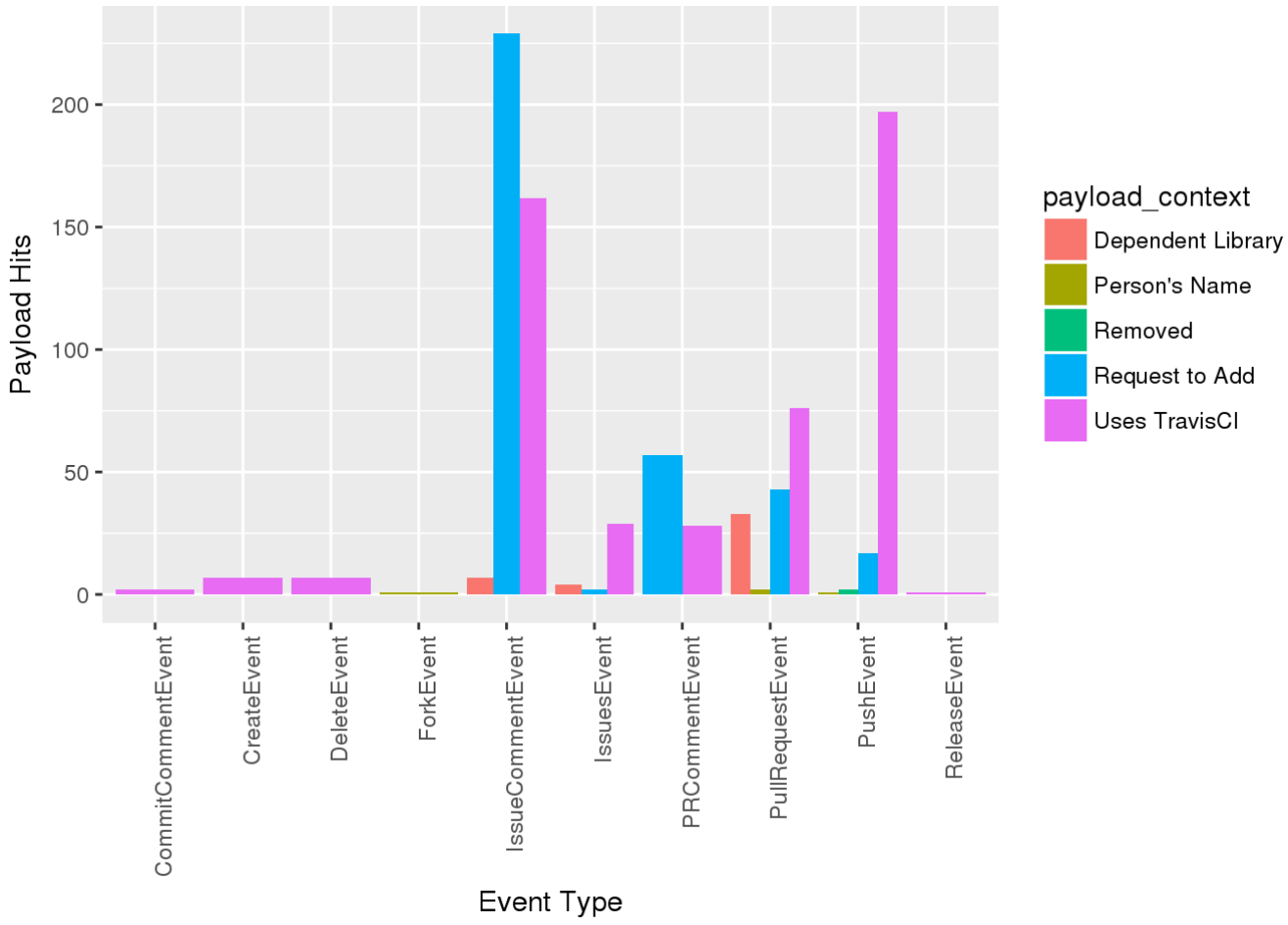
```
travis_name_payload <- readRDS("travis_name_payload.rds")
travis_name_payload <- travis_name_payload %>%
  mutate(event_type=ifelse(event_type=="PullRequestReviewCommentEvent", "PRCommentEven
t", event_type))

ggplot(data = travis_name_payload,
       aes(x=event_type, fill="Identified")) +
  geom_bar(position="dodge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("Event Type")
```



```
travis_name_payload_summary <- travis_name_payload %>% mutate(payload_context="Uses Tr
avisCI") %>%
  group_by(event_type, payload_context) %>% summarise(payload_hits = n(), uses_travisc
i=TRUE)

travis_payload_only_summary <- repos_travis_payload_all %>% group_by(event_type, paylo
ad_context) %>%
  summarise(payload_hits = n(), uses_travisci=FALSE)

travis_payload_event_types <- bind_rows(travis_name_payload_summary, travis_payload_on
ly_summary)

ggplot(data = travis_payload_event_types,
       aes(x=event_type, y=payload_hits, fill=uses_travisci)) +
  geom_bar(position="dodge", stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("Event Type") +
  ylab("Payload Hits")
```



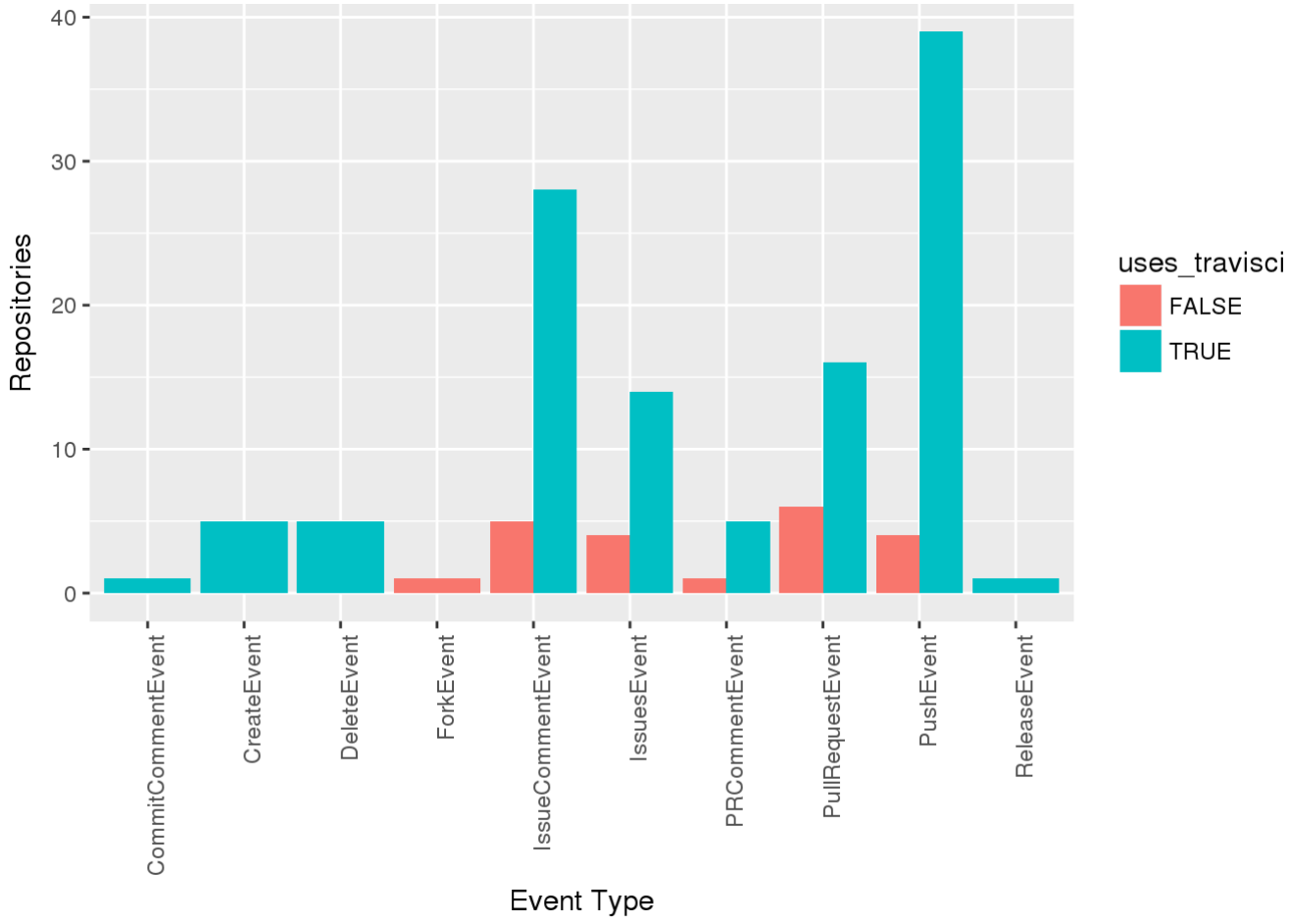

```
ggplot(data = travis_payload_event_types,
       aes(x=event_type, y=payload_hits, fill=payload_context)) +
  geom_bar(position="dodge", stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("Event Type") +
  ylab("Payload Hits")
```



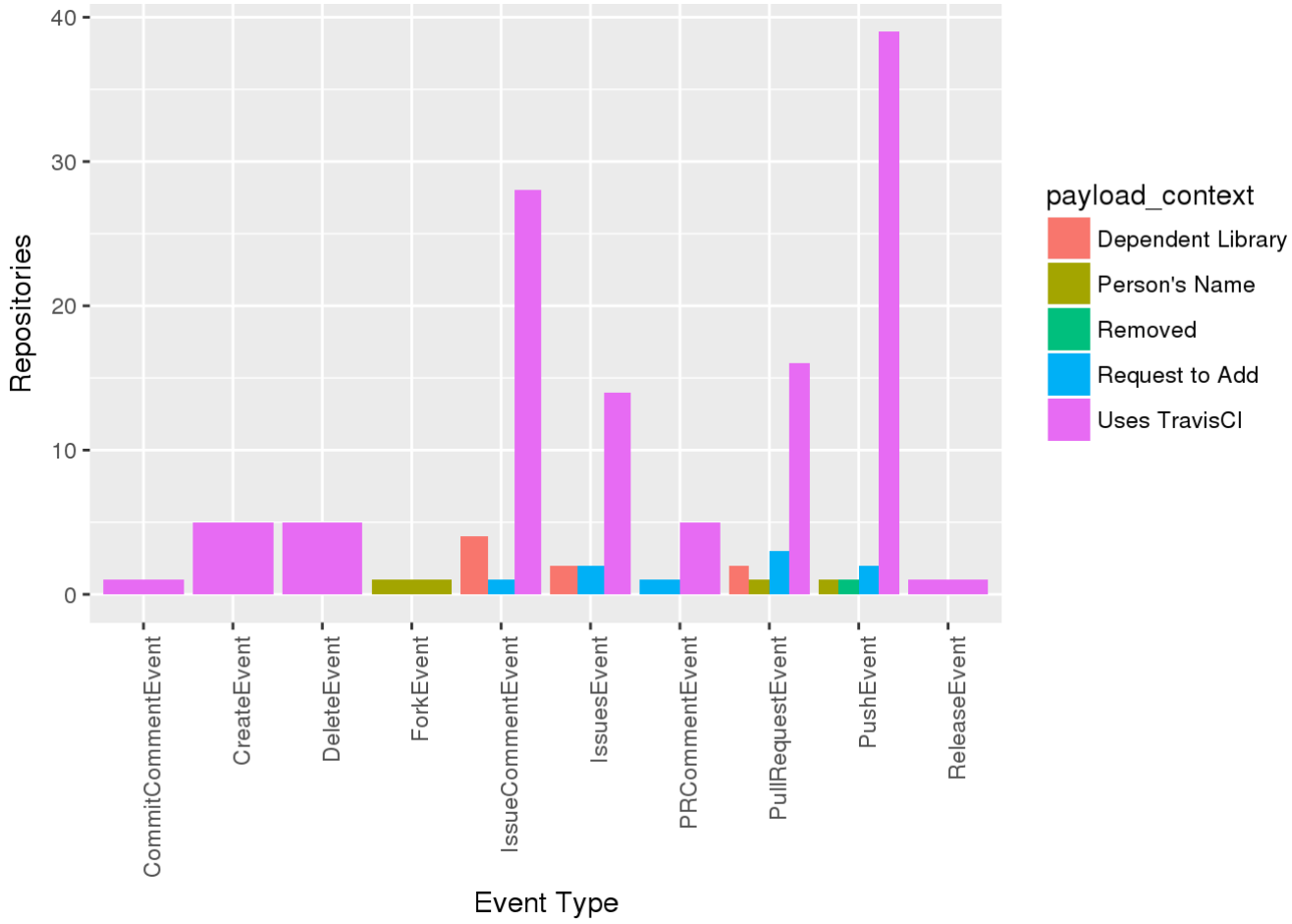

```
ggsave("travis_payload_context_num_hits.png")
```

```
## Saving 7 x 5 in image
```

```
travis_name_payload_summary_repo <- travis_name_payload %>% mutate(payload_context="Us
es TravisCI") %>%
  group_by(event_type, payload_context, repo_slug) %>% summarise(payload_hits = n(), u
ses_travisci=TRUE)

travis_payload_only_summary_repo <- repos_travis_payload_all %>%
  group_by(event_type, payload_context, repo_slug) %>%
  summarise(payload_hits = n(), uses_travisci=FALSE)

travis_payload_event_types_repo <- bind_rows(travis_name_payload_summary_repo,
                                             travis_payload_only_summary_repo)

ggplot(data = travis_payload_event_types_repo,
       aes(x=event_type, fill=uses_travisci)) +
  geom_bar(position="dodge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("Event Type") +
  ylab("Repositories")
```



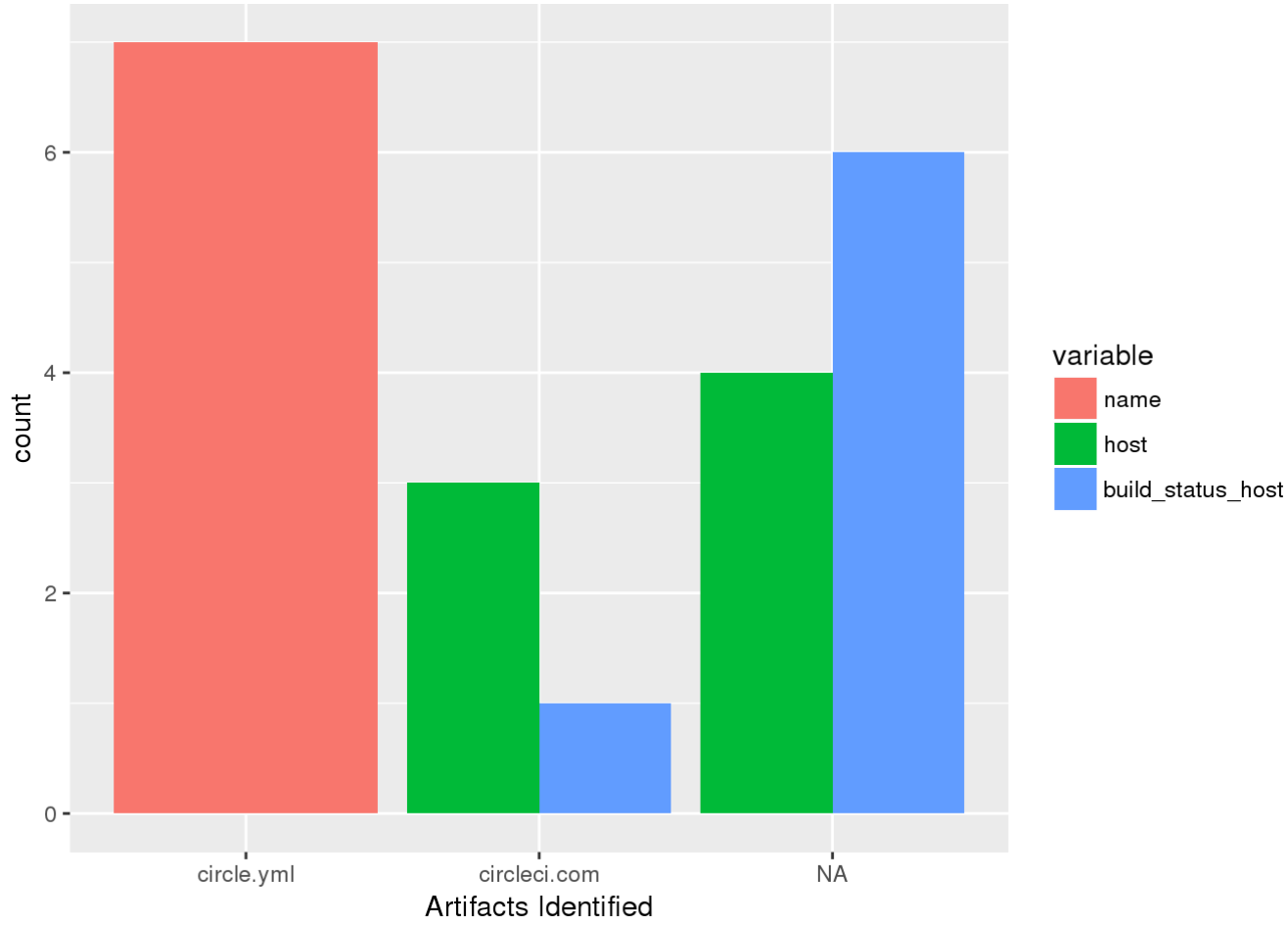

```
ggplot(data = travis_payload_event_types_repo,
       aes(x=event_type, fill=payload_context)) +
  geom_bar(position="dodge") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  xlab("Event Type") +
  ylab("Repositories")
```




```
ggsave("travis_payload_context_num_repos.png")
```

```
## Saving 7 x 5 in image
```

```
overall_ci_circle <- overall_ci %>% filter(ci == "CircleCI") %>% select(repo_slug, nam
e, host, build_status_host)

overall_ci_circle_melt <- melt(overall_ci_circle, id="repo_slug")
overall_ci_circle_summary <- overall_ci_circle_melt %>%
  arrange(repo_slug, variable, value) %>%
  group_by(repo_slug, variable) %>%
  summarise(
    value = first(value)
  )


ggplot(data = overall_ci_circle_summary,
       aes(x=value, fill=variable)) +
  geom_bar(position="dodge") +
  xlab("Artifacts Identified")
```



Conclusions

For repos that use in-repo configuration, the best way to identify them is to use the Github api to look for those files in a sample. To identify what these CI might be, checking the hosts in the pull request statuses gleaned the best results. The overhead for this is comprable to the build status host in readme as both require a single API call and the use of text parsing to extract the host information.

Finally, payload text search is an option to consider for CI that do not use in-repo configuration, however the overhead is potentially very high as these need to be cross-checked against other identification methods and the accuracy is poor. Payload text search, however, has the potential to indicate past CI usage or a desire for future CI usuage. Additionally, there is an opportunity for applying Machine Learning or NLP techniques to discover patterns that could indicate whether a repo is likely to be using CI or not.

TravisCI is the most popular CI, easy to identify accurately, and many repos using it also use other CI. Therefore future analysis for purposes of this research will focus on TravisCI. These identification methods may be useful down the road to check the TravisCI analysis results.