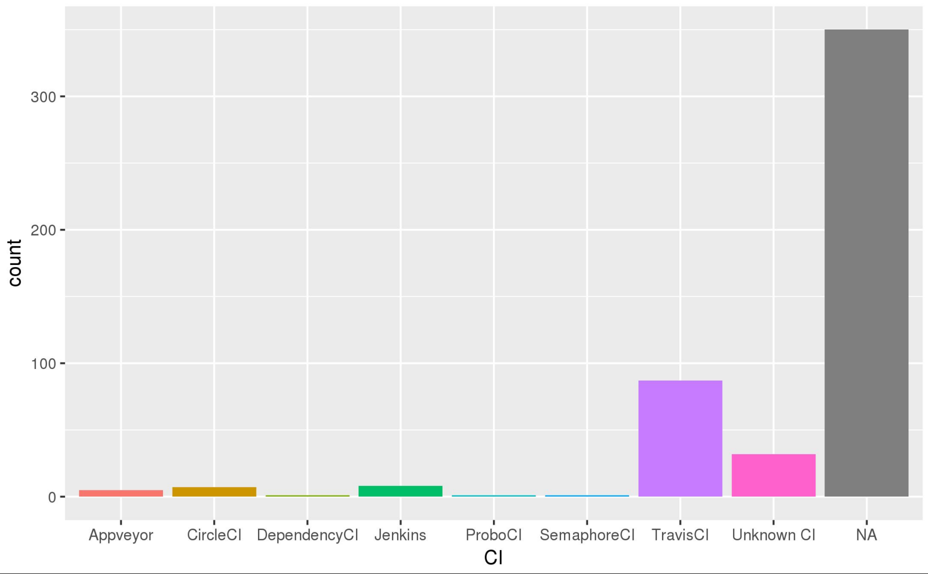
Identifying CI in Github

CI Identification Results

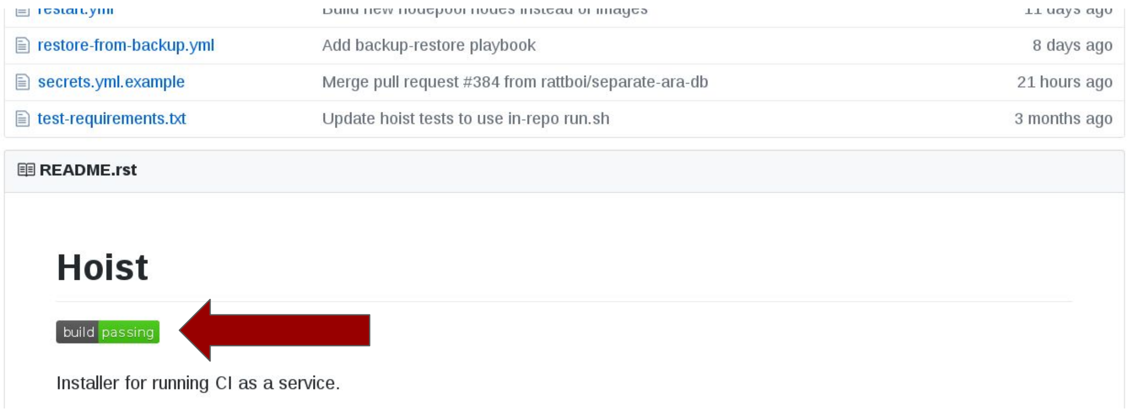


Overall, the majority of repositories in the sample were not identified as using CI. Of the ones that were identified, TravisCI had the highest hits.

Methods

- Extract Host from Build Status Tag in Readme
- Extract Host from Pull Request Statuses
- Text Search in Github Archive Events Payload
- Fetch Known In-Repo Configuration Files via Github API

Build Status Host


restart.yml
Build new nodepool nodes instead of images
11 days ago
restore-from-backup.yml
Add backup-restore playbook
8 days ago
secrets.yml.example
Merge pull request #384 from rattboi/separate-ara-db
21 hours ago
test-requirements.txt
Update hoist tests to use in-repo run.sh
3 months ago
README.rst
Hoist
build passing
Installer for running CI as a service.

```
[![Build Status](https://travis-ci.org/foo/bar.svg?branch=master)]

[![Build Status](https://jenkins.foo.com/icon?job=master)]
```

Extract the hostname from the Build Status tags in the repo's Readme. This was done by requesting the README contents via the Github API and then using a Regex. Note that for this analysis, we only extracted the first matching result, so that may impact the final match frequency as compared to other methods that may have matched multiple CI's per repo.

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.

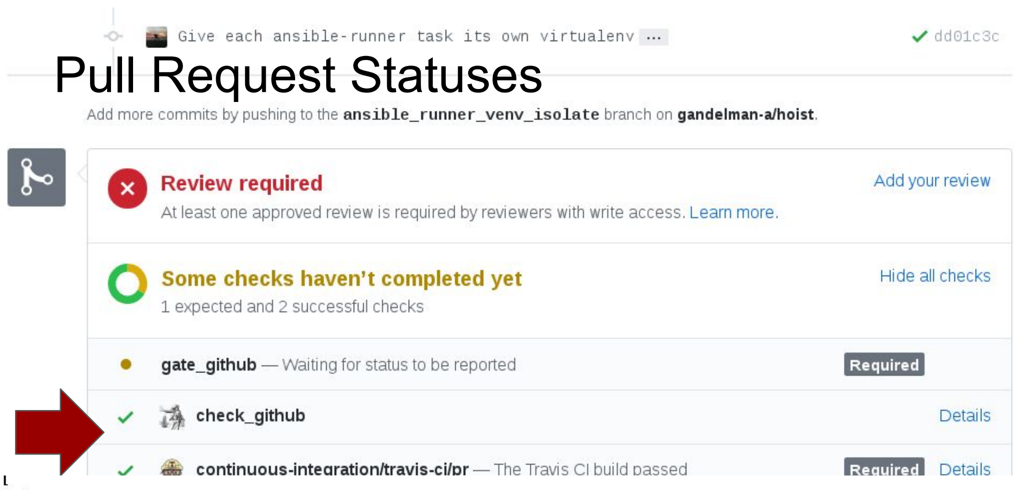
Good:

- Discovery: find out what CI systems in use without any further knowledge

Bad:

- Accuracy: subject to human error in editing, depends on Readme information being current/accurate
- Breadth: depends on Readme existing, didn't result in a huge number of id's

Pull Request Statuses



Give each ansible-runner task its own virtualenv … ✓ dd01c3c

Add more commits by pushing to the **ansible_runner_venv_isolate** branch on **gandelman-a/hoist**.

Review required — Add your review

At least one approved review is required by reviewers with write access. Learn more.

Some checks haven't completed yet — Hide all checks

1 expected and 2 successful checks

● **gate_github** — Waiting for status to be reported — Required

✓ **check_github** — Details

✓ **continuous-integration/travis-ci/pr** — The Travis CI build passed — Required — Details

```
[
  {
    "url": "https://api.github.com/repos/BonnyCI/hoist/statuses/dd01c3cafafb93f58691c4a991649a5b7ca0e5c2",
    "id": 1210950031,
    "state": "success",
    "description": "The Travis CI build passed",
    "target_url": "https://travis-ci.org/BonnyCI/hoist/builds/230476007?utm_source=github_status&utm_medium=notification",
    "context": "continuous-integration/travis-ci/pr",
    "created_at": "2017-05-09T19:10:43Z",
    "updated_at": "2017-05-09T19:10:43Z",
    "creator": {
```

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. Status messages provide host information in the data fields. Similar to the build status tag approach, the hosts have been extracted from these fields.

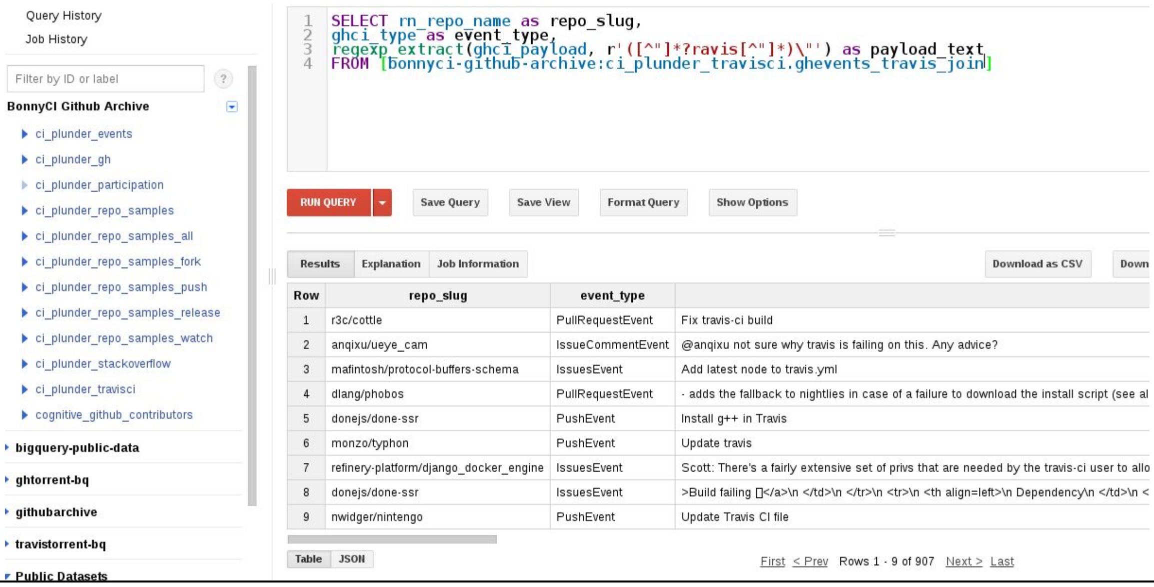
Good:

- Accurate - the repo was clearly configured at some point to talk to the CI
- Discovery - shows interactions with external systems that may not have artifacts within the repo

Bad:

- depends on Read permissions (some orgs don't allow "pull" access)
- depends on overall community/project workflow
- Depends on whether Github is configured to integrate with their external CI (~38% of repos in sample had PR's, 14% had both PR's and statuses)

Event Text Searches



```sql
SELECT rn_repo_name as repo_slug,
ghci_type as event_type,
regexp_extract(ghci_payload, r'([^"]*?ravis[^"]*)\"') as payload_text
FROM [bonnyci-github-archive:ci_plunder_travisci.ghevents_travis_join]
```

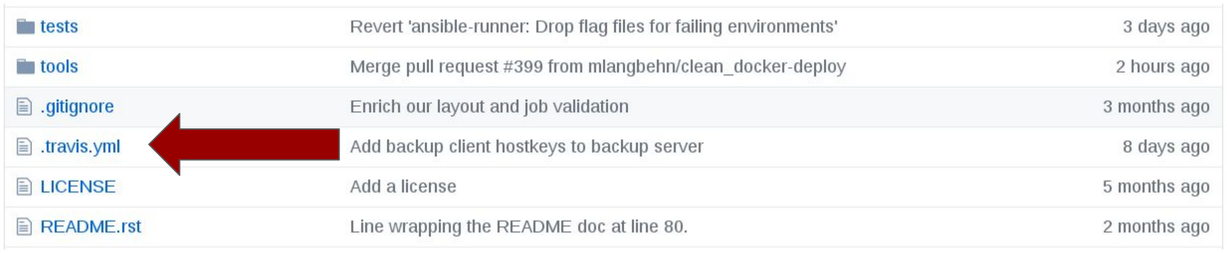
Row	repo_slug	event_type	
1	r3c/cottle	PullRequestEvent	Fix travis-ci build
2	anqixu/ueye_cam	IssueCommentEvent	@anqixu not sure why travis is failing on this. Any advice?
3	mafintosh/protocol-buffers-schema	IssuesEvent	Add latest node to travis.yml
4	dlang/phobos	PullRequestEvent	- adds the fallback to nightlies in case of a failure to download the install script (see al
5	donejs/done-ssr	PushEvent	Install g++ in Travis
6	monzo/typhon	PushEvent	Update travis
7	refinery-platform/django_docker_engine	IssuesEvent	Scott: There's a fairly extensive set of privs that are needed by the travis-ci user to allo
8	donejs/done-ssr	IssuesEvent	>Build failing □</a>\n </td>\n </tr>\n <tr>\n <th align=left>\n Dependency\n </td>\n <
9	nwidger/nintengo	PushEvent	Update Travis CI file

Rows 1 - 9 of 907

A simple text search was used on the Github Events Archive data to look for mentions of CI, Travis, or Jenkins. Ultimately this might be a better Machine Learning task but it was interesting to see if it was even worth pursuing.

Good: Broad discovery, especially for the “CI” text search. Pretty good number of hits. Can do it on the large GBQ Github Events dataset with little overhead.
Bad: Not accurate, results need to be verified with another method or manually verified.

In-Repo Configuration Files



tests	Revert 'ansible-runner: Drop flag files for failing environments'	3 days ago
tools	Merge pull request #399 from mlangbehn/clean_docker-deploy	2 hours ago
.gitignore	Enrich our layout and job validation	3 months ago
.travis.yml	Add backup client hostkeys to backup server	8 days ago
LICENSE	Add a license	5 months ago
README.rst	Line wrapping the README doc at line 80.	2 months ago

https://api.github.com

```
repos/{owner}/{repo}/contents/.travis.yml
```

Searching the repository contents for a specific configuration file, retrieved from the Github API

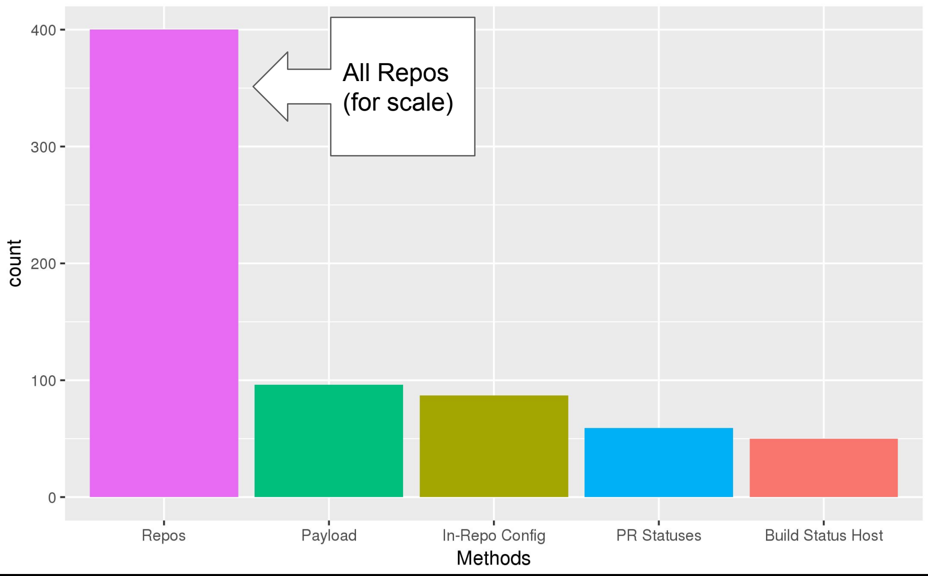
Good:

- Accurate: if the file exists, strong possibility the repo is/was/will use the CI
- Frequency: Highest number of repos identified

Problems:

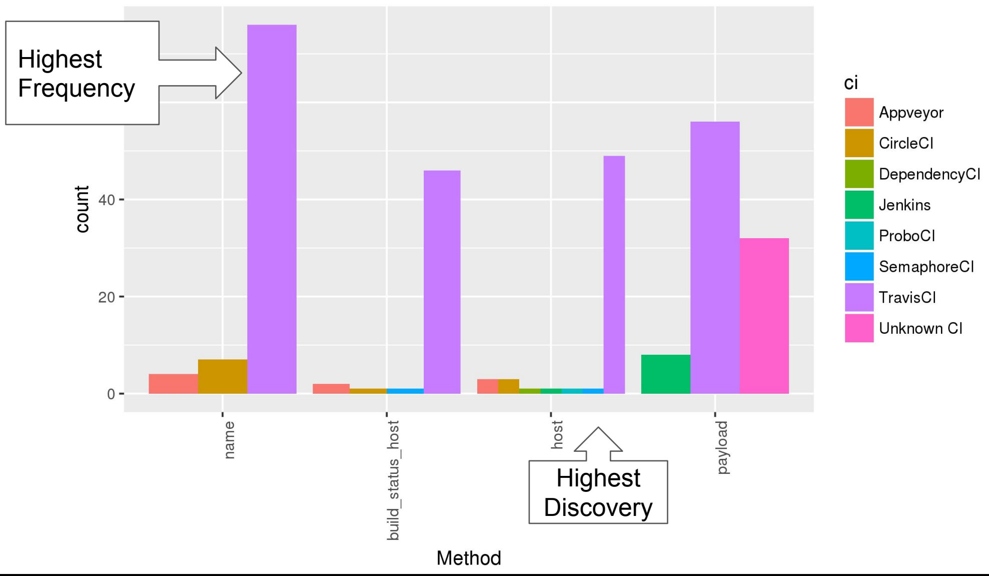
- CI must use in-repo configuration
- filename must be consistent and easily searchable across many repos
- file must be in the same place in each repository

CI Identified per Method (Overall)



Payload looks like it got more hits than In-Repo config, but these were a) not accurate and b) the actual CI being used wasn’t necessarily identifiable.

CI identified per Method (detailed)



Unknown CI refers to a text match for “CI”.

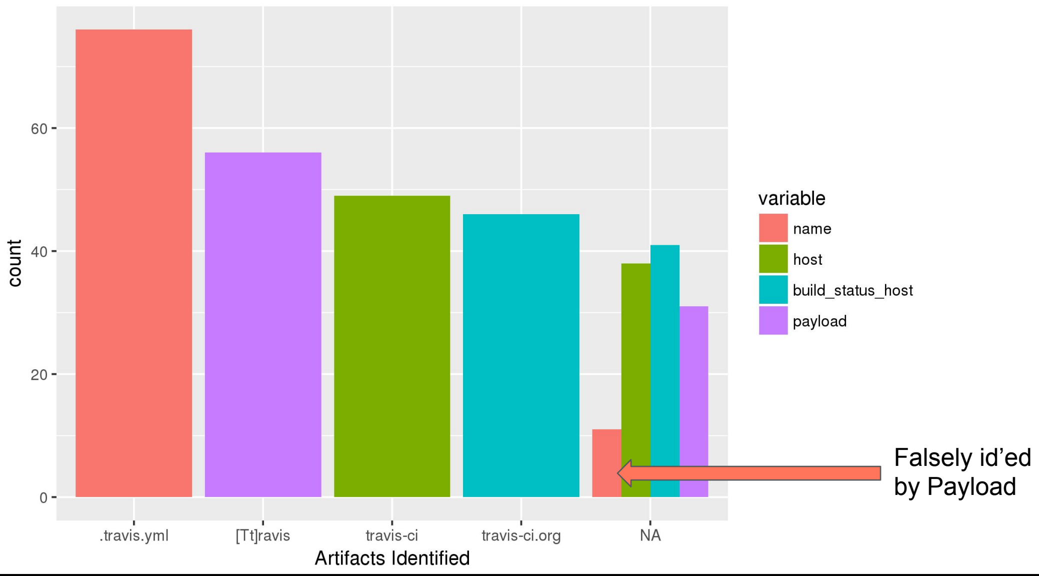
CI Identification Method Comparison

	Discovery	Frequency	Accuracy
Build Status Host	☀	❄	❄
PR Status	☀	❄	☀
Payload Text Search	☀	☀	❄
In Repo Config	❄	☀	☀

Comparing Travis CI Identification

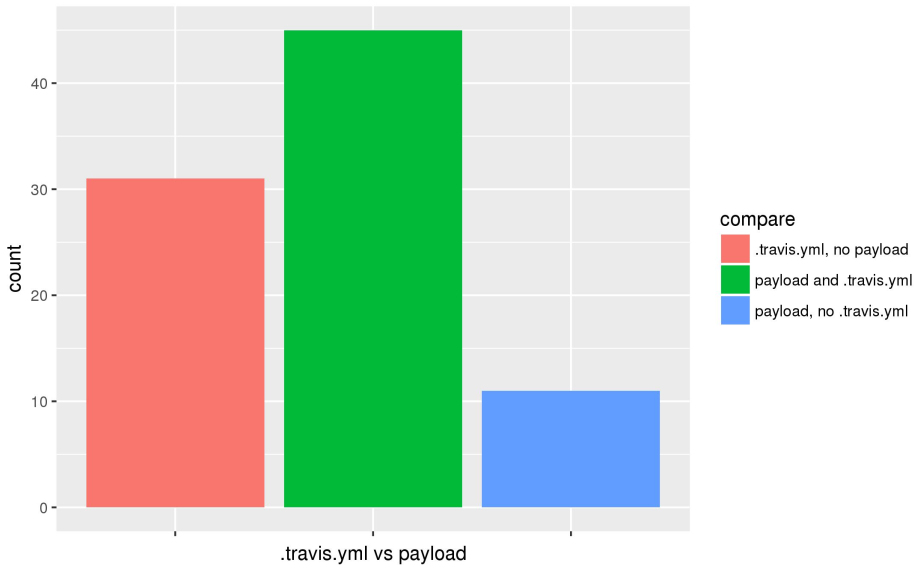
- TravisCI is the most popular CI
- Uses In-repo config (easy to check for accuracy)

TravisCI Identification



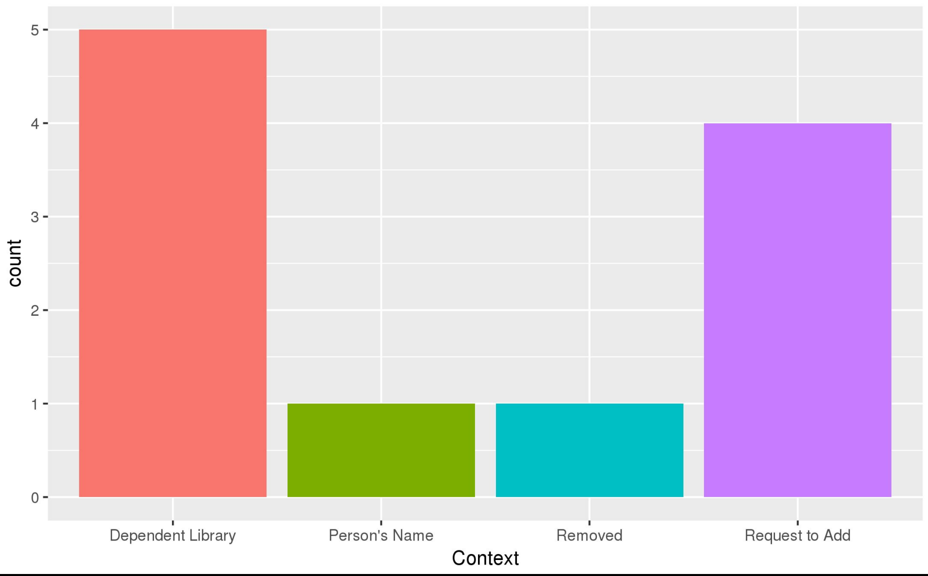
For each repo identified as using TravisCI, this shows what methods successfully identified each. If an artifact was identified, the value field contains the name of the artifact, otherwise it contains NA. The “NA” in the chart indicates repos that did not have the artifact that would identify them by the method indicated by the fill color. All of the TravisCI repos were identified by the in-repo configuration.

In-Repo Config vs Payload



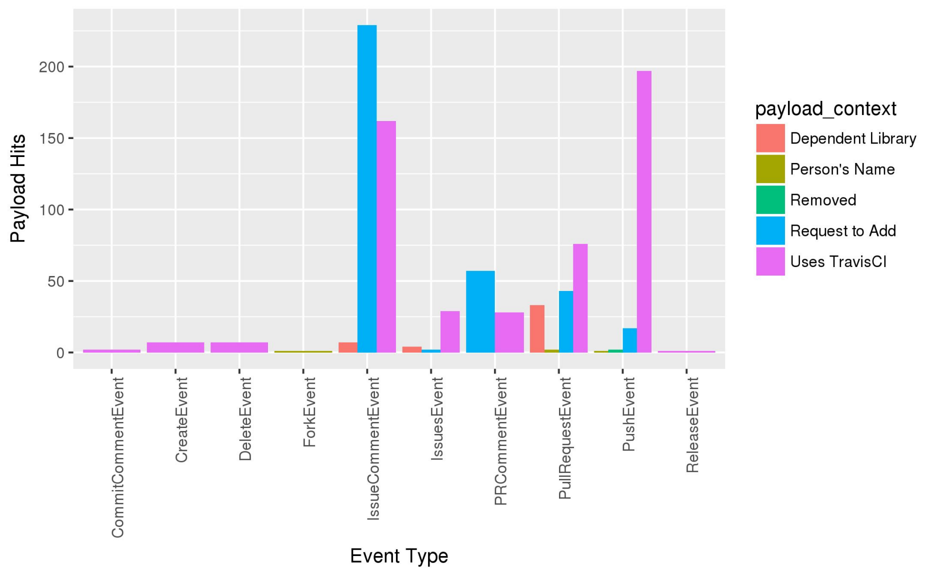
11 repos were identified via the payload text search but did not have a .travis.yml file. Note that payload failed to identify over 30 repos that were identified through in-repo configuration.

Payload False Hits



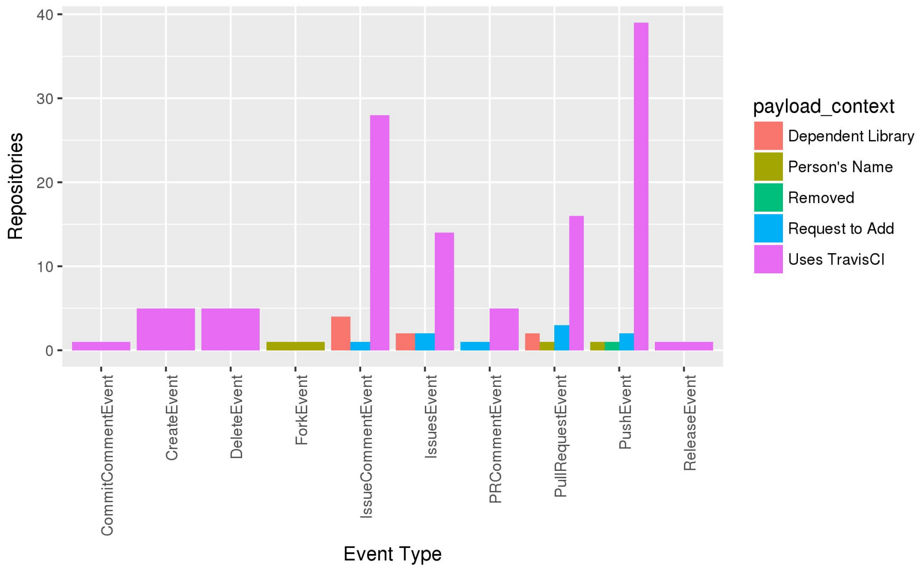
These 11 repos were manually verified and the context of why Travis was mentioned in the payload text were summarised into these categories. Dependent Library means the mention of TravisCI came from discussion or notification of an external dependency that uses TravisCI. Person’s name means a user’s name or username contained “Travis”. Removed means the repo switched from TravisCI to another CI, so while the match wasn’t entirely wrong, they aren’t currently using TravisCI. Request to Add means there is an open Pull Request with TravisCI integration code or an Issue indicating a desire to use TravisCI, but the repository doesn’t currently use TravisCI.

Payload Hits by Event Type



This looks to see if some event types show a higher rate of false positives so future searches could maybe filter these out. It looks like PushEvents had the best results with the false positives being mostly work to add Travis functionality indicating that the project wants to use it. Additionally, the project that removed TravisCI to switch to a different CI also was identified through PushEvents. IssueCommentEvents show a high rate of identification but the high false positive rate suggests this might be more trouble than is worth.

Repos with Payload Hits by Event type



This shows the same data as the previous slide but in terms of repo count instead of individual payload hits. Push Events show the highest rate of identification. Issues and IssueCommentEvents also show a high rate of identification but it also has the highest number of false positives.

Conclusions

- PR Status Host -> best for discovery
- In-Repo Config -> best for identifying current usage
- Text Search of Push Event Payloads -> best for discovering future potential use