Projects with tests

406 out of 1000 projects (41%).

Projects with unit tests

392 out of 1000 projects (41%).

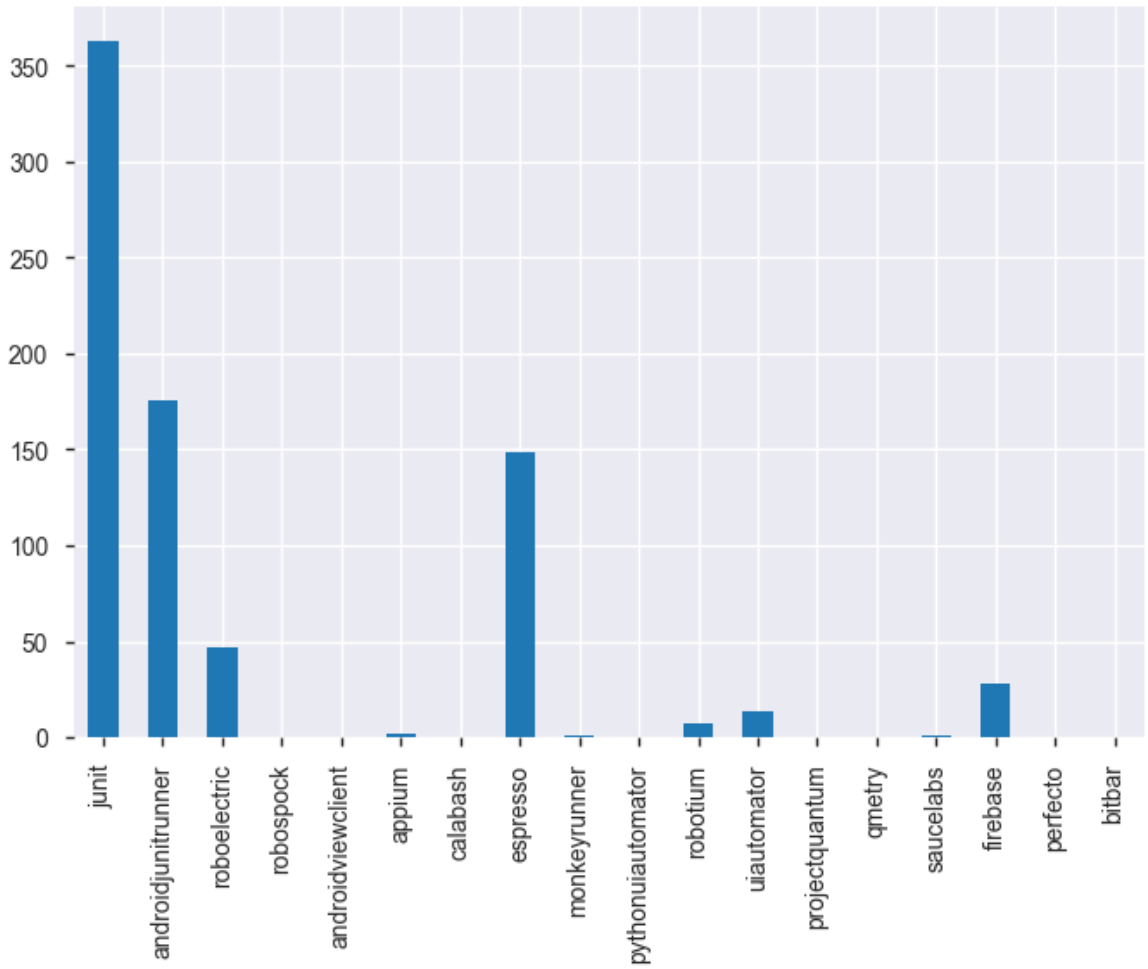
Projects using UI tests

154 out of 1000 projects (15%).

Projects using cloud test services

29 out of 1000 projects (3%).

Number of projects by framework



Conjecture¶

Unit test¶

JUnit is with no doubt the most used framework. This can be explained by the fact that Google recommends *Junit* for unit tests.
In addition, other Android testing tools developed by Google (e.g., Espresso, and UIAuomator) rely on *JUnit* for the creation of tests.
AndroidJunitRunnner is also very popular, since it allows to run unit tests on the device. This is the tool developed and recommended by Google and it is used not only for unit testing but also for other Google instrumentation frameworks.

UI tests¶

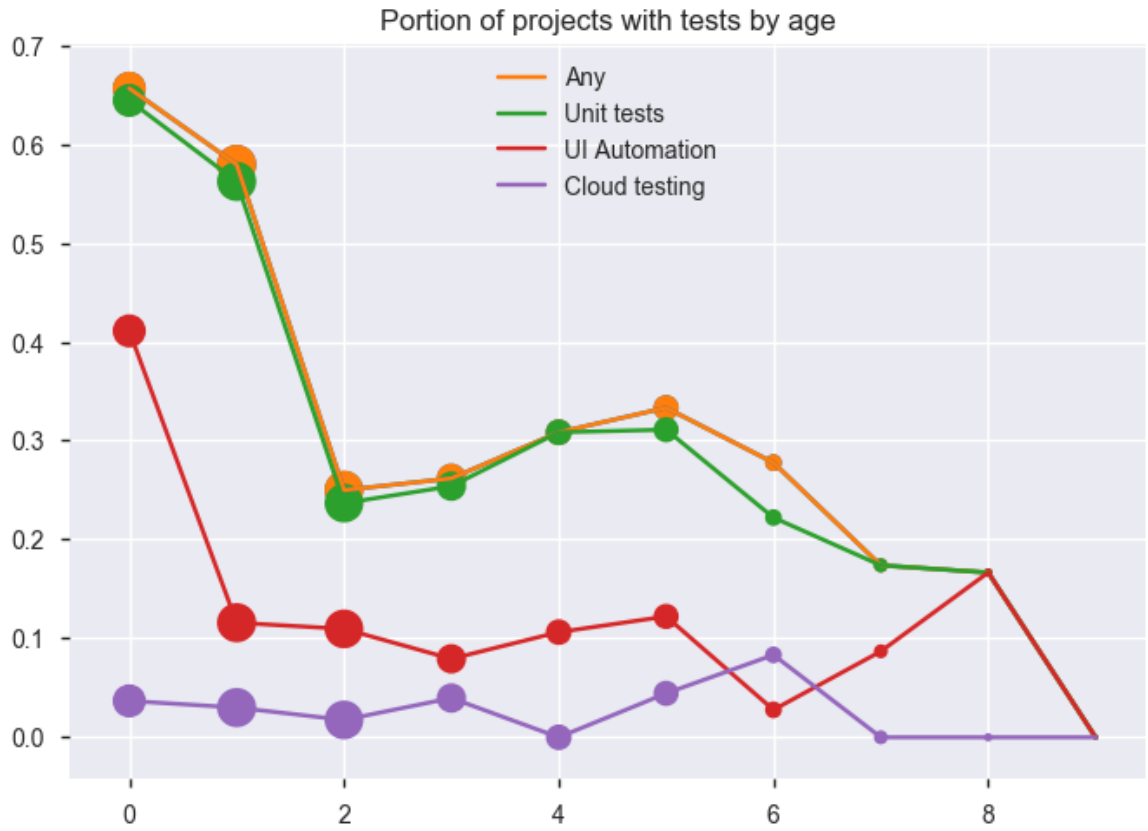
Espresso is the clear winner. The main reasons for this probably rely on the fact that Google Docs and Tutorials for Android tests first introduce Espresso. One strong feature of Espresso is the fact that it only manipulates and asserts on the application UI while it is at rest. This is very handy and makes the creation of tests a lot easier, since other

frameworks require using wait and poll mechanisms.

Cloud test services¶

Cloud test services are still very shy in the OSS world of Android apps. Only 3% (29 out of 1000) of apps use it. Google Firebase is the most popular, followed by Saucelabs. None of the other frameworks are being used by the the apps in this study.

Tests in projects by time since creation



Conjecture¶

There is an increasing trend of test usage in Android apps. Most ptojects with less than 2 years are using automated tests. In addition UI automation has became popular in the last year. Two considerations can be taken from this trend:

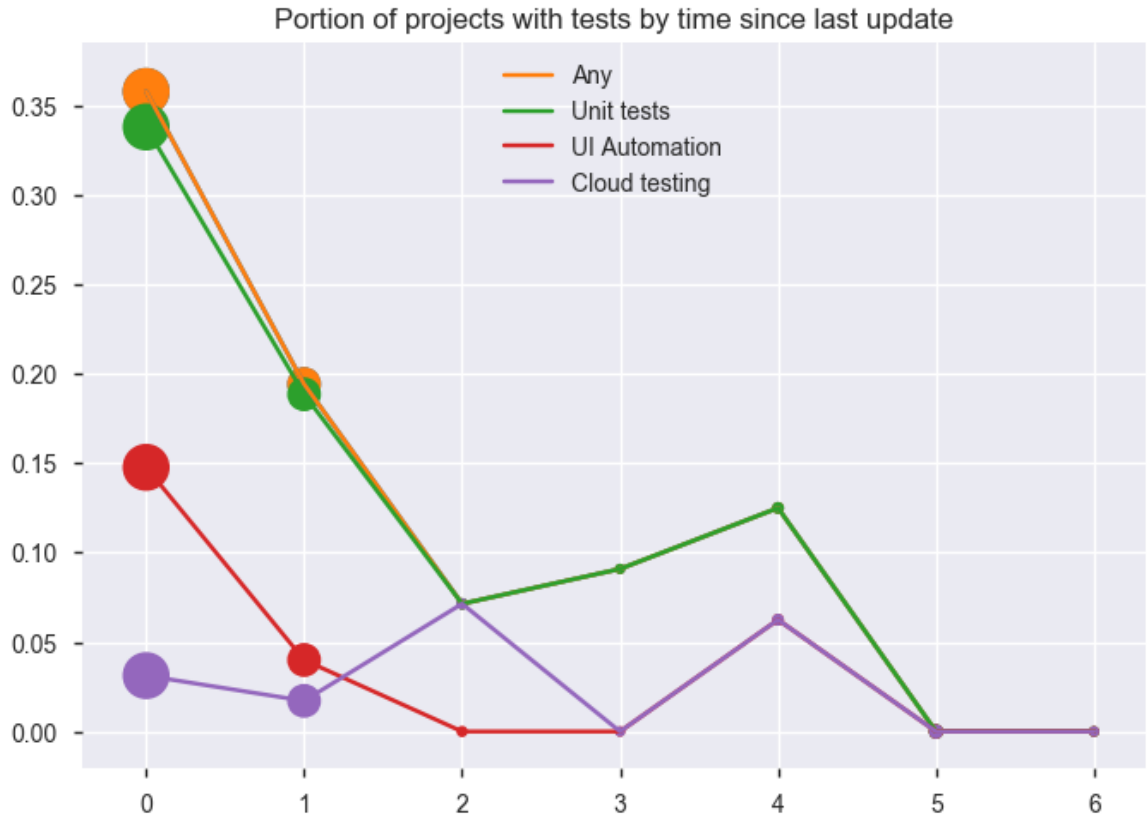
1. Test for Android apps are becoming more popular. Either because more tools are available or developers are more aware of the

importance of using automated tests in their apps.

2 Projects have tests at the begining of the project, but as soons as they get more mature and complex, tests easily become obsolete and are difficult to maintain. Hence, tests tend to be removed from these projects.

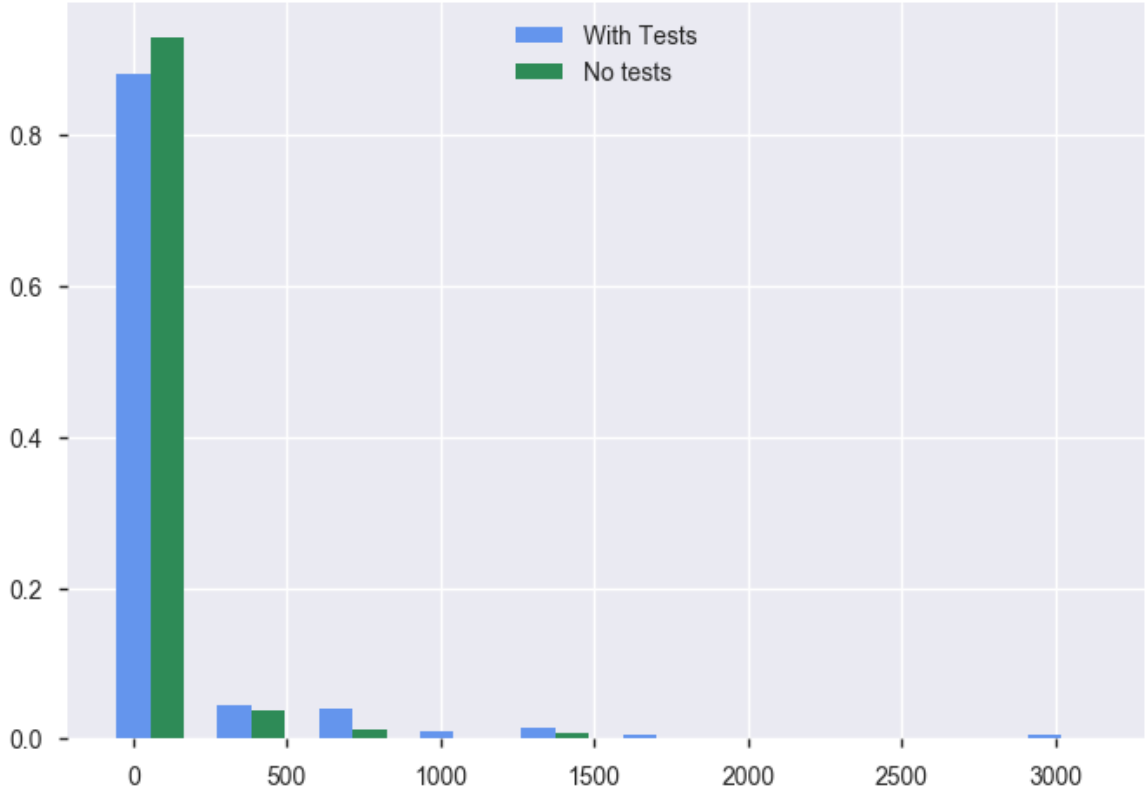
This last conjecture is very unlikely, since one could expect that some of these tests are simple or cover a stable part of the app's core, and thus could be kept in the project. Anyhow, we further analyse this data by restinging it to projects that were created before 2 years ago. Another conclusion from this visualization is that most projects that use UI automation also use unit test frameworks.

Projects that started 2 years ago by date time since last update



This visualization comprises only projects that are 2+ years old. It shows an increase interest on having automated tests in recent years.

Analysis of the impact of having tests in a project



Conjecture¶

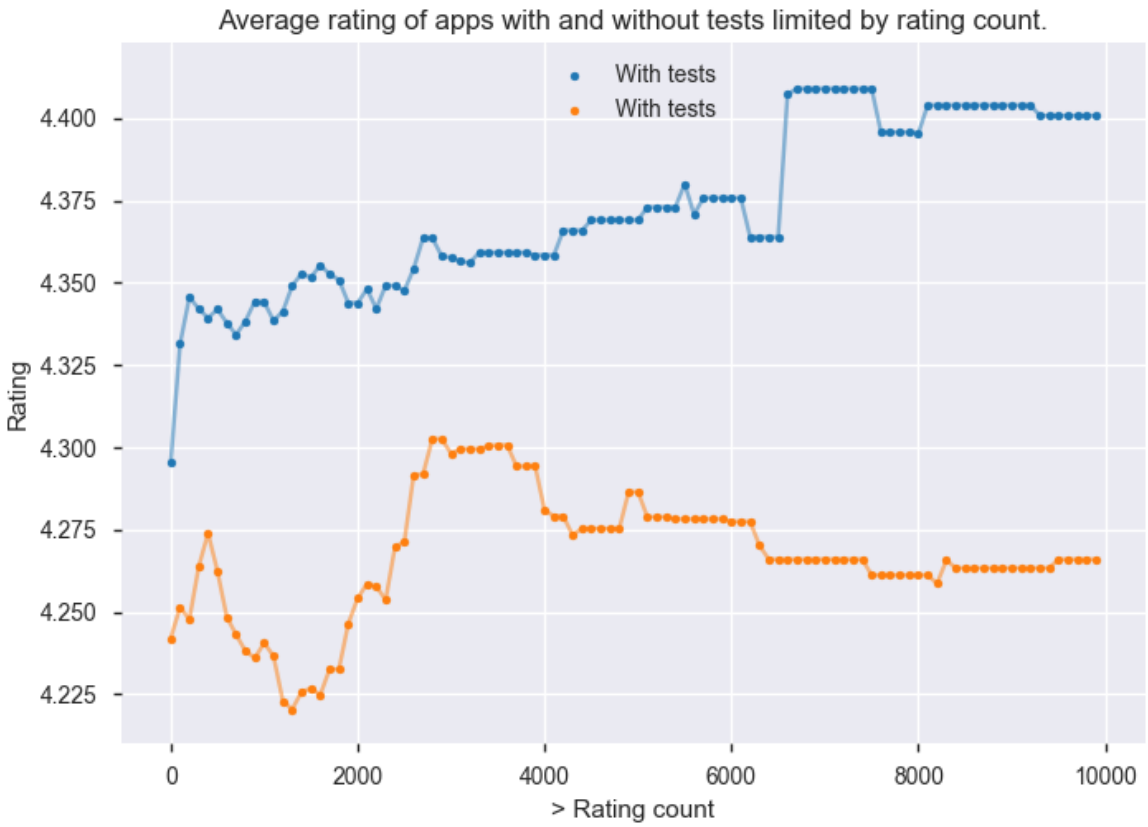
This last histogram shows that most projects have a small number of stars. However, for projects with a bigger number of stars there are more projects with tests than without.

Contributors vs Tests

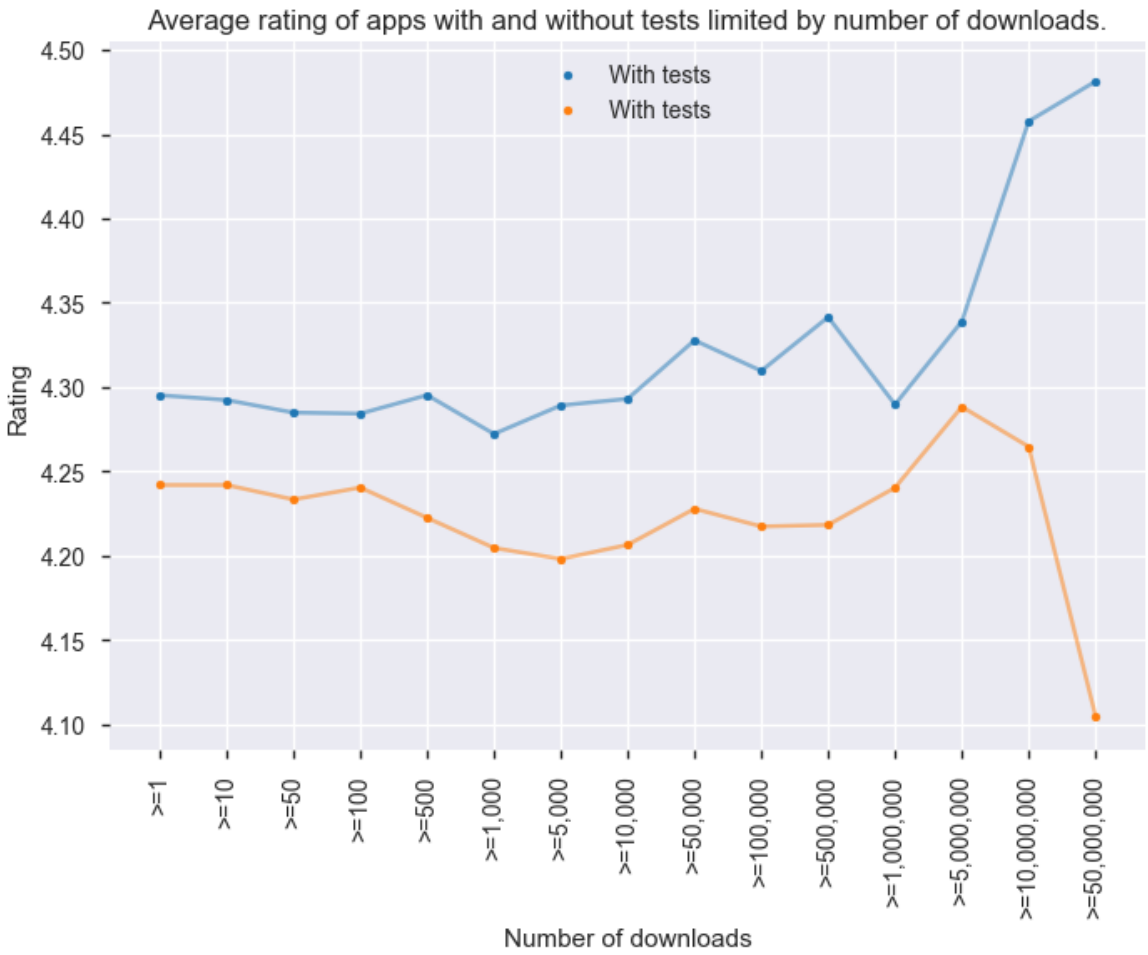
Conjecture

Projects that have tests have more contributors. This suggests that in order to get developers engaged to a project and willing to contribute, it is important that the project has tests.
The engagement of a community is very important in an Open Source project.

Google Play Rating



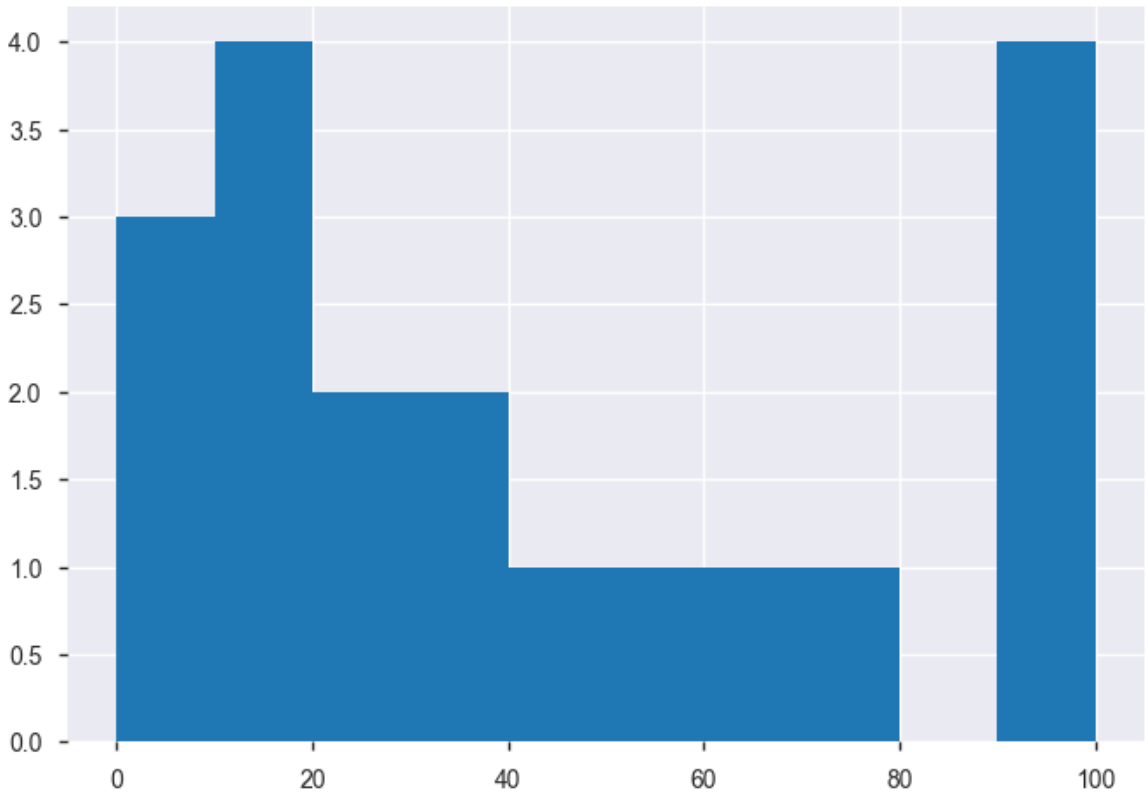
Albeit the impact of the tests on rating is small. This impact grows as long as the number of ratings in the project increases. (Question: change this graph to mean diffference?)



Projects with tests are have significantly higher rating: MannwhitneyuResult(statistic=43873.0, pvalue=0.020881059048314152).
The plot above shows the mean average ratings for apps with at least a given number of downloads (given by x axis).
Note that the x axis is in a non-linear scale (perhaps we should skip the Fives (50,500,5000) to make the plot easier to interpret).
It reveals that in apps with more than 10M users this difference is more evident.

Apps without tests are **more affected by corner cases** that are not being tested before deployment.

Coverage



Conjecture¶

Only 19 projects are using a Coverage online service (Codecov or Coderalls). From these 19, only 4 have coverage above 80%. Projects with coverage are more likely to be popular on Github. The difference of user's ratings on Google Play was not significant, due to the small number of apps using online coverage on google Play (N=12).