

# **Zen and the Art of Telemetry:**

## An Inquiry into Performance



ANTHONY ZHANG

# What?

I work with the Desktop Performance team.  
We find slow things and turn them into fast things.



Performance <3 data.  
That's where Telemetry comes in.

# Enter Telemetry

Firefox obtains (anonymous) metrics while browsing.  
We receive several hundred gigabytes of these per day.  
All that data is put together and made publicly available.

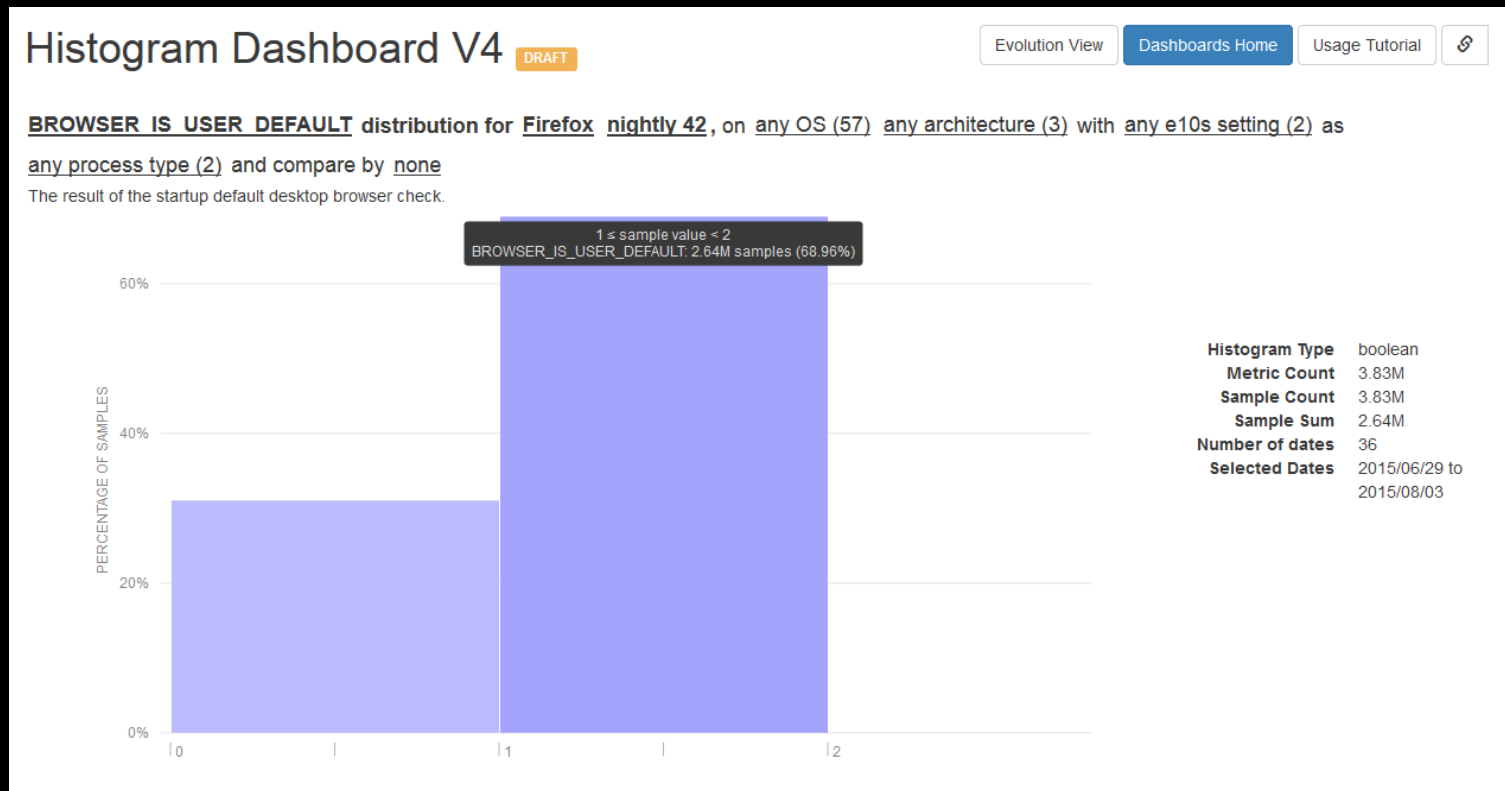
The screenshot shows the 'Telemetry Dashboards' website. At the top right is a 'Give feedback' button. The main content is organized into three columns:

- You probably want...:** Two blue buttons for 'Histogram Dashboard' (View measure value distributions as a histogram.) and 'Evolution Dashboard' (View the evolution of measure percentiles over time.).
- For power users:** A list of four dashboard options: 'Histogram Dashboard V4' (DRAFT), 'Evolution Dashboard V4' (DRAFT), 'Advanced Dashboard', and 'Classic Dashboard'. Each includes a brief description.
- Documentation and resources:** A list of links including 'Dashboard usage tutorial', 'telemetry.js documentation', 'Histogram simulator', 'How to do custom Telemetry analyses', 'How to do GFX Telemetry analyses', and 'telemetry-dashboard repository'.
- Telemetry @ Twitter:** A list of three tweets from Anthony Zhang, Mozilla Telemetry, and Eric Mill, all related to the Telemetry project.
- Other dashboards:** A list of links to various analysis dashboards: 'Addon Bootstrap Performance', 'Addon Startup Correlations', 'Addon Shutdown Correlations', 'Background Hang Reporting', 'Chrome Hangs', 'Experiment Monitoring', 'Fennec App Not Responding (ANR)', and 'Main Thread I/O'.

Check out [telemetry.mozilla.org](https://telemetry.mozilla.org)!

# Q. How often is Firefox the default browser?

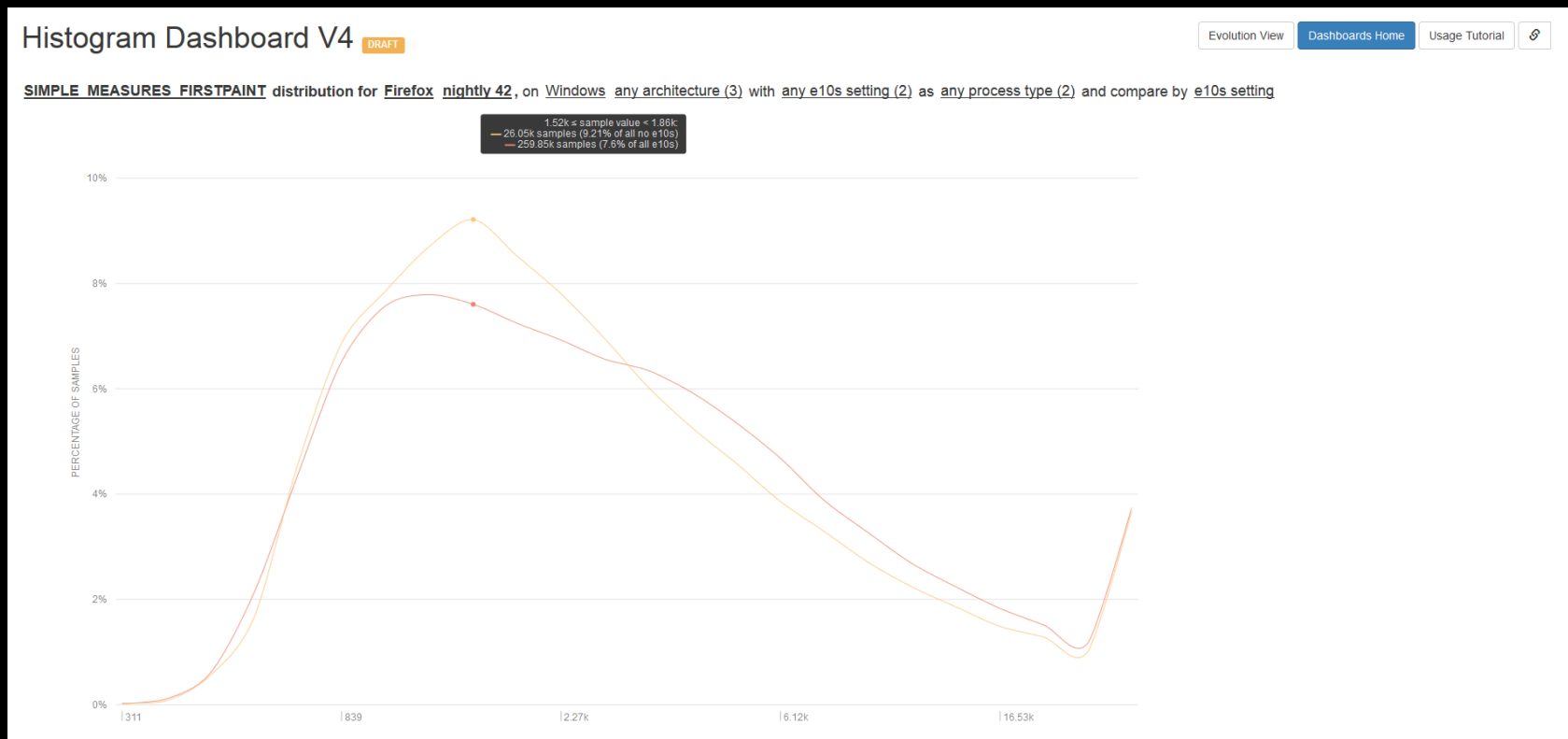
The purpose of Telemetry is to answer questions.



A. 68.96% of our nightly users have Firefox as their default browser.

# Q. Does e10s make startup faster?

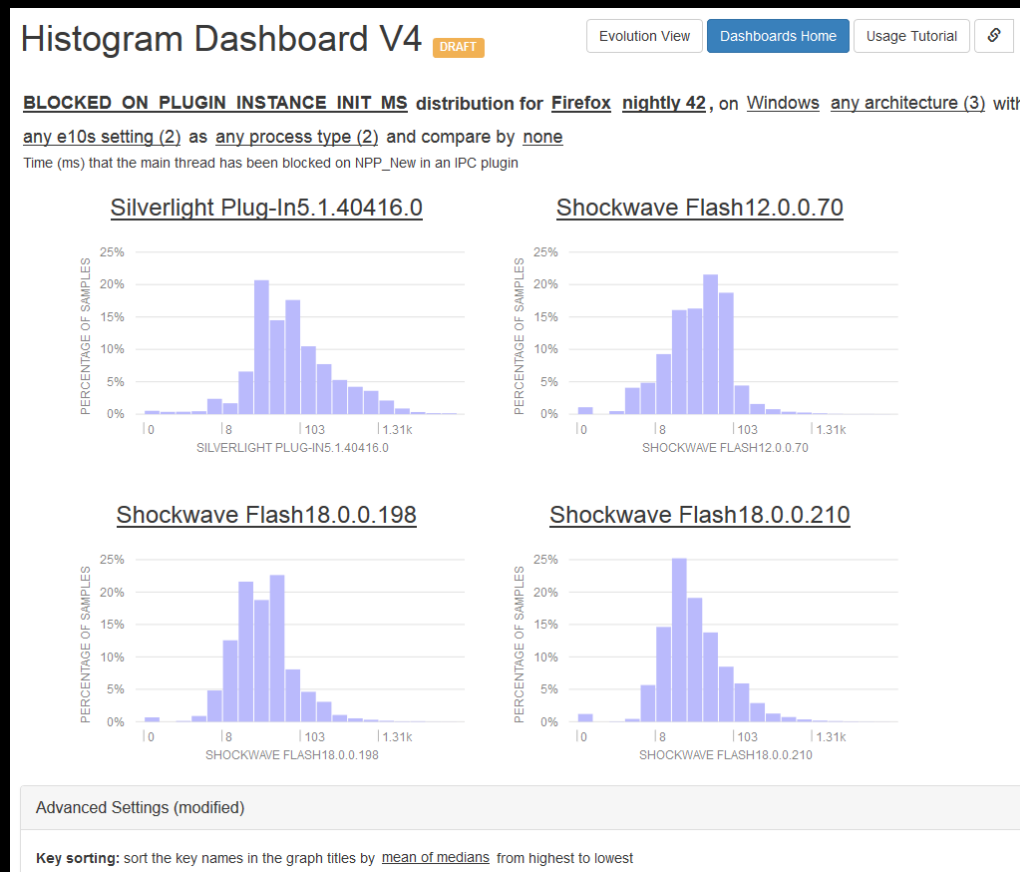
The dashboards on [telemetry.mozilla.org](https://telemetry.mozilla.org) cover many common use cases.



A. No, it's slightly slower.

# Q. Which plugins tend to freeze the browser on load?

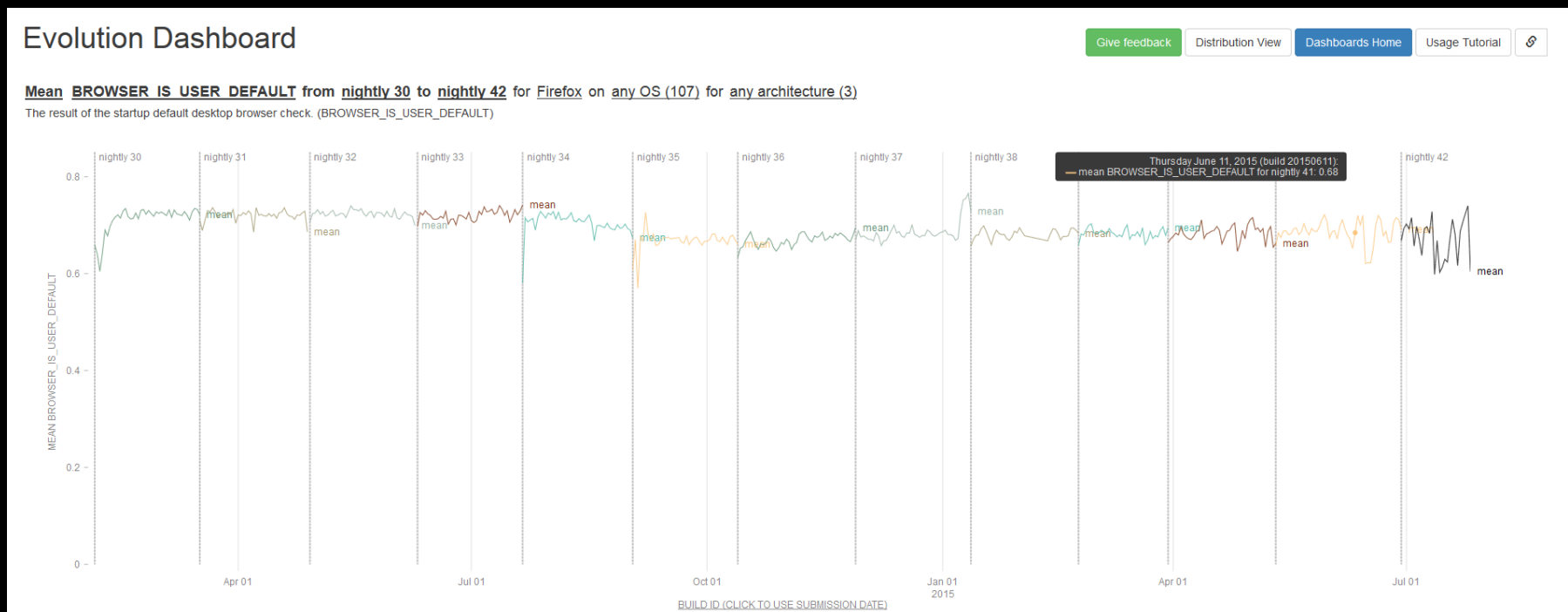
Better dashboards make it easier to make data-driven decisions.



A. Silverlight and Flash (is anyone surprised?)

# Q. How has default-browser-ness changed over time?

There are also lots of other, more specialized views available on [telemetry.mozilla.org](https://telemetry.mozilla.org).



A. The fraction of nightly users with Firefox as their default browser is consistently around 70%.

# Telemetry.js

Can't answer your question with the dashboards?  
Make your own with Telemetry.js!

```
    console.log("Available dates:\n" + evolutionMap[""].dates().join("\n"));  
  });  
});
```

Getting the overall median value of GC\_MS on nightly 42 on Windows from July 19, 2015 to August 1, 2015:

```
Telemetry.init(function() {  
  Telemetry.getEvolution("nightly", "42", "GC_MS", {os: "Windows_NT"}, true, function(evolutionMap) {  
    var evolution = evolutionMap[""].dateRange(new Date("2015-07-19"), new Date("2015-08-01"));  
    var histogram = evolution.histogram();  
    console.log("Median GC_MS: " + histogram.percentile(50));  
  });  
});
```

## API Reference

```
Telemetry.init(function callback() { ... })
```

Initializes Telemetry.js with various pieces of metadata that are used by a lot of the other functionality in the library. When

The library runs in the browser or in Node.js.  
We also have an API!



**I still have questions!**

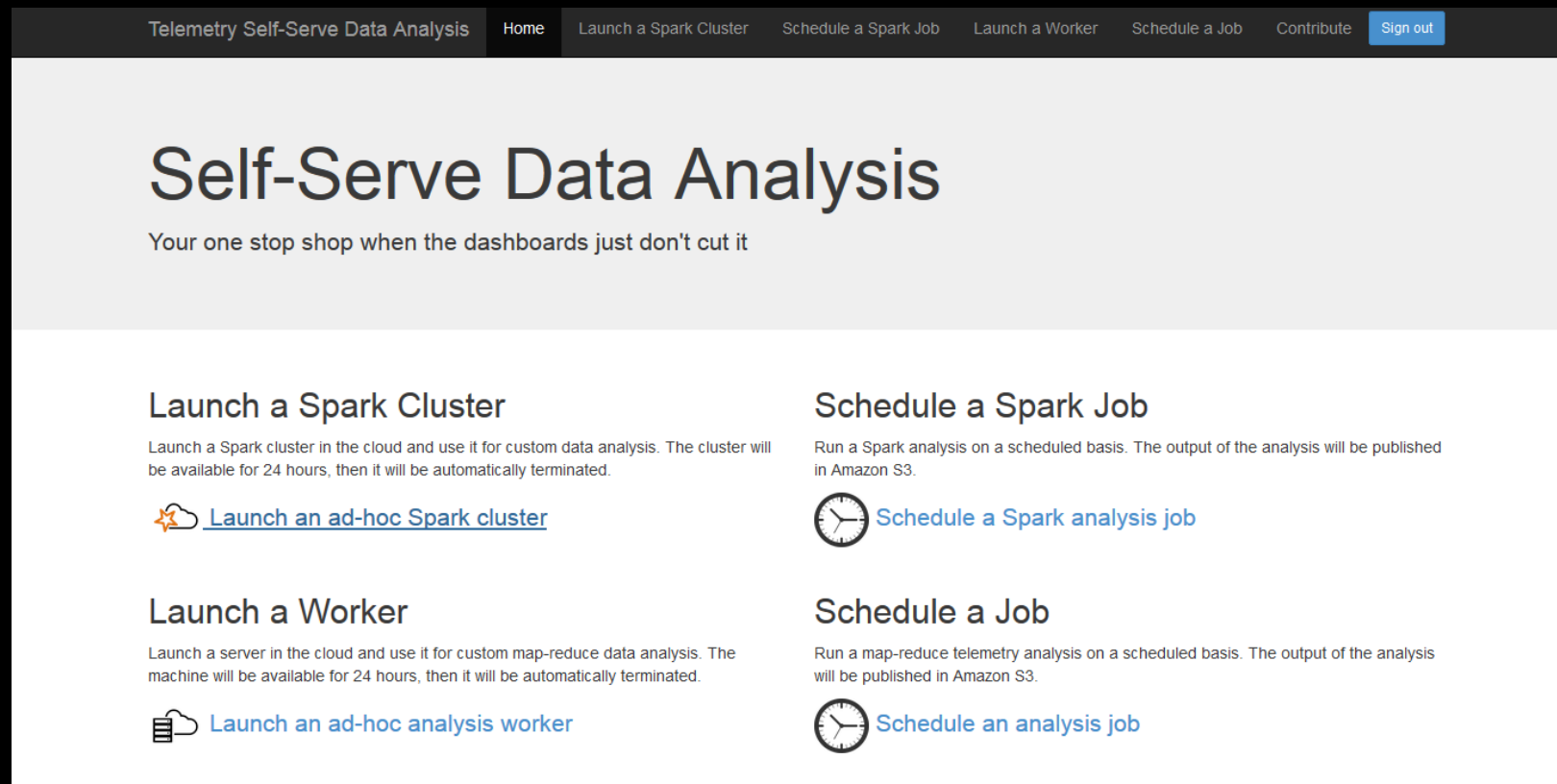
We've got you covered.  
Let's say we have a more unusual request.

**To what extent is start-up time correlated  
to the current phase of the moon?**

Follow along with the code on [git.io/vOhTI](https://git.io/vOhTI).

# Start your browsers...

For everything else, there's custom Telemetry analyses.



The screenshot shows a web dashboard with a dark navigation bar at the top containing links: Telemetry Self-Serve Data Analysis, Home, Launch a Spark Cluster, Schedule a Spark Job, Launch a Worker, Schedule a Job, Contribute, and Sign out. The main content area has a light gray header with the title 'Self-Serve Data Analysis' and the subtitle 'Your one stop shop when the dashboards just don't cut it'. Below this are four cards arranged in a 2x2 grid. Each card has a title, a brief description, and a link with an icon.


Telemetry Self-Serve Data Analysis Home Launch a Spark Cluster Schedule a Spark Job Launch a Worker Schedule a Job Contribute Sign out

## Self-Serve Data Analysis

Your one stop shop when the dashboards just don't cut it


### Launch a Spark Cluster

Launch a Spark cluster in the cloud and use it for custom data analysis. The cluster will be available for 24 hours, then it will be automatically terminated.

 [Launch an ad-hoc Spark cluster](#)


### Schedule a Spark Job

Run a Spark analysis on a scheduled basis. The output of the analysis will be published in Amazon S3.

 [Schedule a Spark analysis job](#)


### Launch a Worker

Launch a server in the cloud and use it for custom map-reduce data analysis. The machine will be available for 24 hours, then it will be automatically terminated.

 [Launch an ad-hoc analysis worker](#)

### Schedule a Job

Run a map-reduce telemetry analysis on a scheduled basis. The output of the analysis will be published in Amazon S3.

 [Schedule an analysis job](#)

Head on over to [telemetry-dash.mozilla.org](https://telemetry-dash.mozilla.org)!

# Titles are hard

Let's write code instead:

## Moon Phase Correlation Analysis

```
In [ ]: from moztelemetry import get_pings, get_pings_properties, get_one_ping_per_client
```

This [Wikipedia article](#) has a nice description of how to calculate the current phase of the moon. In code, that looks like this:

```
In [2]: def approximate_moon_visibility(current_date):
        days_per_synodic_month = 29.530588853 # change this if the moon gets towed away
        days_since_known_new_moon = (current_date - dt.date(2015, 7, 16)).days
        phase_fraction = (days_since_known_new_moon % days_per_synodic_month) / days_per_synodic_month
        return (1 - phase_fraction if phase_fraction > 0.5 else phase_fraction) * 2

        def date_string_to_date(date_string):
            return dt.datetime.strptime(date_string, "%Y%m%d").date()
```

So far, so good.

# Let's get messy

## Time to get ourselves some data!

Let's randomly sample 10% of pings for nightly submissions made from 2015-07-05 to 2015-08-05:

```
In [4]: pings = get_pings(sc, app="Firefox", channel="nightly", submission_date=("20150705", "20150805"), fraction=0.1, schema="v4")
```

Extract the startup time metrics with their submission date and make sure we only consider one submission per user:

```
In [5]: subset = get_pings_properties(pings, ["clientId", "meta/submissionDate", "payload/simpleMeasurements/firstPaint"])
subset = get_one_ping_per_client(subset)
cached = subset.cache()
```

Obtain an array of pairs, each containing the moon visibility and the startup time:

```
In [16]: pairs = cached.map(lambda p: (approximate_moon_visibility(date_string_to_date(p["meta/submissionDate"])), p["payload/simpleMeasurements/firstPaint"]))
pairs = np.asarray(pairs.filter(lambda p: p[1] != None and p[1] < 100000000).collect())
```

This part takes about 10 minutes to run.

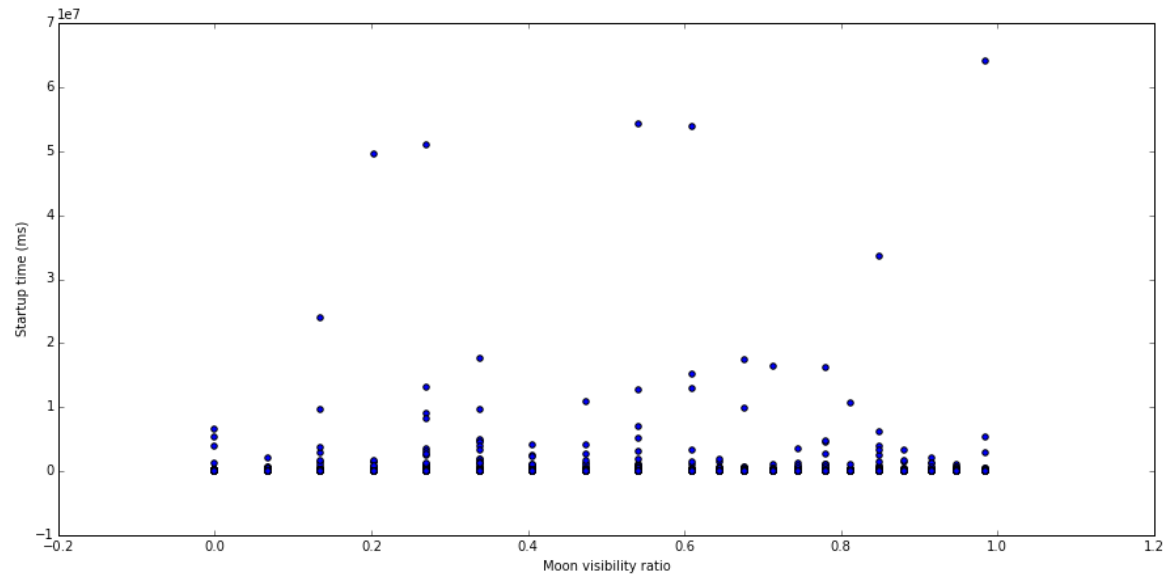
There's about 900k total unique users submitting start-up time metrics in the specified period.

# Show me the numbers!

Plots are a great way to check your answers:

Let's see what this data looks like:

```
In [18]: plt.figure(figsize=(15, 7))
plt.scatter(pairs.T[0], pairs.T[1])
plt.xlabel("Moon visibility ratio")
plt.ylabel("Startup time (ms)")
plt.show()
```



The correlation coefficient is now easy to calculate:

```
In [19]: np.corrcoef(pairs.T)[0, 1]
```

```
Out[19]: 0.00048989909014640089
```

## **The verdict**

We see that the correlation is roughly 0.0005.

**The moon does not have a significant  
effect on Firefox start-up time.**

**(in the last month)**

**(on nightly)**

**(for now)**

# Links and stuff

## **Telemetry Dashboards/Documentation**

[telemetry.mozilla.org](https://telemetry.mozilla.org)

## **Telemetry Demystified**

[anthony-zhang.me/blog/telemetry-demystified](https://anthony-zhang.me/blog/telemetry-demystified)

## **Custom Telemetry Analyses**

[telemetry-dash.mozilla.org](https://telemetry-dash.mozilla.org)

## **Performance Team @ Mozilla**

[wiki.mozilla.org/Performance](https://wiki.mozilla.org/Performance)

## **Example Analysis: Moon Phase Correlation**

[git.io/vOhTI](https://git.io/vOhTI)

We're on Twitter too! @MozTelemetry

Also, #perf @ irc.mozilla.org.