mozilla

Telemetry Onboarding

Firefox Data Platform Team

Public

Structure

1. Privacy & Policies
2. What is Telemetry?
3. Summary dashboards
4. Telemetry dashboards
5. Adding a probe
6. What is a ping?
7. Data Pipeline
8. Experiments
9. Offline Processing
10. Stats

Public

Before we Begin

https://analysis.telemetry.mozilla.org/

Public

Privacy Policy

1. No surprises: use and share information in a way that is transparent and benefits the user

2. User control: develop products and advocate for best practices that put users in control of their data and online experiences

3. Limited data: collect what we need, de-identify where we can and delete when no longer necessary

4. Sensible settings: design for a thoughtful balance of safety and user experience

5. Defense in depths: maintain multi-layered security controls and practices, many of which are publicly verifiable

https://www.mozilla.org/privacy/

Public

Data Collection Policy

mozilla

When proposing a new measurement or data system, consider the requirements and the necessary data properties, e.g:

- Is it necessary to take a measurement from all users? Or is it sufficient to measure only prerelease users?
- Is it desirable to track data changes over time? With what frequency? With what latency?

For every new measurement, even a trivial measurement, please request approval by setting the feedback flag for the module owner or a peer.

Owner: Benjamin Smedberg

https://wiki.mozilla.org/Firefox/Data_Collection

Public

Questions?

Public

What is Telemetry?

Telemetry is a system that:

- measures how Firefox behaves in the real world
- collects non-personal information about performance, hardware, and other stuff,
- allows us to identify bugs, issues, and regressions
- allows us to conduct longitudinal studies and experiments

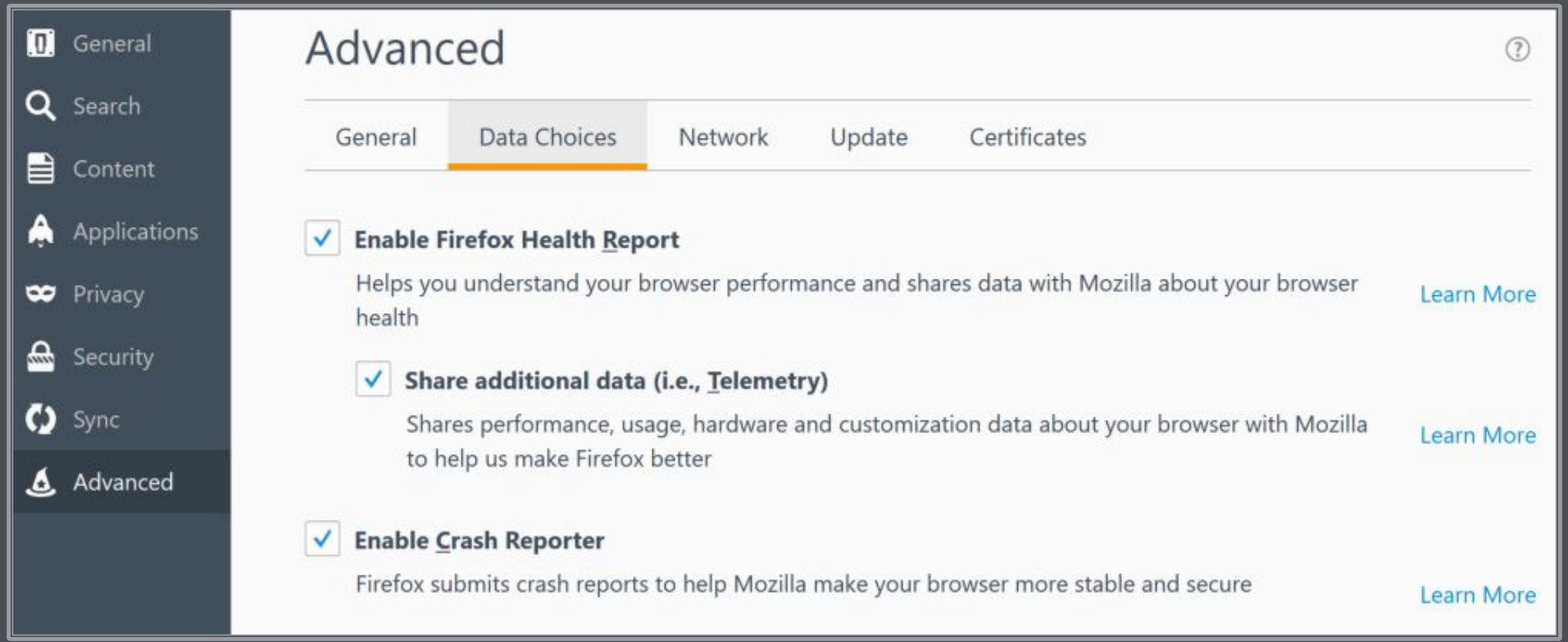
Public

Data Sets

- Base Telemetry (formerly FHR) contains critical, representative data that supports longitudinal studies
  - Enabled by default on all channels

- Extended Telemetry send richer performance and usage data
  - Disabled by default on release
  - Enabled by default on prerelease

Public

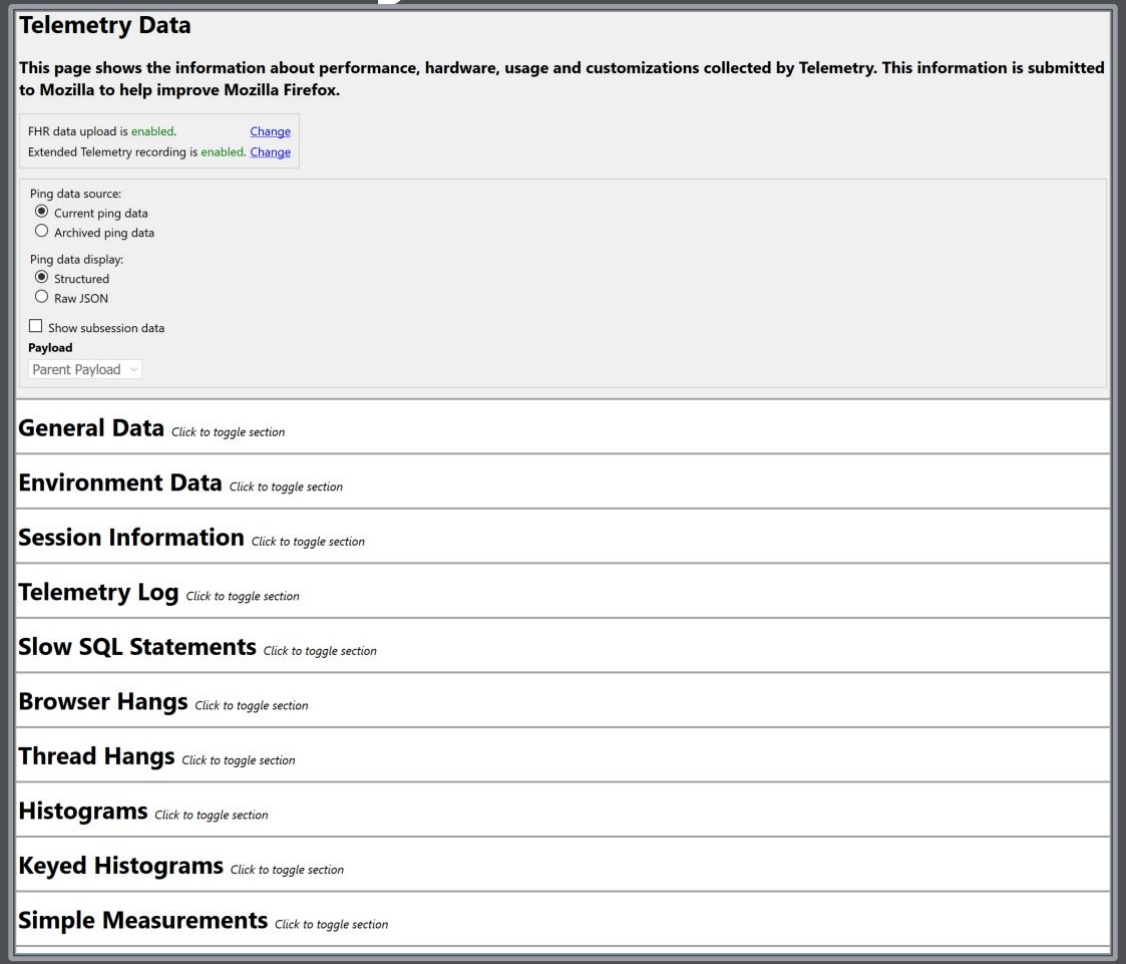



```
export MOZ_TELEMETRY_REPORTING=1
#export MOZILLA_OFFICIAL=1
```

Public

about:telemetry

mozilla



Public

References

https://wiki.mozilla.org/Telemetry

https://wiki.mozilla.org/Firefox_Health_Report

Public

Questions?

Public

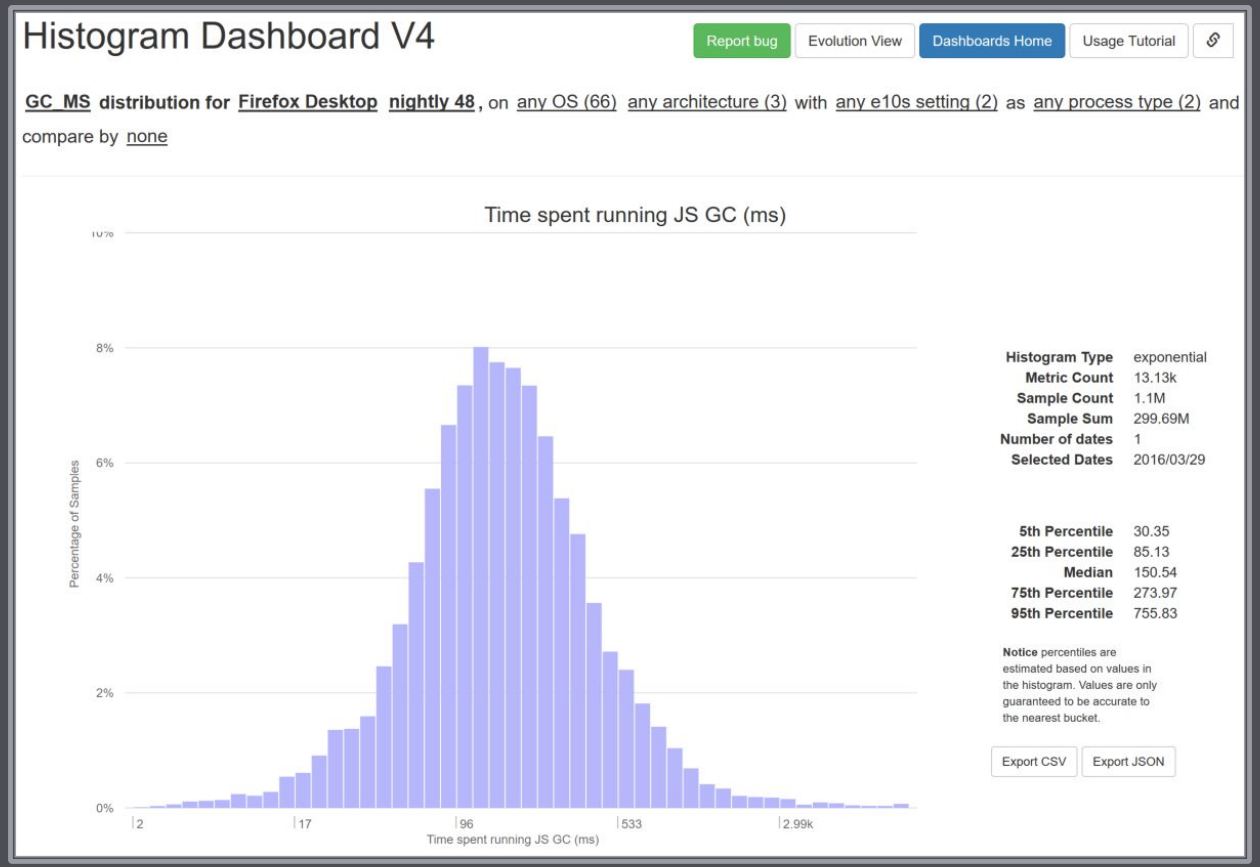
Summary Dashboards

https://metrics.services.mozilla.com/

Questions?

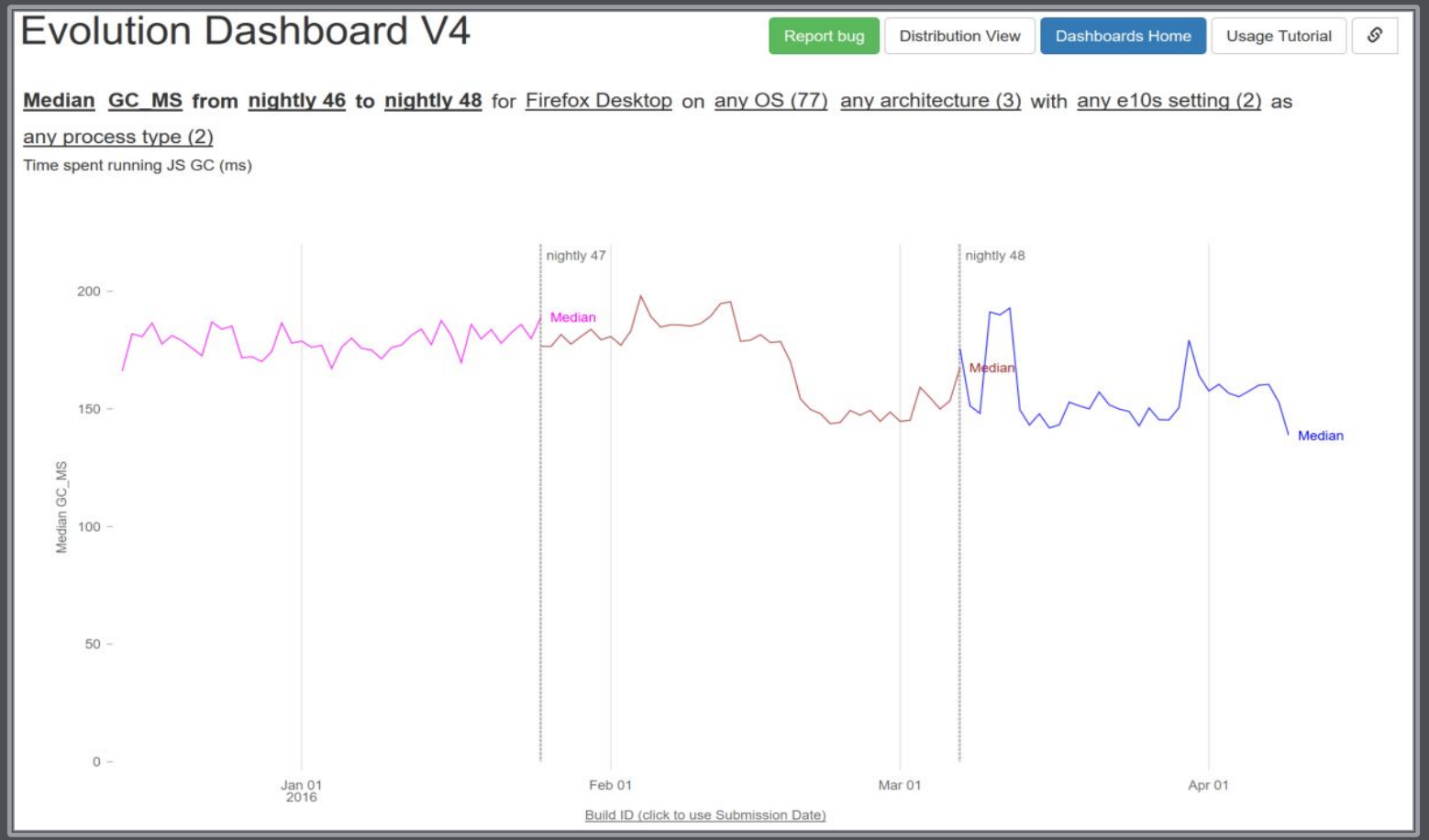
Public

Telemetry Dashboards

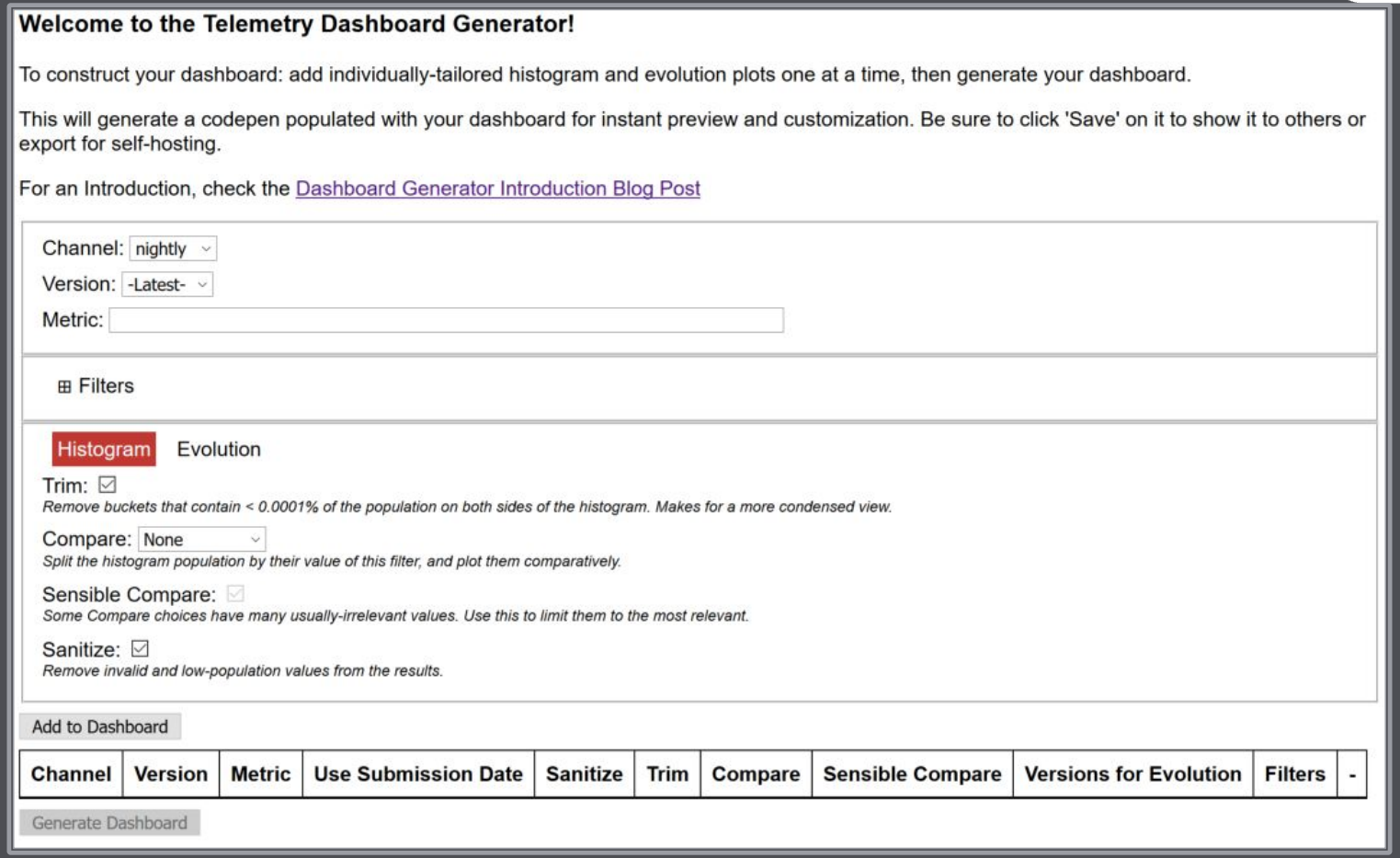


https://telemetry.mozilla.org/

Public



https://telemetry.mozilla.org/new-pipeline/evo.html



https://telemetry.mozilla.org/dashboard-generator/index.html

Public

Aggregates by JS

telemetry.js v2:

```
Telemetry.init(function() {
    var versions = Telemetry.getVersions("nightly/40", "nightly/42");
    console.log("Versions between nightly 40 to nightly 42 (inclusive):\n" + versions.join("\n"));
});
```

https://github.com/mozilla/telemetry-dashboard/blob/master/v2/doc.md

telemetry-wrapper.js:

```
window.TelemetryWrapper.go(params, parentEl);
```

https://github.com/mozilla/telemetry-dashboard/tree/gh-pages/wrapper

Public

Aggregates by HTTPS

mozilla

Get available channels:

```
curl -X GET http://SERVICE/aggregates_by/build_id/channels/
["nightly","beta","aurora"]
```

Get a list of options for the available dimensions on a given channel and version:

```
curl -X GET "http://SERVICE/filters/?channel=nightly&version=42"
{"metric":["A11Y_CONSUMERS","A11Y_IATABLE_USAGE_FLAG",...],
 "application":["Fennec","Firefox"],
 ...}
```

Get a list of available build-ids for a given channel:

```
curl -X GET "http://SERVICE/aggregates_by/build_id/channels/nightly/dates/"
[{"date":"20150630","version":"42"}, {"date":"20150629","version":"42"}]
```

Given a set of build-ids, retrieve for each of build-id the aggregated histogram that complies with the requested filters:

```
curl -X GET "http://SERVICE/aggregates_by/build_id/channels/nightly/?version=41&dates=20150615,20150616&metric=GC_MS
{"buckets":[0, ..., 10000],
 "data":[{"date":"20150615",
          "count":239459,
          "sum": 412346123,
          "histogram":[309, ..., 5047],
          "label":""},
         {"date":"20150616",
          "count":233688,
          "sum": 402241121,
          "histogram":[306, ..., 7875],
          "label":""}],
 "kind":"exponential",
 "description":"Time spent running JS GC (ms)"}
```

https://github.com/vitillo/python_mozaggregator/blob/master/README.md

Public

Telemetry alerts

Login

Histogram Regression Detector

Telemetry Main-thread IO

Unpacked Add-on File Scan

Filter Metrics By:

TAB_ANIM

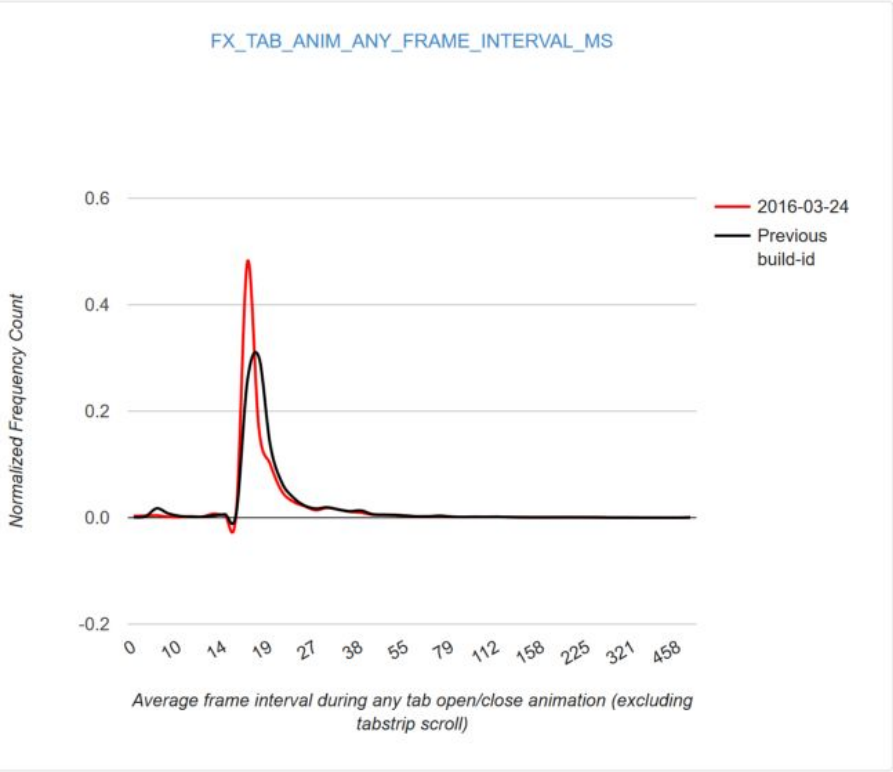
Select Metric:

From:

2016-03-11

To:

2016-04-11



https://alerts.telemetry.mozilla.org/

References

http://robertovitillo.com/2015/07/02/telemetry-metrics-roll-ups/

https://anthony-zhang.me/blog/telemetry-demystified/

http://robertovitillo.com/2014/07/28/regression-detection-for-telemetry-histograms/

https://telemetry.mozilla.org/

https://metrics.services.mozilla.com/

https://chuttenblog.wordpress.com/tag/telemetry/

Public

Questions?

Public

Adding a Histogram

Telemetry histograms are the preferred way to track numeric measurements. There are six types:

- flag, e.g. FXA_CONFIGURED
- boolean, e.g. E10S_WINDOW
- count, e.g. CONTENT_DOCUMENTS_DESTROYED
- enumerated, e.g. DEVICE_RESET_REASON
- linear, e.g. GC_MAX_PAUSE_MS
- exponential, e.g. GC_MARK_MS

https://developer.mozilla.org/docs/Mozilla/Performance/Adding_a_new_Telemetry_probe

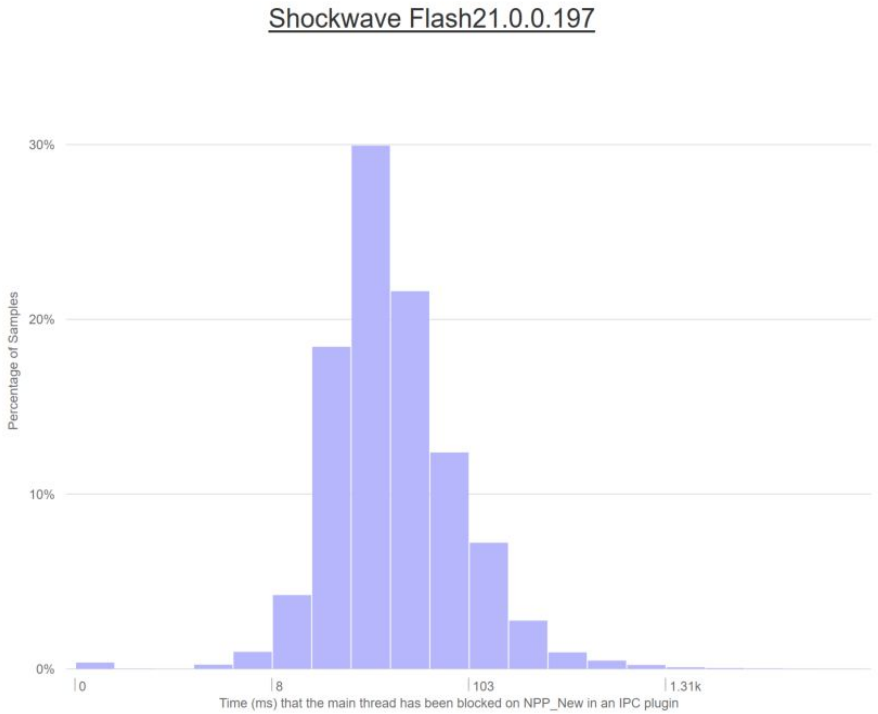
Public

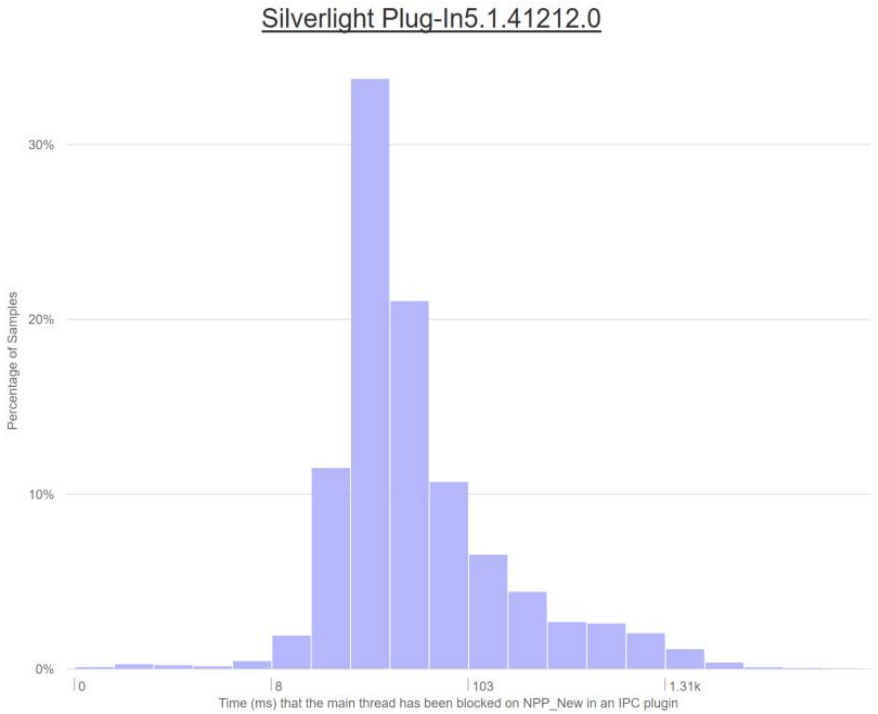
Keyed Histograms

mozilla

BLOCKED_ON_PLUGIN_INSTANCE_INIT_MS distribution for **Firefox Desktop** **nightly 48**, on any OS (66) any architecture (3) with any e10s setting (2) as any process type (2) and compare by none

Time (ms) that the main thread has been blocked on NPP_New in an IPC plugin





Declaring a Histogram

toolkit/components/telemetry/Histograms.json

```
"TELEMETRY_TUTORIAL_PROBE": {
    "alert_emails": ["rvitillo@mozilla.com"],
    "bug_numbers": [1242013],
    "expires_in_version": "55",
    "kind": "exponential",
    "high": 1000,
    "n_buckets": 50,
    "description": "Telemetry tutorial probe (ms)"
},
```

./mach build toolkit/components/telemetry
./mach run

Public

Accumulating Data

mozilla

Go to about:telemetry and bring up a console with Ctrl+Shift+K

```
Telemetry.isOfficialTelemetry
<- false

h = Services.telemetry.getHistogramById("TELEMETRY_TUTORIAL_PROBE")
<- JSHistogram { , 4 more… }

h.add(42)
<- function add()
```

Public

References

https://developer.mozilla.org/en-US/docs/Mozilla/Performance/Adding_a_new_Telemetry_probe

Public

Questions?

Public

What is a Ping?

A Telemetry ping is the data that we send to Telemetry servers.

On the client it is stored as JSON

Pings follow a common data format. Different types of pings have different types of data payloads.

https://gecko.readthedocs.org/latest/toolkit/components/telemetry/telemetry/

Public

Ping Types

“main”: contains most of the measurements that track the performance and health of Firefox instances in the wild. Its “reason” field documents what triggered the ping (e.g. shutdown)

“crash”: captured after the parent Firefox process crashes

others: there are others of more specific utility, and you can also define and submit custom ones if you’d like.

Public

http://gecko.readthedocs.org/latest/toolkit/components/telemetry/telemetry/pings.html#ping-types

Common Ping Format

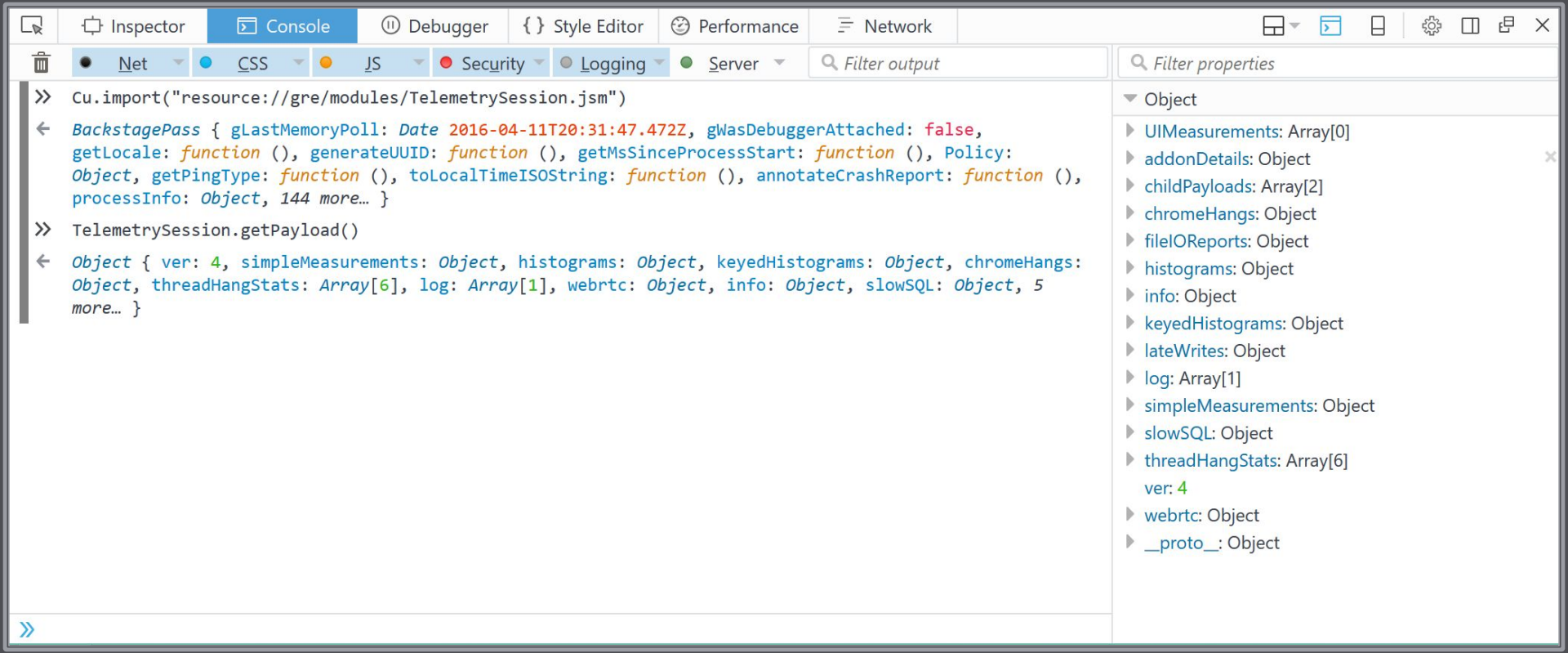
```
{
  type: <string>, // "main", "activation", "deletion", "saved-session", ...
  id: <UUID>, // a UUID that identifies this ping
  creationDate: <ISO date>, // the date the ping was generated
  version: <number>, // the version of the ping format, currently 4

  application: {
    architecture: <string>, // build architecture, e.g. x86
    buildId: <string>, // "20141126041045"
    name: <string>, // "Firefox"
    version: <string>, // "35.0"
    displayVersion: <string>, // "35.0b3"
    vendor: <string>, // "Mozilla"
    platformVersion: <string>, // "35.0"
    xpcomAbi: <string>, // e.g. "x86-msvc"
    channel: <string>, // "beta"
  },

  clientId: <UUID>, // optional
  environment: { ... }, // optional, not all pings contain the environment
  payload: { ... }, // the actual payload data for this ping type
}
```

http://gecko.readthedocs.org/latest/toolkit/components/telemetry/telemetry/common-ping.html

Public

Main Ping Payload

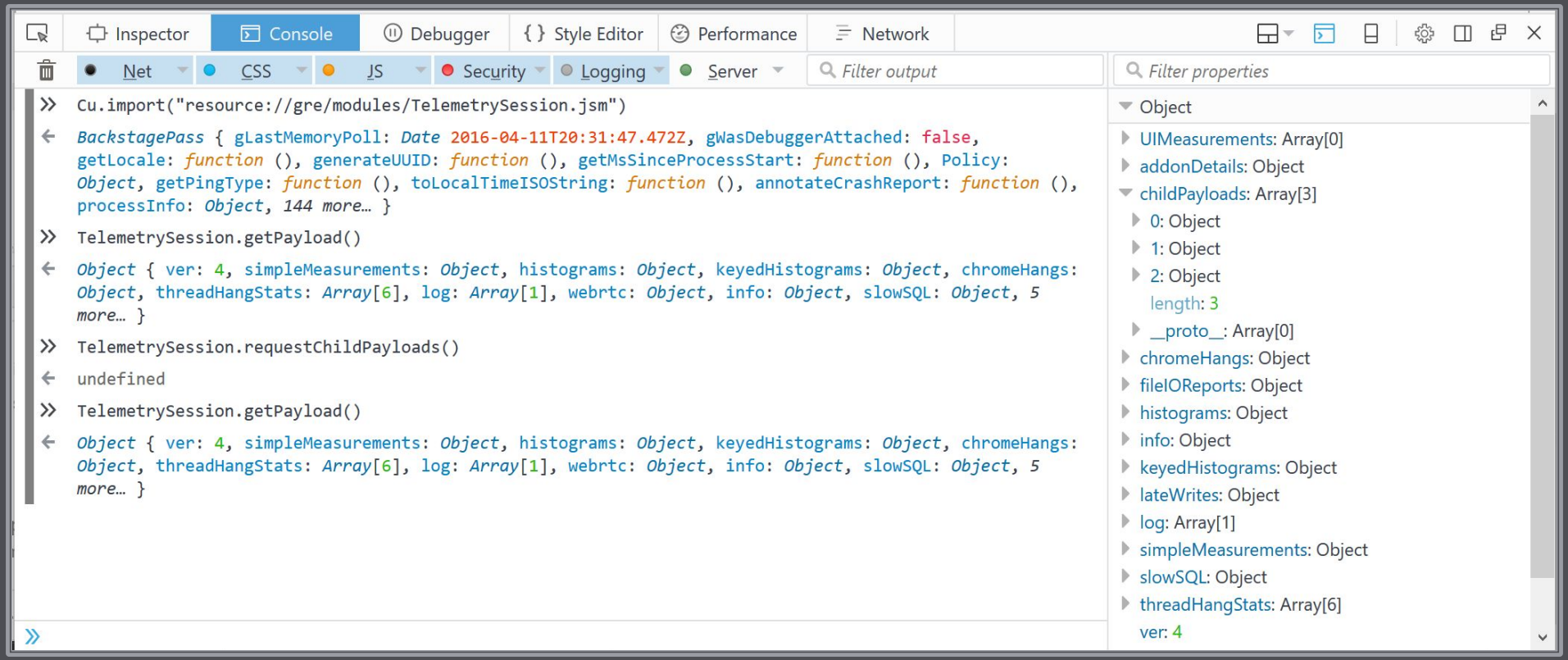
mozilla



Cu.import("resource://gre/modules/TelemetrySession.jsm")

Public

E10s Caveat



Public

Environment

Data that is expected to be characteristic of performance and other behaviour.

Not expected to change too often.

Changes to many of these fields is detected and leads to a session split in the “main” ping.

http://gecko.readthedocs.org/latest/toolkit/components/telemetry/telemetry/environment.html

Public

References

http://gecko.readthedocs.org/latest/toolkit/components/telemetry/telemetry/index.html

Public

Questions?

Public

Break Time

Public

Data Pipeline

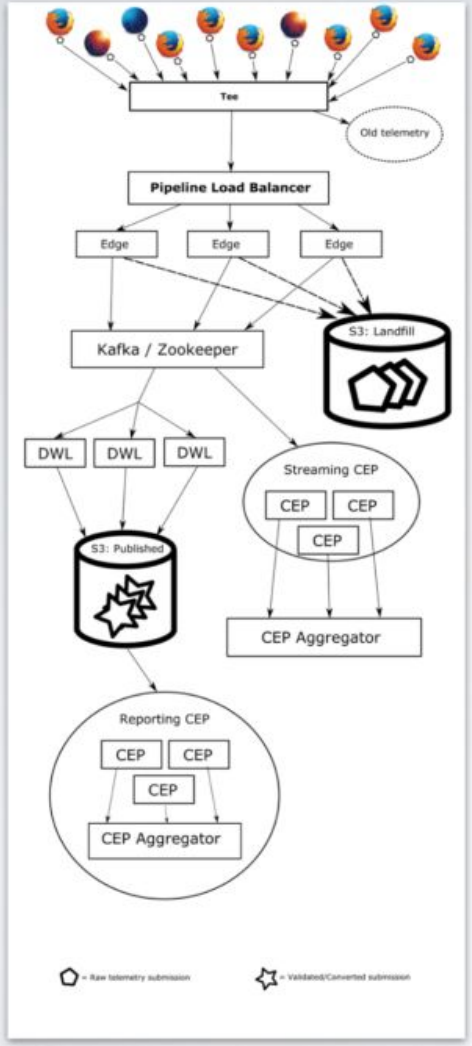
How we ingest, transform, store and analyse incoming data

The endpoint is an HTTP server that

- listens for POST/PUT from Firefox
- does some decoding/preprocessing
- sends data off for streaming analyses (Heka)
- archives data to S3 for offline analyses (Spark)

https://wiki.mozilla.org/CloudServices/DataPipeline

Public



CEP = Complex Event Processor (basically streaming analysis or reporting)

DWL = Data Warehouse Loader

Landfill = Shorthand for "write-mostly store for backup / recovery purposes"

mozilla

Public

References

https://wiki.mozilla.org/CloudServices/DataPipeline

https://github.com/mozilla-services/data-pipeline

https://github.com/mozilla-services/heka

https://github.com/apache/spark/

Public

Questions?

Public

Offline Processing

IPython, Pandas, and Spark

https://github.com/vitillo/telemetry-onboarding/tree/master/notebooks

Public

Offline Processing

mozilla

Using SQL

https://sql.telemetry.mozilla.org/

Public

Questions?

Public

Experiments

Targeted, restartless addons

Currently only available on desktop Firefox

Must undergo Data Collection Review; may require privacy/security reviews as well

Product approval required

https://wiki.mozilla.org/Telemetry/Experiments

Public

Flags for Testing

```
experiments.force-sample-value = "0.0"

experiments.logging.level = 0

experiments.manifest.cert.checkAttributes =
false

experiments.manifest.uri = "http://localhost:
8000/firefox-manifest.json

xpinstall.signatures.required = false
```

Public

```
let {classes: Cc, interfaces: Ci, utils: Cu} = Components;

Cu.import("resource:///modules/experiments/Experiments.jsm");
Cu.import("resource://gre/modules/Task.jsm");
Cu.import("resource://gre/modules/Preferences.jsm");

var gStarted = false;

const kSELF_ID = "flash-protectedmode-beta35@experiments.mozilla.org";

function startup() {
  // Seems startup() function is launched twice after install, we're
  // unsure why so far. We only want it to run once.
  if (gStarted) {
    return;
  }
  gStarted = true;

  Task.spawn(function*() {
    let branch = yield Experiments.instance().getExperimentBranch(kSELF_ID);
    switch (branch) {
      case "control":
        return;
      case null:
        let r = (Math.random() >= 0.5);
        if (!r) {
          yield Experiments.instance().setExperimentBranch(kSELF_ID, "control");
          return;
        }
        yield Experiments.instance().setExperimentBranch(kSELF_ID, "experiment");
        // FALL THROUGH
      case "experiment":
        let defaultPrefs = new Preferences({defaultBranch: true});
        defaultPrefs.set("dom.ipc.plugins.flash.disable-protected-mode", true);
        return;
      default:
        throw new Error("Unexpected experiment branch: " + branch);
    }
  }).then(
    function() {
    },
    function(e) {
      Cu.reportError("Got error during bootstrap startup: " + e);
    });
}

function shutdown() {
  let defaultPrefs = new Preferences({defaultBranch: true});
  defaultPrefs.set("dom.ipc.plugins.flash.disable-protected-mode", false);
}
```

https://bugzilla.mozilla.org/show_bug.cgi?id=1110215

Public

```
{
  "publish"      : true,
  "priority"     : 2,
  "name"         : "Flash Protected-Mode Testing",
  "description"  : "Measuring the effect of Flash protected mode on crashes, hangs, and other browser jank.",
  "info"         : "<p><a href=\"https://bugzilla.mozilla.org/show_bug.cgi?id=1110215\">Related bug</a></p>",
  "manifest"     : {
    "id"                : "flash-protectedmode-beta35@experiments.mozilla.org",
    "startTime"         : 1418601600,
    "endTime"           : 1421280000,
    "maxActiveSeconds"  : 2764800,
    "appName"           : ["Firefox"],
    "channel"           : ["beta"],
    "os"                : ["WINNT"],
    "minVersion"        : "35.0",
    "minBuildID"        : "20141215000000",
    "maxVersion"        : "37.*",
    "sample"            : 0.1,
    "disabled"          : true
  }
}
```


http://hg.mozilla.org/webtools/telemetry-experiment-server/file/tip/experiments/flash-protectedmode-beta/manifest.json

Bug 1111791

“Telemetry report: effect of the Flash protected-mode experiment”

https://bugzilla.mozilla.org/show_bug.cgi?id=1111791

Public

References

https://wiki.mozilla.org/Telemetry/Experiments

https://developer.mozilla.org/en-US/Add-ons/Bootstrapped_extensions

https://wiki.mozilla.org/QA/Telemetry#Telemetry_Experiments.2FFHR_Documentation

http://codefirefox.com/video/install-telmetry-experiment

http://hg.mozilla.org/webtools/telemetry-experiment-server/file/tip/experiments

https://bugzilla.mozilla.org/show_bug.cgi?id=1110215

https://bugzilla.mozilla.org/show_bug.cgi?id=1111791

Public

Questions?

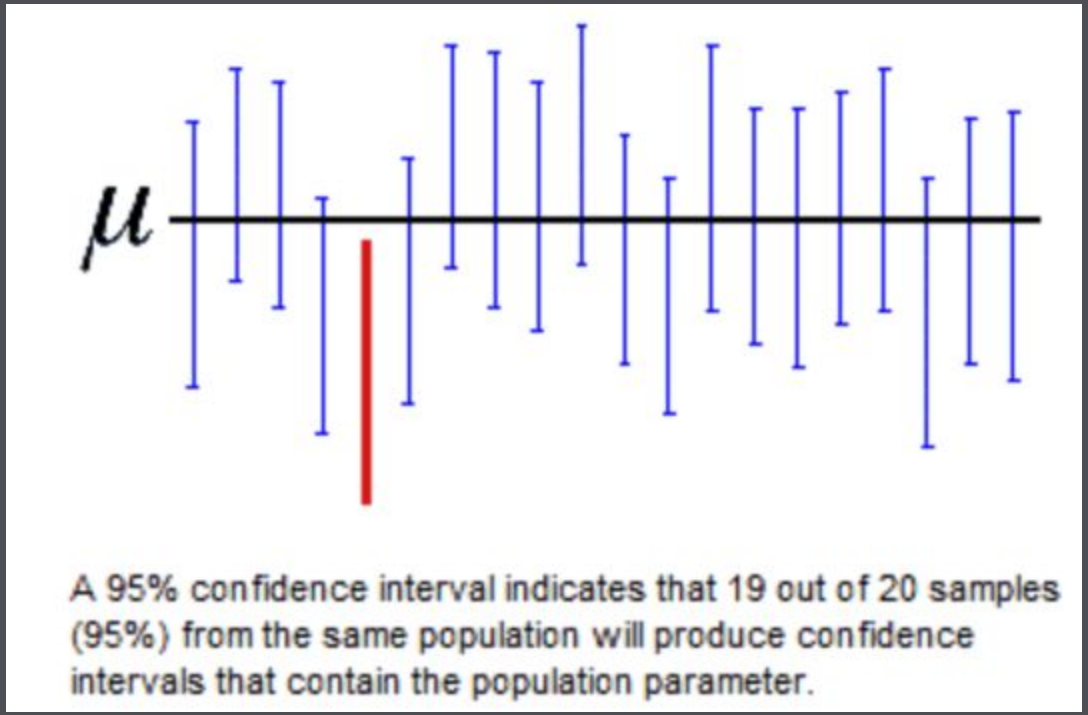
Public

Stats

Public

Use Representative Samples

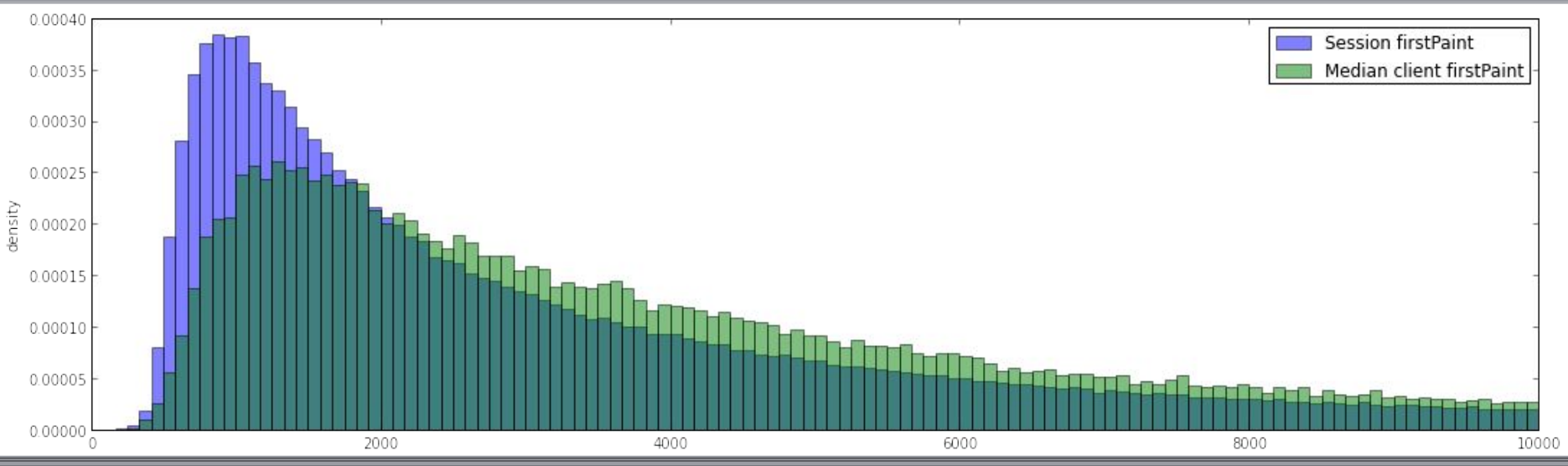
Use Sufficient Data



https://robertovitillo.com/2015/03/15/confidence-intervals-and-hypothesis-tests-for-engineers/

Public

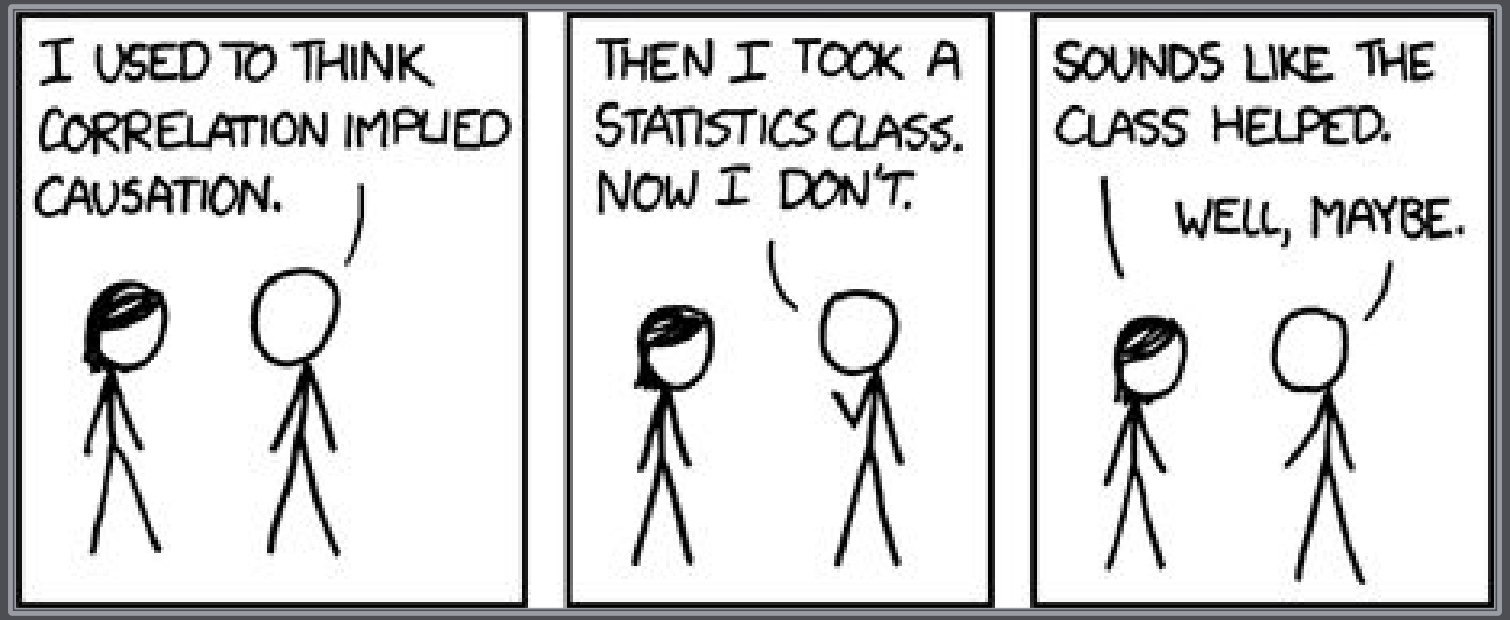
Beware of Pseudoreplication



http://robertovitillo.com/2014/12/19/clientid-in-telemetry-submissions/

Public

Correlation is not Causation



Public

Use the Correct Average

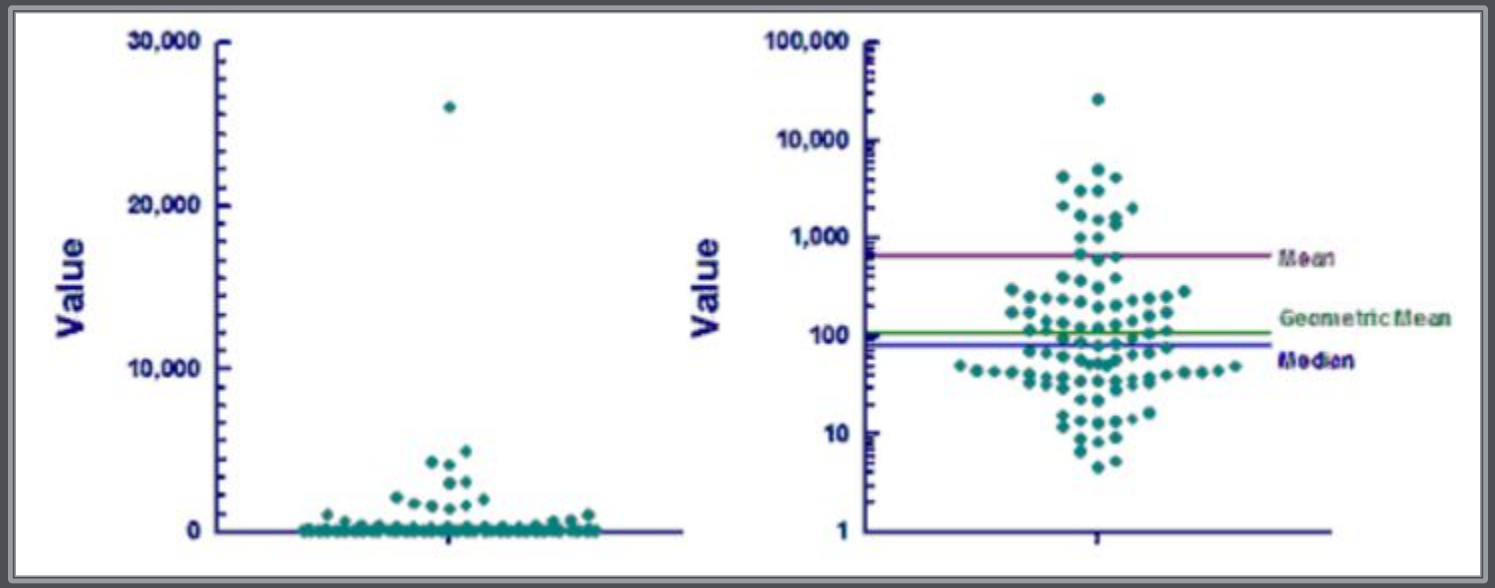
Arithmetic Mean: add up N values, divide by N

Median: sort the values and pick the middle one

Public

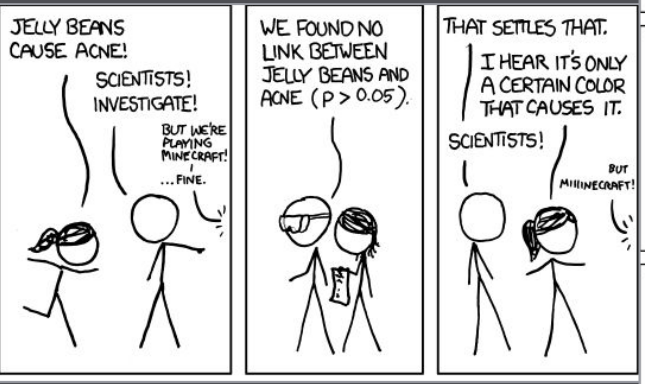
Use the Correct Average

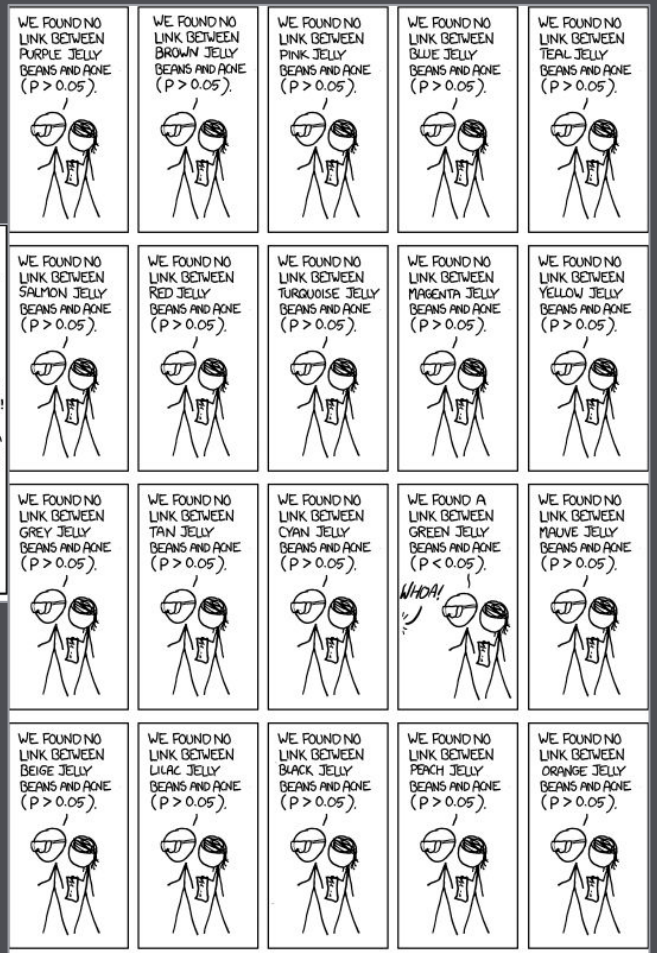
Geometric Mean: take the Nth root of the product of the N values

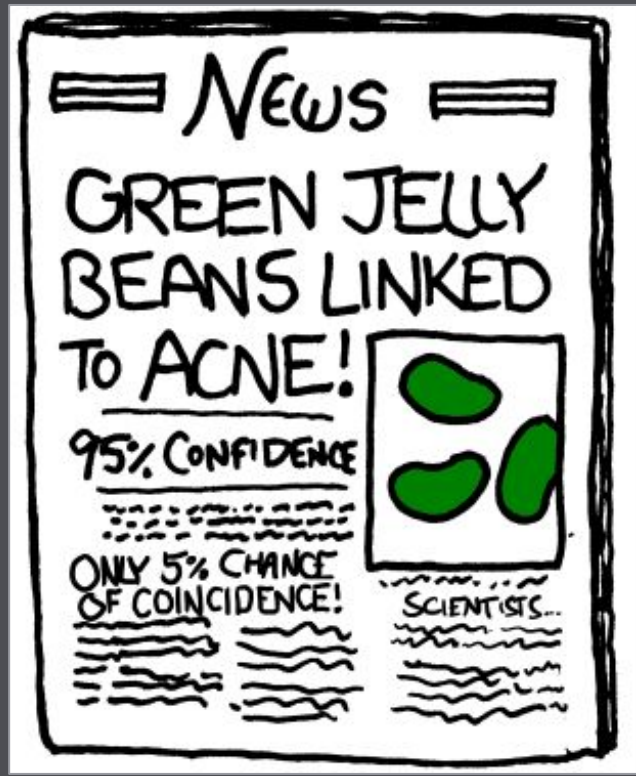


Public

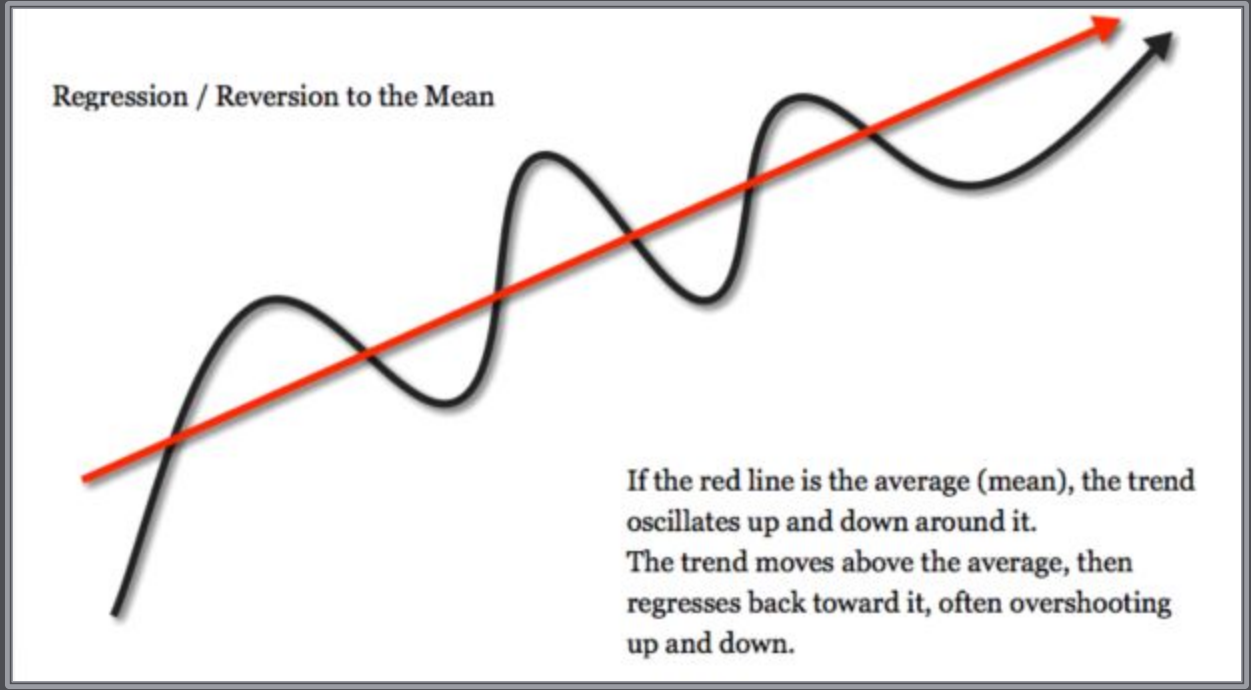
Control the False Discovery Rate







Regression Towards the Mean



Public

Reproducible Analyses

mozilla

```
In [1]: import ujson as json
        import matplotlib.pyplot as plt
        import pandas as pd
        import numpy as np
        import plotly.plotly as py

        from moztelemetry import get_pings, get_pings_properties, get_one_ping_per_client, get_clients_history

        %pylab inline

        Populating the interactive namespace from numpy and matplotlib

In [7]: pings = get_pings(sc, app="Firefox", channel="release", submission_date="20150928", build_id="20150917150946")

In [8]: def telemetry_enabled(ping):
            return ping.get("environment", {}).get("settings", {}).get("telemetryEnabled", False)

In [*]: pings.count()

Out[9]: 1242836
```

Public

References

http://www.statisticsdonewrong.com/data-analysis.html

http://www.amazon.com/How-Lie-Statistics-Darrell-Huff/dp/0393310728

http://www.slideshare.net/RobertoAgostinoVitil/all-you-need-to-know-about-statistics

Public

Questions?

Public

Contact

Don’t get frustrated, get help.

IRC: #telemetry #datapipeline

Teams: Measurement, Metrics, Data Engineering

FHR-dev: https://mail.mozilla.org/listinfo/fhr-dev
FX-Data-Platform: https://groups.google.com/a/mozilla.com/forum/#!forum/fx-data-platform

Public

Fin

Public