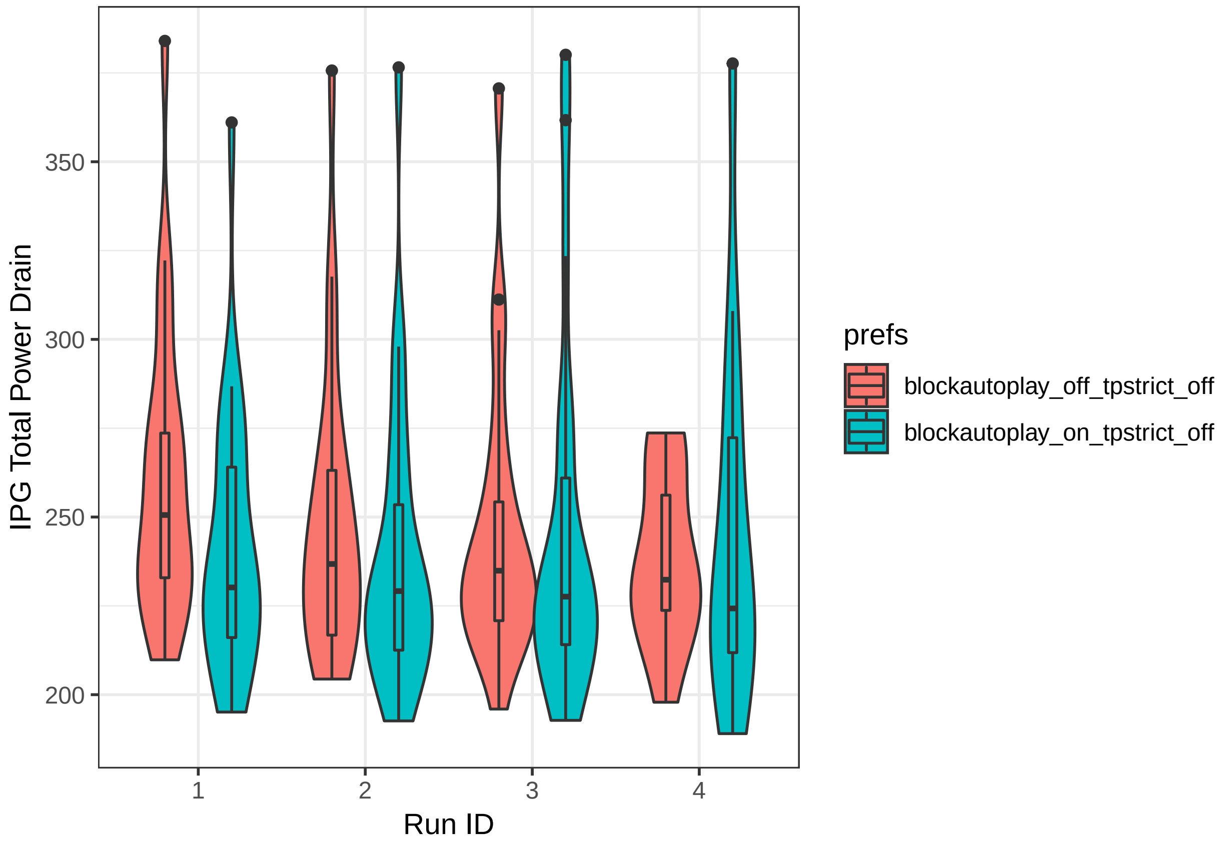
Block Autoplay: Comparison of Battery Drain

Corey Dow-Hygelund, Mozilla Data Science

A suite of experiments were performed, as previously described, with both the block autoplay on and off to determine its influence on power usage/battery consumption. The current experimentation process utilizes *Intel Power Gadget* to measure battery consumption using `Processor Energy` as a proxy.

Full experiment

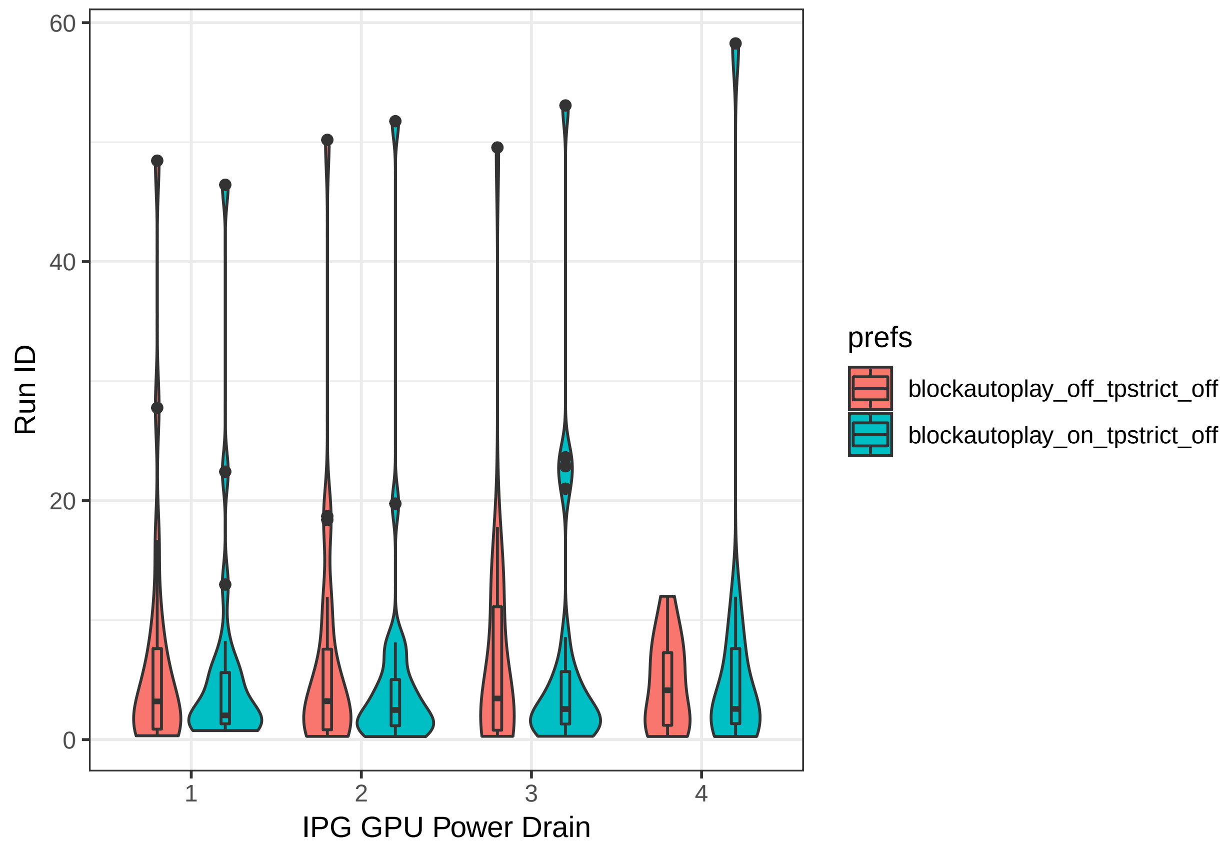


The missing high outlier for run #4 is due to an incomplete final Lingcars run (battery ran out). There is the expected drop in the block autoplay power consumption. Another interesting effect, is the significantly higher run #1 power drain values for block autoplay off, which drops for run #2, and even more to #3. This is also true, to a lesser degree, for block autoplay on for run #1.

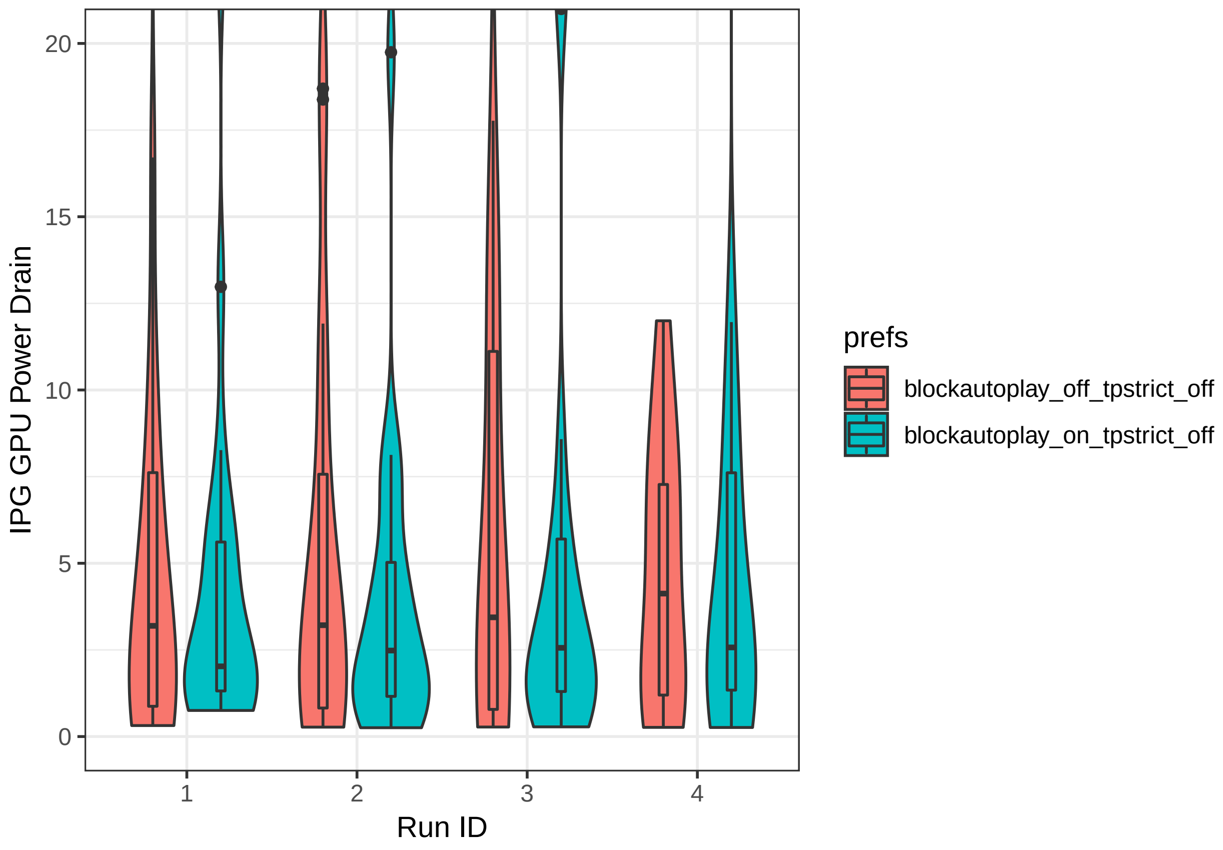
Question: Is this an effect due to the drain rate being dependent upon charge? Or is it solely due to caching?

- **TODO**: Reform experiment with mutiple charges due to determine the influence of caching versus charge capacity.

GPU should be changing significantly between the pref-flip:

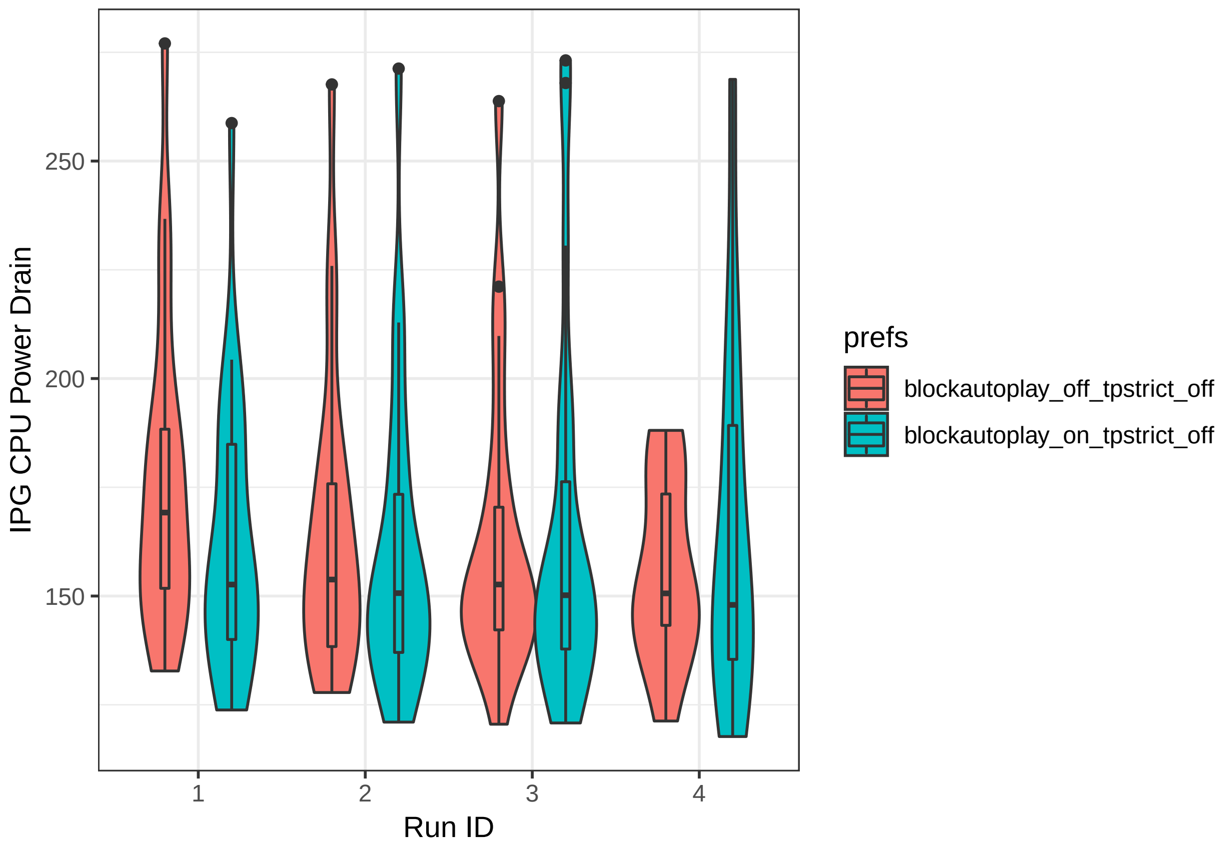


The spread and quantiles are greater for block autoplay off. Zooming in



The higher values in run #1 and #2 is not observed for GPU usage. In fact for block autoplay on #1 is lower.

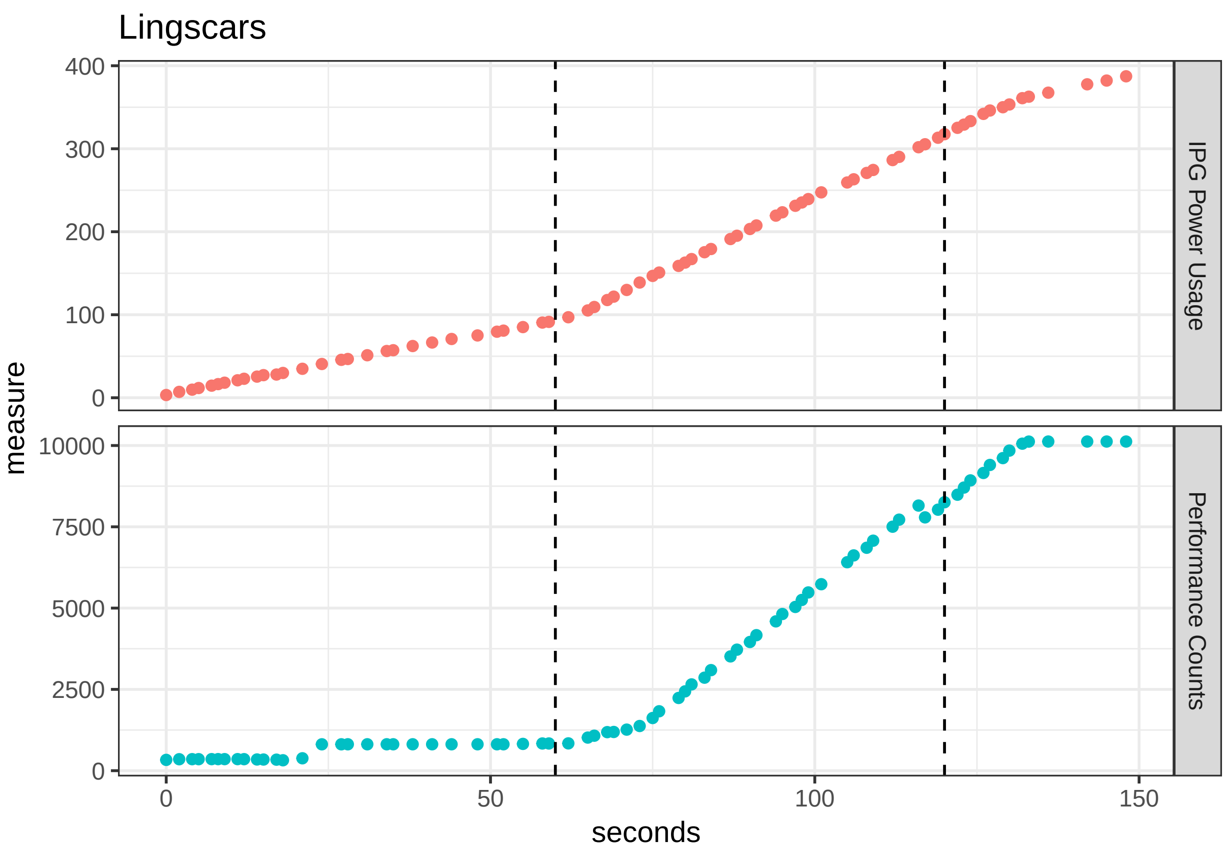
Observing the CPU/cores power usage:



The increased usage in run 1-2 is due to CPU usage. It is interesting to note the block autoplay influences *both* the CPU and GPU power drain.

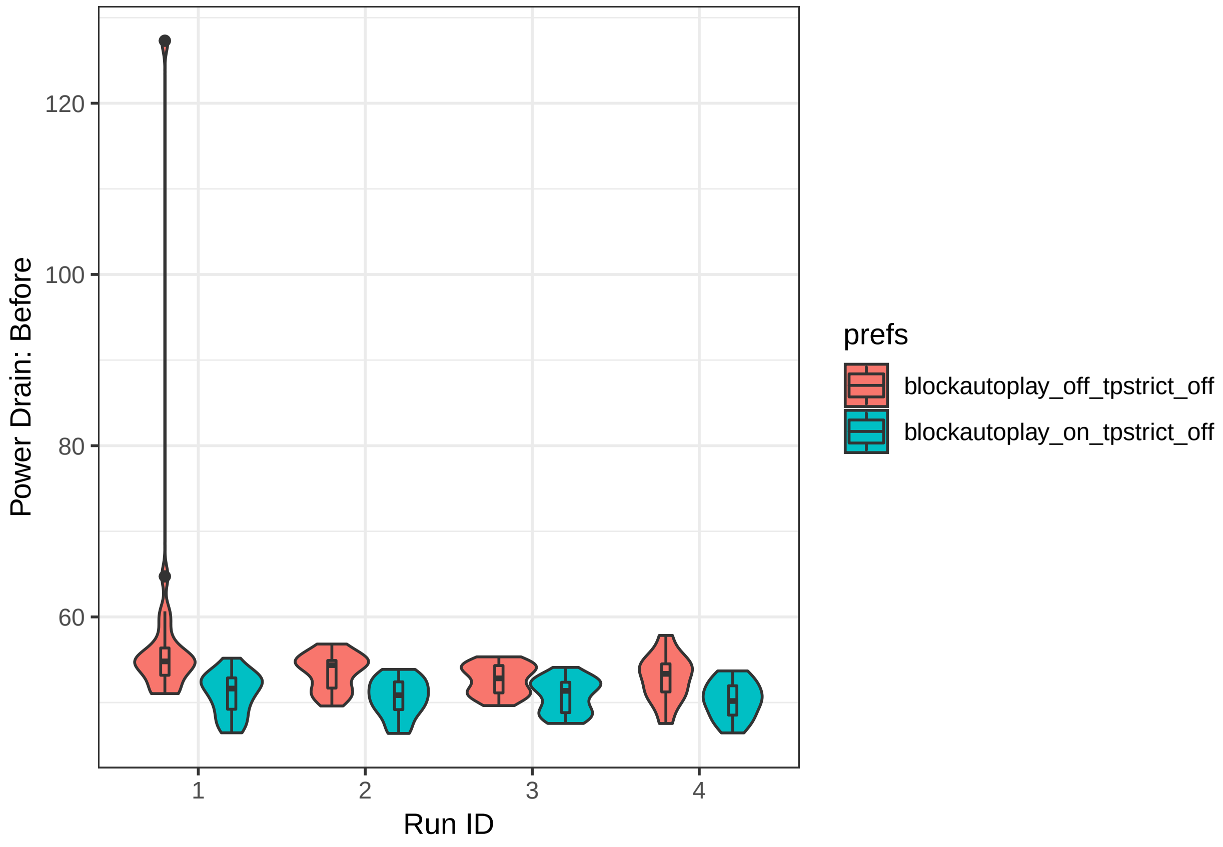
Before/After/During URL Navigation

It has been observed that the battery level might have an influence on the rate of battery drain. Moreover, there is the possibility of run order also influencing this rate. One item to investigate is the rate of power usage before, during, and after navigation to the URL. An informative plot:



Navigation to Lingscars occurs between two dotted line, with the higher rate of power usage.

Before

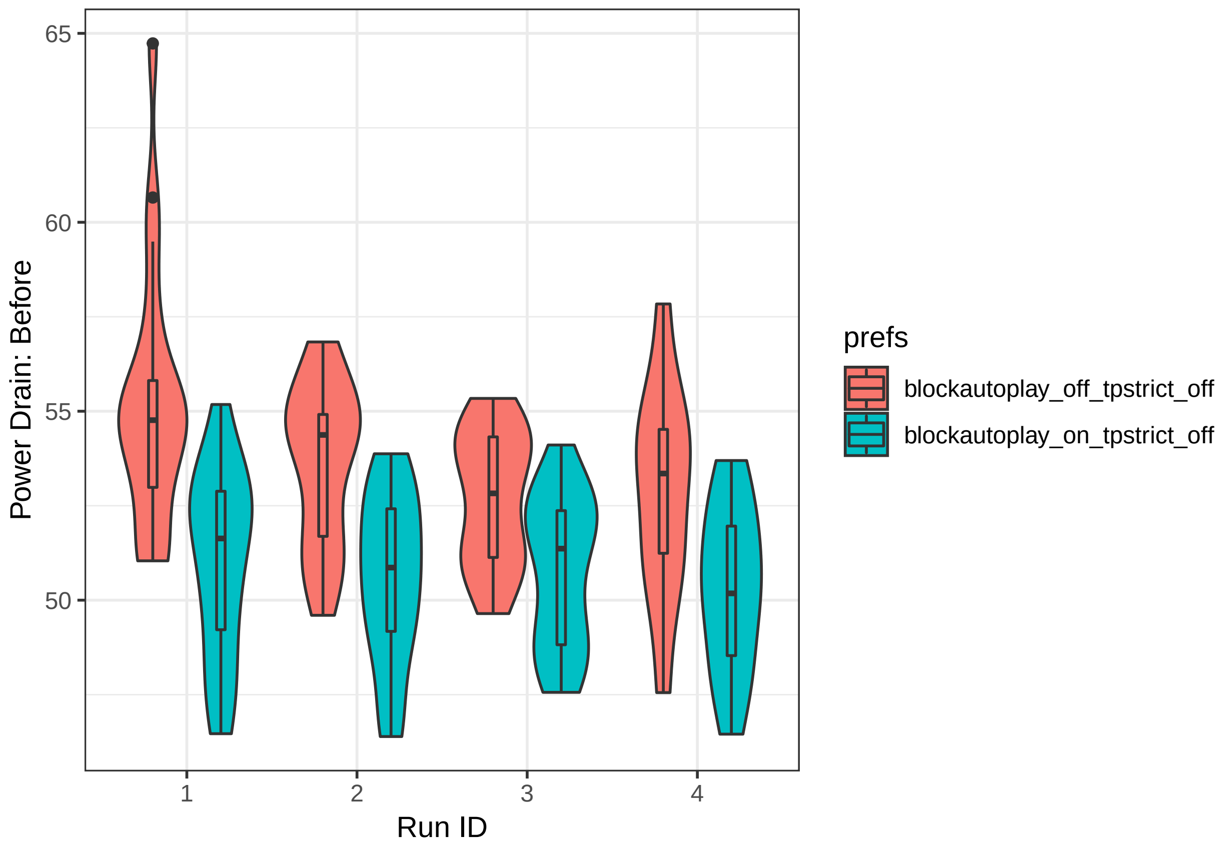


What is the the huge outlier:

```
kable(nav_diff[nav_diff$before >120, ], format='latex')
```

run_id	before	during	after	exp	prefs
1	127.303	68.785	23.332	exp_google	blockautoplay_off_tpstrict_off

Popping out this record and replotting:



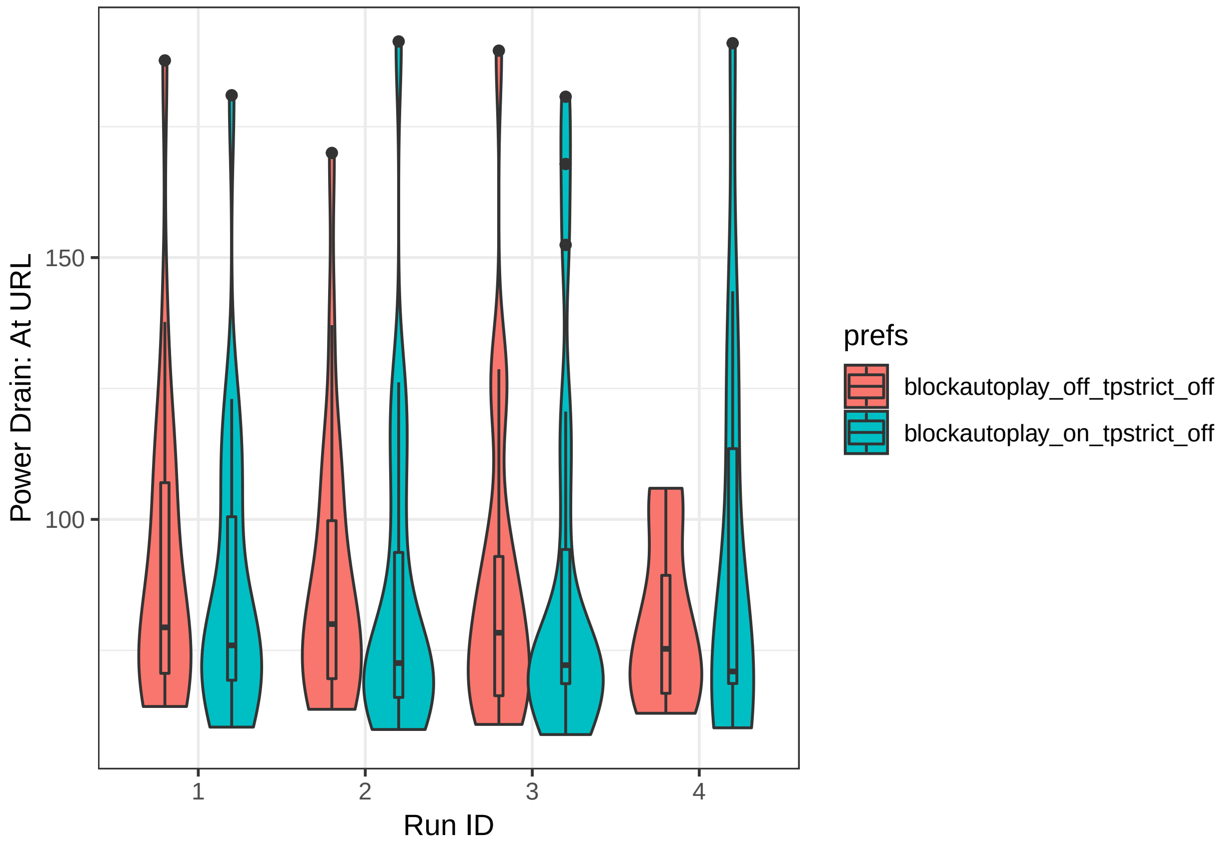
The #1 run for both prefs is higher than the remaining run, but is especially significant for block autoplay off. This is what is resulting in the large difference in observed total power conusmption.

Observation: What is interesting is that there is a difference between the prefs right after Firefox has been started, *before* navigation to the experiment URL has occurred. This is consistent across all experiment runs!

TODO: Another experiment needs to be performed to determine the robustness of this result. (see above)

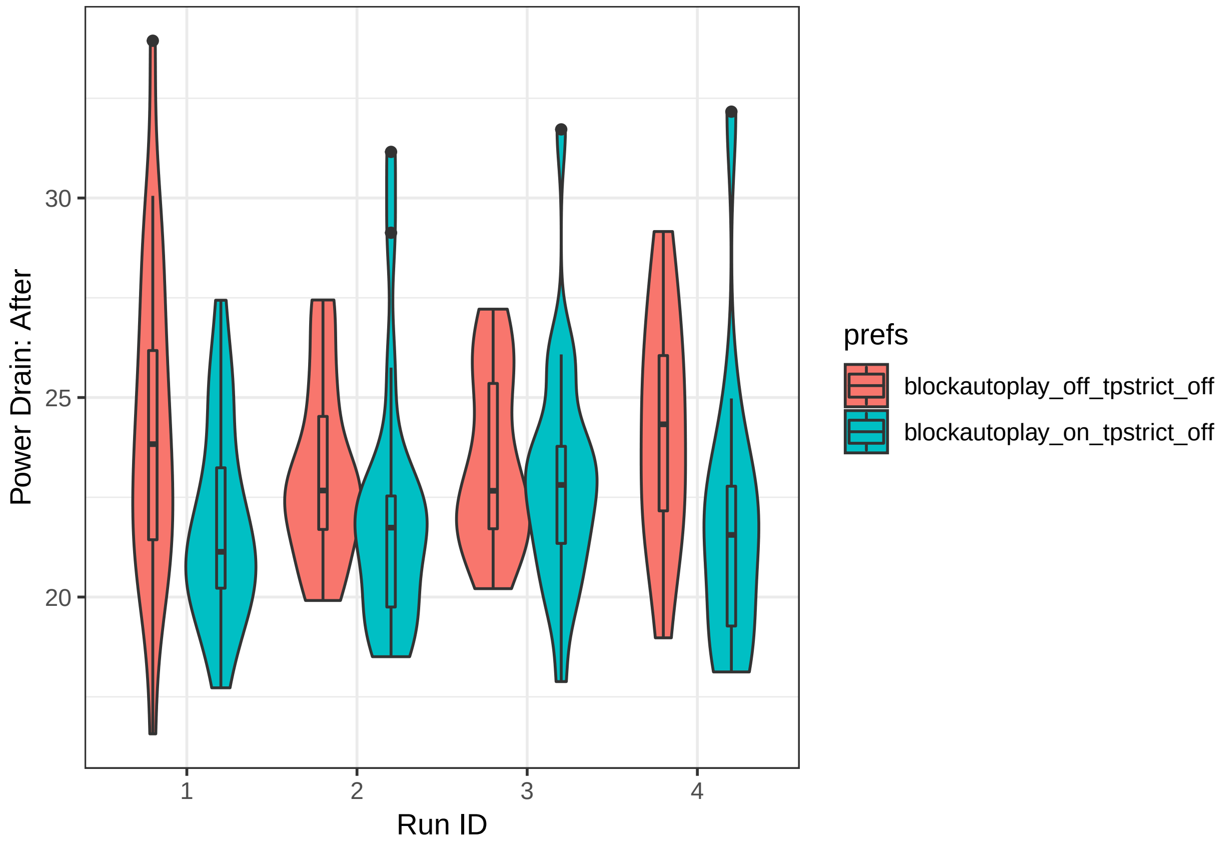
- Perform experiments with pref-flipping between each run, rather than running a full suite with pref frozen

During



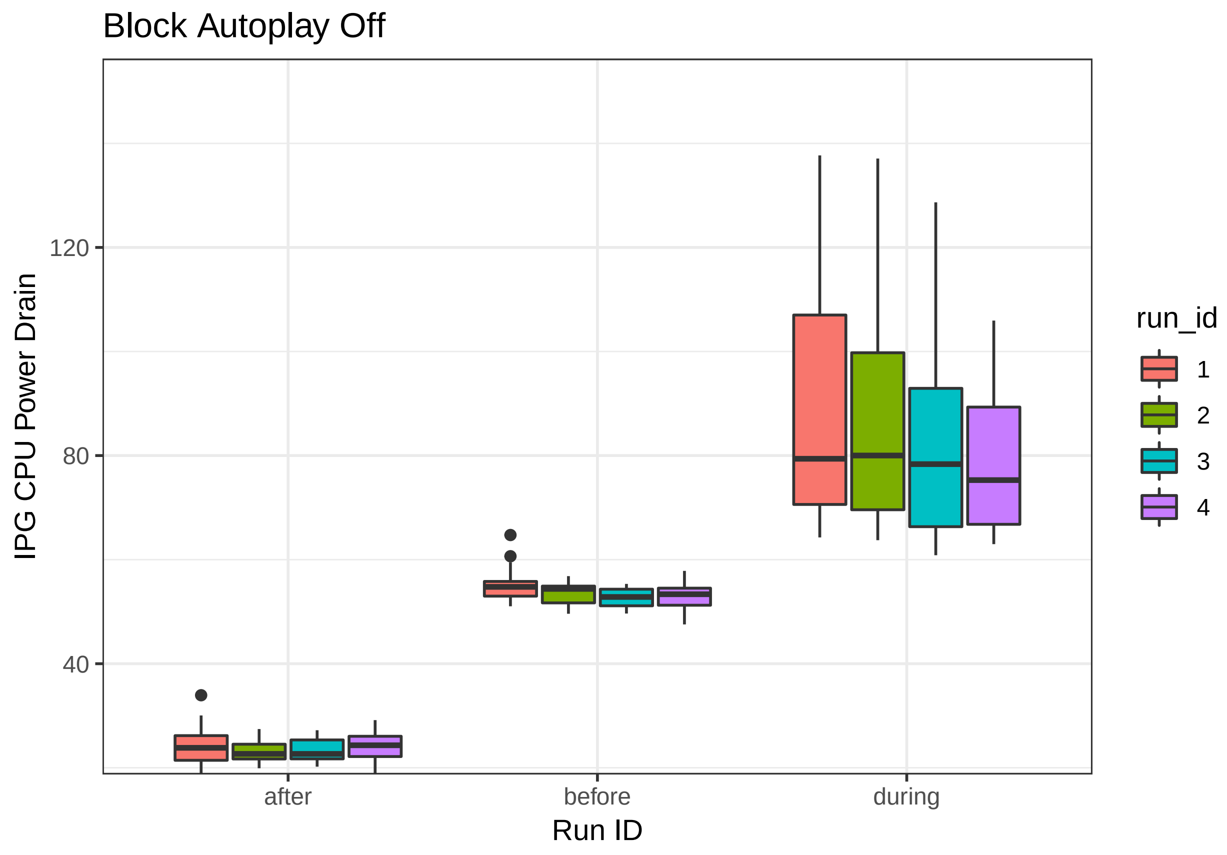
As expected, increase power usage while at the URL.

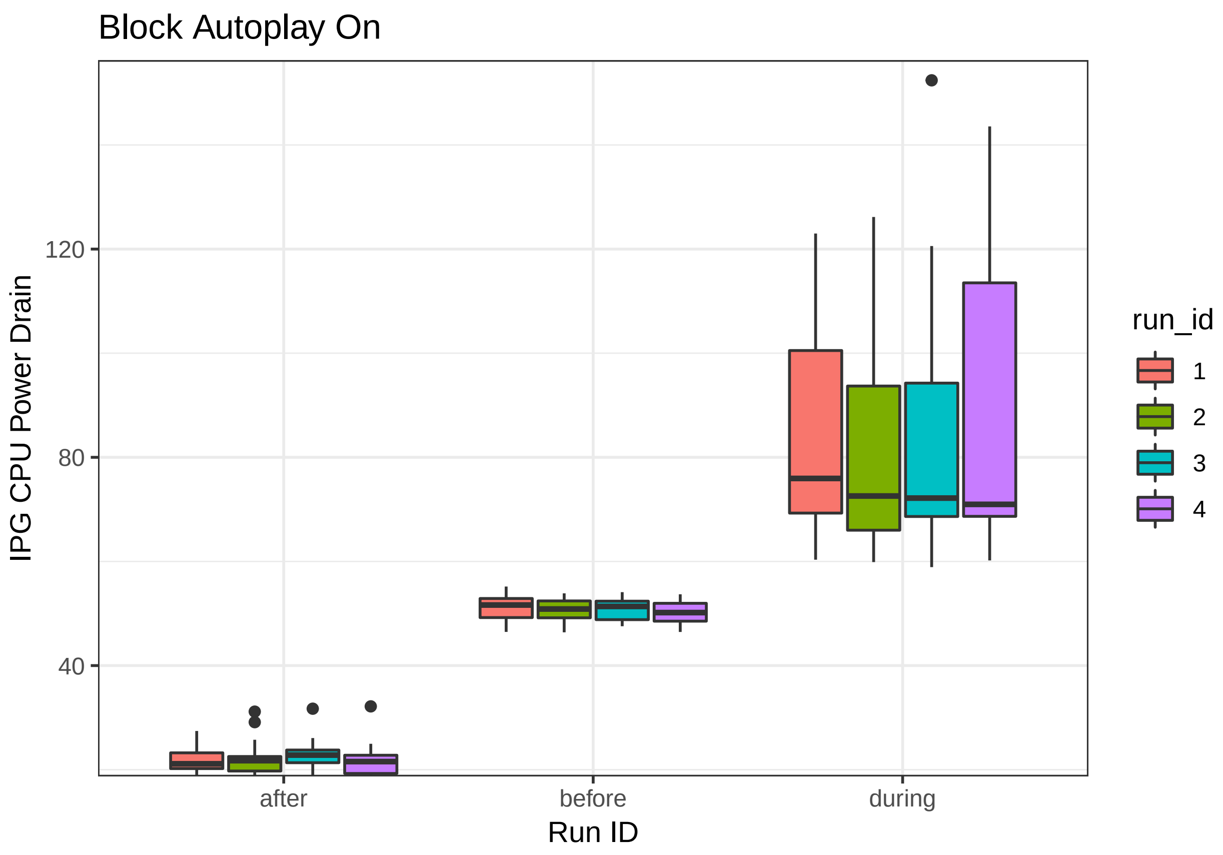
After



The first run for block autoplay off is again high. There is a lot of variation between the prefs, with the observed increase in block autoplay off in runs 1,2 and 4, but absent in 3.

By Pref


Block Autoplay Off
IPG CPU Power Drain
120
80
40
after
before
during
Run ID
run_id
1
2
3
4



The first run for both prefs does have higher distribution values for the initial start-up to URL navigation. Again, this needs to be investigated with further experiments.