Investigating the Correlation between Perfume.js Performance Metrics and Energy Efficiency of Web Applications

Maggie Mackenzie-Cardy
2673451
VU Amsterdam
m.mackenziecardy@student.vu.nl

Corneliu-Dumitru Soficu
2675454
VU Amsterdam
c.soficu@student.vu.nl

Andrei-Ioan Bolos
2701860
VU Amsterdam
a.i.bolos@student.vu.nl

Yuxue Liu
2659145
VU Amsterdam
yuxue.liu@student.vu.nl

Dinga Mădălina
2682178
VU Amsterdam
m.dinga@student.vu.nl

ABSTRACT

Context - The limitation of battery life of mobile devices and the increase in energy costs have put a spotlight on the development of energy-efficient software products lately. The main goal of developers was to have an extremely responsive product, without even considering that energy efficiency plays an important role in the overall performance of the application. There are many performance profiling tools for the developers to measure performance, but when it comes to energy efficiency however, these tools seem to be lacking.

Goal - The purpose of this research paper is to analyze the correlation between performance and energy efficiency of web application running on mobile devices, at load time. By proving this, developers could get an insight into how to build energy-efficient applications and thus waste less energy over time.

Method. We performed an empirical experiment in an controlled environment where 30 web apps were randomly chosen. These web apps were grouped in three different performance treatments: high, average and low. In order to cut off the effects of network condition, the experiment was further split into two different blocks: 3G and Wi-Fi. In order to assess if energy efficiency is related to fixed performance treatment, we collected Perfume.js data using a mobile Android device. Furthermore, the resulting data is normalized and grouped by category. Finally, we quantified the scores for each performance category by transforming the ratio measures into ordinal estimations.

Results. The test results show that in case of Wi-Fi network, it is difficult to draw conclusions based on QQ-plot. Shapiro-Wilk normality test resulted in a p-value of 0.07, hence the affirmation that energy data is normally distributed could not be rejected. We analyzed the square, the natural logarithm and the reciprocal of energy efficiency in attempt to reduce skewness. There was a reduction from 0.65 to -0.22 for the natural lograithm and 0.42 for reciprocal. As a result, we considered the reciprocal of energy efficiency when performing statistical tests to test the hypotheses. As far as 3G was concerned, in two of our performance score diagrams, the data distributions were opposite. We also performed Shapiro-Wilk normality test, obtaining a p-value of 0.08, meaning that also we could not reject the null hypothesis that energy efficiency data is normally distributed. In order to reduce skewness, we analysed again the reciprocal, the square and the natural logarithm of energy efficiency. We got the values of skewness of 0.43, -0.03 and 0.17 in the square, the natural logarithm and the reciprocal of energy efficiency respectively. As a result, we considered again the reciprocal when performing statistical tests to test the hypotheses.

Conclusions. The main contribution of our report shows that there is no significant statistical correlation between performance and energy efficiency over a Wi-Fi network, but using a 3G connection there is such a correlation indicating that slower load speed and navigation timing leads to worse energy efficiency on android based system. However, this general correlation alone would not be efficient enough to motivate developers write code in a green way. Therefore, we would like focus on more factors to become more specific. For example, more network technologies could be taken into consideration, such as 4G or 5G. In addition, newer phone technologies could have an impact on performance and energy efficiency, thus we would use different low-end and high-end devices. Finally, software technologies could influence our experiment, so altering selection to websites written in a wider variety of programming languages could be useful.

1 INTRODUCTION

Rising energy costs and the desire to reduce the carbon footprint of digital technology have determined a strong interest in the design and implementation of 'green', energy-efficient software products lately [1]. This requires developers to understand the factors and processes that affect energy consumption in order to be able to optimize the amount of energy required by applications to perform certain tasks.

As mentioned in one of the very few energy-efficient software development guides: 'Even small inefficiencies in apps add up across the system, significantly affecting battery life, performance, responsiveness, and temperature'[2], energy efficiency is an increasingly important property of software. For instance, the more energy efficient software is, the less batteries will be exploited and worn out in time. Therefore, it is essential to take energy-efficiency into account when developing software applications. Nevertheless, defining whether software is energy efficient or not is difficult. Developers can know how much energy has been consumed but they may not know what is the optimum consumption[3]. To solve this problem, the international and regional experts addressed a discussion to develop definitive quantitative metrics for energy efficiency. Efficiency metrics can provide developers an additional tool to optimise the energy consumption [4]. In addition, correlating energy efficiency with other software properties such as performance, could guide developers into effectively optimizing their software from an energy-efficiency perspective. Progress in measuring performance characteristics has already been made, as an attempt to make web applications faster, which makes performance more practical to measure than energy.

There are several bench-marking tools that can monitor web performance. In this experiment, *Perfume.js*[1], a JavaScript-based web performance monitoring library, developed by Leonardo Zizzamia, is used to measure several loading and interaction indicators cross-browser. It is highly precise, by using the latest browser Performance APIs and it is capable of reporting field data to third-party analytics tools.

The research goal of this paper is to investigate the correlation between performance and energy efficiency of web applications running on mobile devices, at load time. If proven, the correlation between the performance metrics and energy efficiency can give developers a better understanding of how to build energy-efficient applications, by studying and tuning user-centric performance metrics, by means of *Perfume.js*. As a result, providing users with a desirable experience could also contribute to improving the energy efficiency of web applications.

2 RELATED WORK

This subsection presents an overview on existing studies focused on the correlation between performance and energy efficiency.

Ivano Malavolta et al. [5] perform an investigation similar to the one we propose, where a correlation between performance metrics gathered by Google Lighthouse [6] and the energy efficiency on an Android device running 21 different web apps was found. This correlation showed that better results in performance metrics correlated with a decrease in energy consumption on the device. This gives a validity to our experiment, since we will be expanding upon these results. Our experiment differs from this one in a number of ways. Firstly, it differs in terms of the technical aspects. We will be using Perfume.js to gather performance metrics instead of Google Lighthouse, and Android Runner to collect data from the smartphone instead of Greenspector [7]. Furthermore we integrate additional factors such as a high-end Android device vs a low-end one, and the speed of the network when conducting our experiments. Finally, our experiment is much more fine grained since we are analysing the specific metrics that make up the final performance score of the Perfume.js analytics, while this paper only took the aggregated Lighthouse score into account.

The relationship between Android energy efficiency and the performance of apps is further explored by X. Chen et al [8]. The authors seek to measure in detail the power consumption of different Android applications, investigate the effect of using different programming languages and implementation (e.g. recursion vs iteration) on power consumption and consequently conclude whether the design of these applications can lead to better energy efficiency on the Android device. Our investigation differs from this one in its emphasis - while the paper approaches the problem from the point of view of the design of the applications and the consequent efficiency, we first seek to establish a relationship between power consumption and efficiency and investigate other environmental factors such as the network speed and how modern the Android device is when investigating this relationship.

L. Ying-Dar et al. [9] perform time profiling for a web browser under 3 scenarios: booting, browsing and streaming. For the browsing scenario, which is the most important scenario for us, the experiment consisted of loading a 2128 KB webpage with eight attached images. This was done using three different networking technologies: Wi-Fi, 3G and 2.75G. They concluded that the browsing performance with 2.75G and Wi-Fi was the slowest and the fastest among the three, respectively. The results also show that downloading element resources consumes 90% of the browsing time, regardless of the network the device under test connects to. Our experiments are different than the ones discussed in the paper as we are looking at various other metrics and also how they relate to power consumption.

A systematic and comprehensive system for profiling apps was developed in the "ProfileDroid: Multi-layer Profiling of Android Applications" paper[10]. This system has the goal of measuring and profiling Android apps at four different levels: static, user interaction, operating system and network. The authors profiled 27 apps for metrics that include: percentage of traffic that is encrypted through HTTPS, traffic intensity or the ratio of incoming traffic and outgoing traffic. These metrics are relevant to our experiment and can also help us create a better image of the evolution of the Android ecosystem after multiple years. Our experiments differ from the ones in this paper as we are analysing energy efficiency and comparing it with Perfume.js metrics.

Finally, this paper [11] establishes a correlation between the performance of applications and the energy efficiency of an Android device by investigating how the use different languages affect the above. The conclusion of the paper was that JavaScript by far outperformed other languages such as Java in terms of both energy efficiency and performance. Although this paper differs to our investigation in terms of the factors being measured, it is similar in its methodology and the parallel drawn between energy efficiency and improved performance further gives validity to our research. However our investigation is not focused on low level factors such as which language the website was written in when considering the performance metrics, but rather which metrics can influence energy efficiency the most (if they can influence this at all).

[1]https://zizzamia.github.io/perfume/

3 EXPERIMENT DEFINITION

3.1 Goal Definition

The goal of this experiment is presented in Table 1 using the Goal-Question-Metric approach (GQM) [12].

Analyse	Perfume.js performance metrics
for the purpose of	Evaluation
with respect to their	Energy Efficiency
from the point of view of	Software Developers
in the context of	Web Applications
GQM Goal	
Analyse Perfume.js performance metrics, for the purpose of evaluation, with respect to their energy efficiency, from the point of view of software developers, in the context of web applications.	

Table 1: Goal Definition

3.2 Research Questions

This subsection outlines the main research questions, which define the aforementioned goal:

[RQ1] *To what extent does performance correlate with energy efficiency of web applications at first load?*

This research considers several web performance monitoring criteria, such as navigation timing, load speed and responsiveness. Studying the correlation between these performance factors and energy efficiency is highly relevant from the software development perspective, as this:

- proves the influence of web performance on energy efficiency
- informs developers about the bottlenecks affecting performance, and, by correlation, energy efficiency, at first load of the application
- guides developers into adjusting relevant performance metrics in order to improve energy efficiency

In this sense, a separation is made among several *Perfume.js* indicators, which are relevant during the first load of the application. First, the navigation timing indicators, also used by *Perfume.js*, is analysed with respect to network requests. Then, load speed and responsiveness are studied apart, as they represent critical, but distinct phases in application loading. The way in which they affect energy efficiency might be very different (positive, negative or no correlation). The following sub-questions are addressed accordingly:

[RQ1-1] *To what extent does* ***navigation timing*** *affect energy efficiency of web applications at first load?*

Timing information related to navigation, from when loading site has been triggered to the moment the page has fully loaded can offer valuable insights by correlating the diverse indicators with energy efficiency.

[RQ1-2] *To what extent does* ***load speed*** *affect energy efficiency of web applications at first load?*

Proving the effect of load speed on energy efficiency would allow for accessible optimization, by guiding developers into fine-tuning paint timing.

3.3 Metrics

To answer the research questions, it is essential to consider objective and quantifiable metrics related to performance and energy efficiency:

- Navigation timing - performance metrics for the life and timings of a network request:
  - Header size: HTTP header size
  - Fetch time: Cache seek plus response time
  - Total time (ms): Request plus response time (network only)
  - Download time (ms): Response time only (download)
  - Time to First Byte - TTFB (ms): The amount of time it takes for the server to send the first payload to the client
- Load speed - performance metrics for the time spent to load and render all of the visual elements to the screen:
  - First Paint - FP (ms): the exact time taken for the browser to render anything as visually different from what was on the screen before navigation
  - First Contentful Paint - FCP (ms): the time from when the page starts loading to when the first bit of meaningful content is rendered
  - Total Load Time - the time taken for the entire page to be loaded (the time taken until the onload callback is called)
- Energy efficiency (Joules): the energy consumed for loading the application

3.4 GQM Tree

Figure 1 presents the visual tree representation corresponding to the aforementioned GQM model.

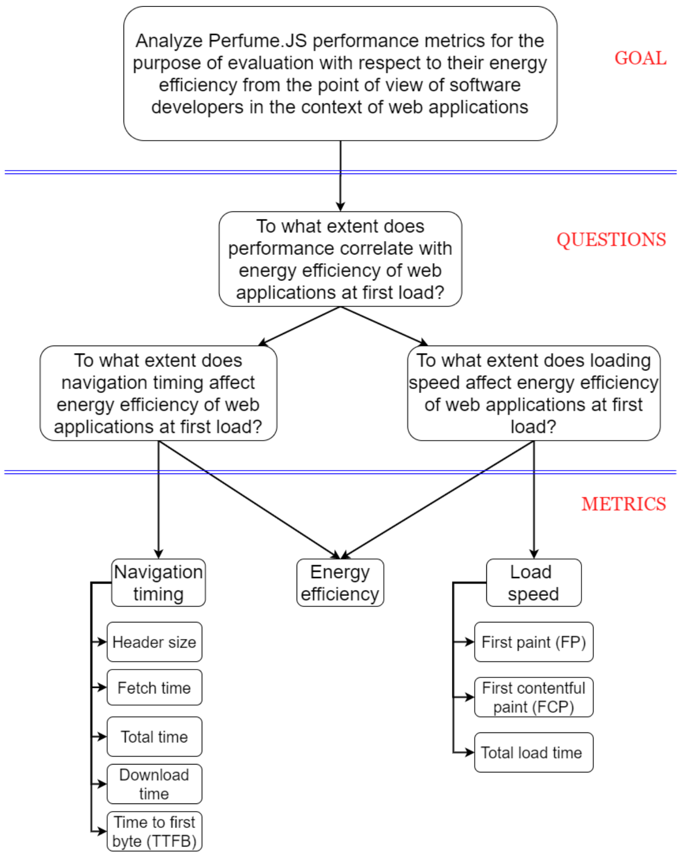

Analyze Perfume.JS performance metrics for the purpose of evaluation with respect to their energy efficiency from the point of view of software developers in the context of web applications
GOAL
To what extent does performance correlate with energy efficiency of web applications at first load?
QUESTIONS
To what extent does navigation timing affect energy efficiency of web applications at first load?
To what extent does loading speed affect energy efficiency of web applications at first load?
METRICS
Navigation timing
Energy efficiency
Load speed
Header size
Fetch time
Total time
Download time
Time to first byte (TTFB)
First paint (FP)
First contentful paint (FCP)
Total load time

Figure 1: GQM Tree

4 EXPERIMENT PLANNING

4.1 Experimental Hypotheses

In this section we formally address the research questions by formulating the hypotheses. We aim to assess whether the administration of the treatments (i.e., high, average, low) on each of the performance categories (i.e., navigation timing and load speed) impacts energy efficiency at first load of the application.

For **RQ1-1**, concerning **navigation levels**, the hypotheses are:

- $H_0 : \mu1_{high} = \mu1_{average} = \mu1_{low}$
  i.e.,: The mean energy efficiency does not significantly differ among web apps having different navigation timing levels
- $H_a : \mu1_{high} \neq \mu1_{average} \vee \mu1_{average} \neq \mu_{low} \vee \mu1_{low} \neq \mu_{high}$
  i.e.,: The mean energy efficiency significantly differs among web apps having different levels of navigation timing for at least one pair of navigation timing levels.

Here $\mu1_{high}$, $\mu1_{average}$, $\mu1_{low}$ represents the mean energy efficiency of the measured web apps having low, average, or high navigation levels.

For **RQ1-2**, concerning **load speed**, the hypotheses are:

- $H_0 : \mu2_{high} = \mu2_{average} = \mu2_{low}$
  i.e.,: The mean energy efficiency does not significantly differ among web apps having different load speed levels
- $H_a : \mu2_{high} \neq \mu2_{average} \vee \mu2_{average} \neq \mu2_{low} \vee \mu2_{low} \neq \mu2_{high}$
  i.e.,: The mean energy efficiency significantly differs among web apps having different levels of load speed for at least one pair of load speed levels

Here $\mu2_{high}$, $\mu2_{average}$, $\mu2_{low}$ represents the mean energy efficiency of the measured web apps having low, average, or high load speed.

4.2 Subjects Selection

The subjects, web applications, are randomly sampled out of the Tranco list [13], containing the top 1 million web applications obtained by combining multiple rankings from different sources (e.g., Alexa Internet Top 1 Million, Cisco Umbrella Popularity List, Majestic Million, Quantcast Top Sites), over a thirty-day period. In the randomization process, the domains appearing multiple times with different extensions, such as *facebook.com* and *facebook.com.au*, are only considered once.

As part of the randomization process, 30 web applications are sampled. This number was chosen such that the time spent running the experiment remains feasible. The average duration for executing one run is 20s, to which we also add an additional 2 min, to make sure that hardware components are in an idle state at the beginning of every run. Also, for an acceptable precision of the experiment, we perform 11 repetitions of each trial. Considering the two network conditions - 3G or Wi-Fi, we conduct a total of 660 runs. Out of these 660 runs, for one single website, we had to increase the profiling time to 25 seconds instead of 20 seconds for the 3G network condition. This was needed in order to be able to catch the perfume.js metrics. In total, this results to approximately 25.6 hours of experiment time.

4.3 Experimental Variables

We consider the following **independent variables** for the experiment:

- **Performance** is the **main factor** of the experiment, analysed based on the aforementioned categories - navigation timing and load speed. After running the experiment, we collect several *Perfume.js* metrics, which we use as a proxy for performance. We first normalize the obtained scores and then average them grouped by category. The result is a composed score for each of the three performance categories, for the selected web applications. Normalization consists in rescaling the score values to range [0, 1], having observed the minimum and the maximum of the *Perfume.js* metrics, using the formula:
  $z_i = \frac{x_i - min(x)}{max(x) - min(x)}$, where $x = (x_1, ..., x_n)$ is the list of scores for a given performance category.
  Finally, in order to simplify the statistical analysis, we quantify the scores for each performance category by transforming the ratio measures into ordinal ones. To this end, we

define three levels for navigation timing and load speed, respectively, using two cutting points. We will obtain an equal number of subjects on each level (low, average and high).

- The **network condition** is treated as a blocking factor with two treatments (3G and Wi-Fi) in the experiment, as we do not want to analyse its interaction with the performance.
- The **device performance** is fixed in order to separate its effect from the main factor. We use a low-end Android device to run the experiment. The reason we chose a low-end device is that performance of software ran on limited resources hardware is essential, small differences can mean a lot and are visible even without measuring tools.
- The **browser** is also fixed. We use Google Chrome to run the experiment, as it is the most used browser according to Statista [2].

The **energy efficiency** is the **dependent variable**, affected by the main factor and the considered co-factors. It represents the energy consumed to load the web application first, measured as ratio.

4.4 Experiment Design

The experiment design determines the optimal set of trials the experiment shall have, in such a way that the effect of the treatments is visible and, at the same time, the experiment remains feasible.

Firstly, we aim to keep performance at the core of the experiment and go as deeper as possible in the interpretation of the results, while also considering network conditions as a factor that might influence energy efficiency. In order to mitigate its effect and to ensure an unbiased assignment, we use two separate blocks, 3G and Wi-Fi, which we will study separately.

The experiment is designed as a 1 factor - 3 treatments (i.e., high, average, low) experiment, over 2 blocks (i.e., 3G and Wi-Fi). We consider a balanced design, with exactly 10 web applications corresponding to each treatment. To this end, we randomly choose 30 different web application from the Tranco list and split the subjects among treatments, as shown in Table 2.

Block	High	Average	Low
Wi-Fi	a_{1_1}	$a_{1_{11}}$	$a_{1_{21}}$
	...	...	...
	$a_{1_{10}}$	$a_{1_{20}}$	$a_{1_{30}}$
3G	a_{2_1}	$a_{2_{11}}$	$a_{2_{21}}$
	...	...	...
	$a_{2_{10}}$	$a_{2_{20}}$	$a_{2_{30}}$

Table 2: Balanced Experimental Design For Performance Category

Every treatment contains $n = 10$ objects, denoted by a_{i_j}, where i is the belonging block, $i = \overline{1..2}$ and j is the index of the web application $j = \overline{1..10}$. The web application partitions for the Wi-Fi $\{a_{i_1}, a_{i_2}, ...a_{i_n}\}$ and 3G $\{a_{j_1}, a_{j_2}, ...a_{j_n}\}$ blocks might be different, depending on how performance is affected when running the experiment under different network conditions.

4.5 Data Analysis

The statistical tests are guided by the experiment design, which will shape the relationship between the energy efficiency and the selected performance metrics of the target web applications. Statistical tests play a crucial role in understanding whether the *Perfume.js* metrics have an impact on the energy efficiency and what are their levels of influence.

Provided that energy efficiency (continuous, dependent variable) is normally distributed between the independent groups, it is preferred to use parametric tests for testing the hypotheses. Specifically, for one factor (i.e., performance) with three treatments (i.e., high, average, low), **One-way ANOVA** can be used in the analysis to test the effect of each performance category (i.e., navigation timing, load speed) on energy efficiency. In this case, we also assume independent samples and residuals should be normally distributed. Also, variance between groups should be the same, which we test by applying Levene's test for homoscedasticity.

However, if the aforementioned assumptions are violated, we will use the one-way non-parametric alternative to ANOVA, specifically the **Kruskal-Wallis** test.

5 EXPERIMENT EXECUTION

An overview of the experiment execution can be seen in Fig. 2.

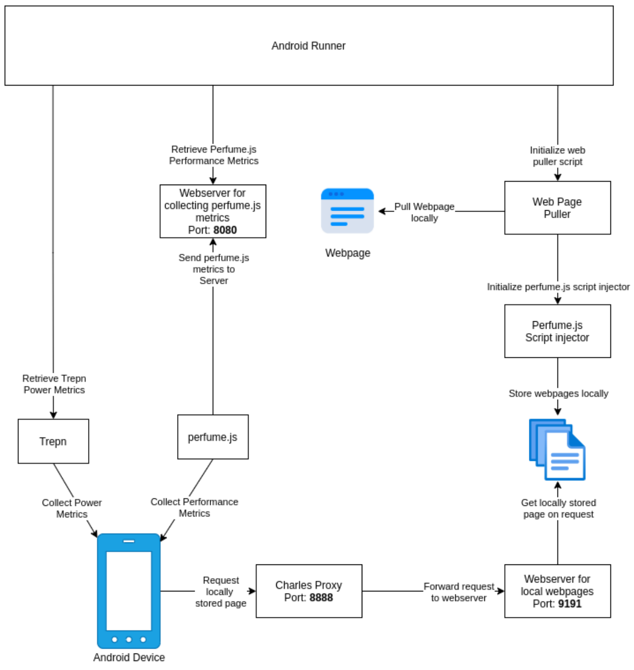

Android Runner
Retrieve Perfume.js Performance Metrics
Initialize web puller script
Webserver for collecting perfume.js metrics Port: 8080
Pull Webpage locally
Web Page Puller
Webpage
Send perfume.js metrics to Server
Initialize perfume.js script injector
Perfume.js Script injector
Retrieve Trepn Power Metrics
Store webpages locally
Trepn
perfume.js
Collect Power Metrics
Collect Performance Metrics
Get locally stored page on request
Request locally stored page
Charles Proxy Port: 8888
Forward request to webserver
Webserver for local webpages Port: 9191
Android Device

Figure 2: Component diagram of the experiment execution

The automation of our experiment is ensured by **Android Runner** [14]. This tool loads every website on Google Chrome on the Android Device by executing Android *adb* commands. Before the

[2]https://www.statista.com/statistics/268299/most-popular-internet-browsers/

load of a web app, the Android Runner tool starts a power profiler for monitoring the execution.

Our experiments are executed on a setup that consists of a laptop having the following specifications: Ubuntu 18.04.3 LTS OS, Intel I7-4710HQ with 16GB RAM. We run the Android Runner framework together with additional web servers needed for the experiments on this setup. The Android device that we will run our experiment on, will be connected via USB to the laptop.

To analyse the performance metrics of various websites we use **Perfume.js** [15]. This is a JavaScript library that allows us to monitor metrics such as Navigation Timing, Load Speed or Load Responsiveness. We will use this library by adding it to every web-page that we will base our experiments on.

We randomly sample 30 websites from the Tranco List using a custom developed script that also checks that a website is valid and reachable and then we download all 30 websites locally before the start of the experiment. This is needed so that we can add the *Perfume.js* library to every web-page that is tested. These locally downloaded webpages will be made available using a local web server running on port **9191**.

To download the web-pages locally we use the *wget* command as follows:

```
$ wget --page-requisites --convert-links
-P DIRECTORY HOSTNAME
```

This command will locally download the HOSTNAME website in the DIRECTORY folder location while also downloading additional items referenced in the HTML page and also converting the references to be accessible from our local web server. We chose this option in order to avoid Cross Origin Policy issues.

Once downloaded locally, using a Python script, we inject the Perfume.js library as a script in the head section of each *index.html* locally downloaded web-page. We also insert a script that gathers the perfume.js metrics and sends them to another webserver that collects the perfume.js metrics.

To serve the locally downloaded web-pages, we use a Python server that exposes a single endpoint which returns a local version of the web-page based on the specified path. Example of a curl request to retrieve the local version of twitter.com:

```
$  curl -X GET localhost:9191/
local_websites/www.twitter.com/index.html
```

In order to collect the Perfume.js metrics that are generated on the mobile device, Android Runner provides a web-server that listens on port **8080** for incoming metrics. These metrics are then stored in a csv file which can be accessed together with the other profiler results.

To simulate the conditions for 3G, we use the **Charles Web Debug Proxy** [16]. We run the Proxy Application on the same machine as the Android Runner Framework and then before the experiment, we configure our Android device to proxy the internet traffic trough the Charles proxy. We achieve this by configuring the WiFi settings of the Android device using the IP address of the machine, Charles is running on and the port **8888**.

To measure the Energy Efficiency we will use the batterystats profiler which is part of the android-runner framework and which uses the Android systrace utility tool. The batterystats profiler calculates the total energy consumed during a time period by listening to android systrace events such as *cpu idle* or *cpu frequency* and then making use of the **power profile xml file** which is found on every Android device, to calculate the energy consumed in **Joules**.

To obtain a dataset containing information about each of the runs, we use our own aggregation logic. To this end, we call a *script* on the various intermediate files generated by Android Runner. This will generate a comma-separated values (CSV) file, where each line corresponds to a single **run**, the columns represent the **factors** (i.e., performance metrics related to navigation timing and load speed) and the dependent variables (i.e., energy efficiency) and the values represent the applied **treatments**.

6 RESULTS

We address the research questions in three phases: data transformation, data exploration, followed by hypothesis testing.

6.1 Data Transformation

After the execution of the experiment, once all data has been collected, we proceed with the analysis. We first apply transformations on our dataset and variables, to prepare it for exploration and for applying the selected statistical tests. The transformation process is represented in Figure 3.

First, we perform an aggregated analysis, by combining the measurements related to single runs for every web application subject (step 1). We therefore consider the mean of the measurements obtained during 11 repetitions for each of our 30 subjects. After this step, the dataset was reduced from 330 lines to 30 lines (one for every web application). Next, we apply **min-max normalization** on the Perjume.js features, by scaling the performance scores to range [0,1] (step 2).

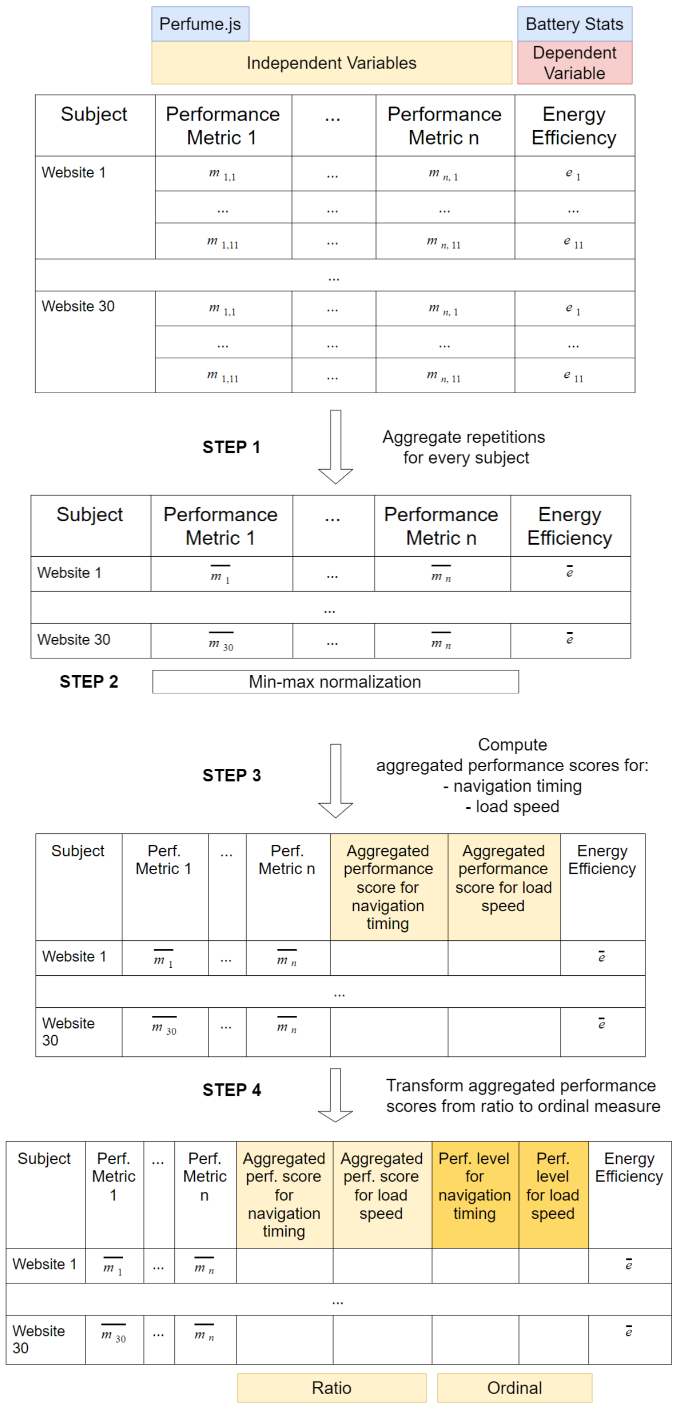

Perfume.js
Battery Stats
Independent Variables
Dependent Variable
Subject
Performance Metric 1
...
Performance Metric n
Energy Efficiency
Website 1
Website 30
STEP 1
Aggregate repetitions for every subject
STEP 2
Min-max normalization
STEP 3
Compute aggregated performance scores for:
- navigation timing
- load speed
Aggregated performance score for navigation timing
Aggregated performance score for load speed
STEP 4
Transform aggregated performance scores from ratio to ordinal measure
Perf. level for navigation timing
Perf. level for load speed
Ratio
Ordinal

Figure 3: Data transformation

We use the normalized features to obtain an aggregated performance score for each category of performance metrics, one for navigation timing and one for load speed (step 3). These scores will be analysed separately in the next phase. Specifically, the **navigation timing performance score** is obtained as the mean of the following normalized metrics: header size, fetch time, total time, download time and time to first byte, while the **load speed performance score** is the mean of the normalized first paint (FP), first contentful paint (FCP) and the total load time.

In terms of the dependent variable - energy efficiency, we transform the ratio measures into ordinal estimations to aid the statistical analysis (step 4). Finally, we obtain a new ordinal variable with three levels of performance as its values (low, average, high). To keep our experiment balanced, we associate each performance level to an equal number of subjects, based on the aggregated performance scores (see subsection 4.4).

The corresponding R script can be found in the *experiment replication package*.

6.2 Data Exploration

In this subsection we present descriptive statistics, along with visual representations (box plots, histograms, Q-Q plots) for a better understanding on the obtained data. In this process we explore each independent block, corresponding to different network conditions, separately.

Before starting the analysis, we set the significance level at 5%.

6.3 Dependent Variable Insights

6.3.1 *Wifi Data Exploration.*

- **Data Insights**
  The following plots summarize the dependent variable, energy efficiency, in relation to navigation timing performance and load speed performance, quantified using ordinal measures. Figure 4 shows that navigation timing performance might have an effect on energy efficiency. It can be observed that when navigation timing performance increases, energy efficiency also does. This does not happen in case of load speed performance, represented in figure ??, where high performance has the lowest mean. Both figures show data is right-skewed for high performance. However, in case of average and low load speed performance it is left-skewed. Also, variation is similar among load speed performance levels, but in case of navigation timing, it decreases for higher levels of performance.

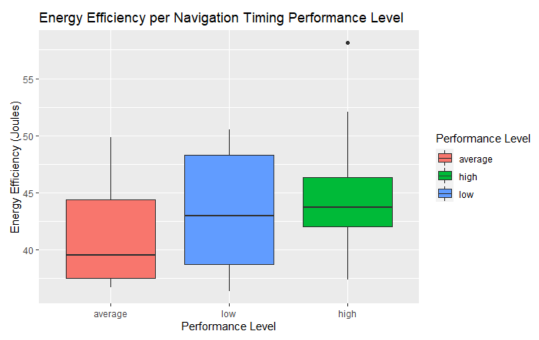


Figure 4: Boxplot Navigation Timing Energy Efficiency per Performance Levels

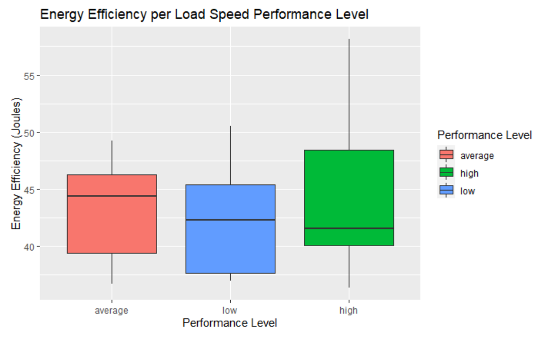


Figure 5: Boxplot Load Speed Energy Efficiency per Performance Levels

Looking at the scatterplots in Figures 6 and 7, we notice the same trend of higher energy efficiency for high navigation timing performance scores. It remains unclear in case of load speed performance though, where energy efficiency does not seem to depend on the performance score.

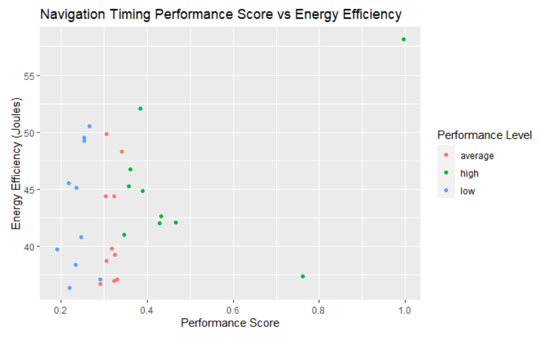


Figure 6: Navigation Timing Performance Score vs Energy Efficiency

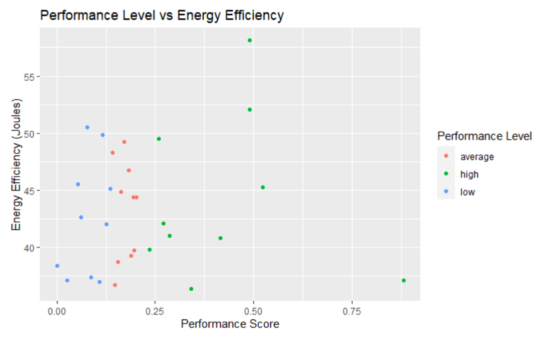
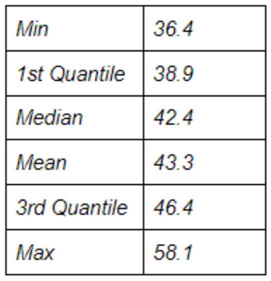


Figure 7: Load Speed Performance Score vs Energy Efficiency

- **Investigating the normality of the dependent variable** Below we present some insights on the dependent variable - i.e., energy efficiency, summarized in Figure 8. The median and the mean are very close, and values lie between 36.4 and 58.1, with a central tendency towards 42.4. We first consider the raw measurements and analyse the fit of the sample energy efficiency to the normal distribution, by looking at the histogram and Q-Q plot in Figure 9.



Min	*36.4*
1st Quantile	*38.9*
Median	*42.4*
Mean	*43.3*
3rd Quantile	*46.4*
Max	*58.1*

Figure 8: Raw Energy Efficiency Summary

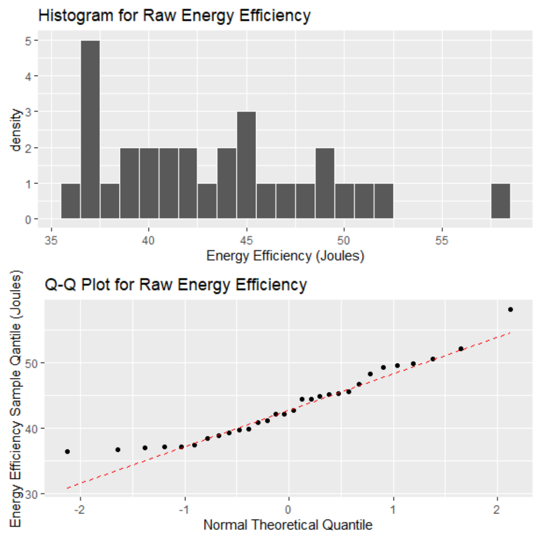


Figure 9: Raw Energy Efficiency (Wifi)

The histogram clearly shows the shape and the spread of the distributions of the data. However, it is more difficult to assess normality for smaller sample sizes, such as the one obtained from the experiment (with 30 observations). This happens because the appearance of the histogram relies on the number of data points and the number of bars in the plot. As a result, other assessment methods, such as QQ-plots can offer a better insight regarding the normality of the data. By looking at the histogram, energy efficiency does not seem to fit the normal distribution.
The QQ-plot shows all of the observations against a standard normal distribution, in other words, the actual values of X against the theoretical values of X under the normal distribution. If the points follow the line, then normality has been met. The QQ-plot shows some fluctuations of the data

at the extreme ends. However, we can say that the data is not far from normality, but given the small sample size, it is difficult to draw conclusions.
We also perform the Shapiro-Wilk normality test, suitable for small sample sizes. We obtain a p-value of 0.07, hence we cannot reject the null hypothesis that energy efficiency data is normally distributed.
We also check the symmetry of the data using box plots. The top left box plot in Figure 10 shows fairly symmetric data. However, the coefficient of skewness [17] returns a positive value of 0.65, meaning that data is slightly positively skewed, with the majority of data values less than the mean 43.3.

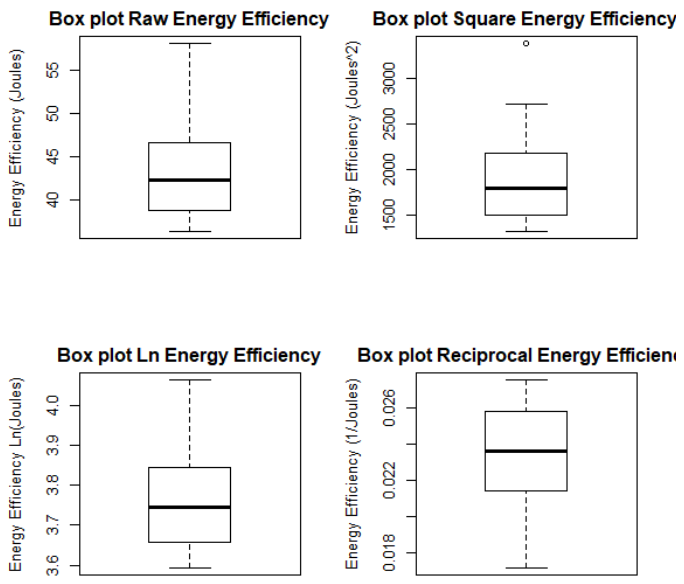


Figure 10: Box Plots Energy Efficiency Raw vs Transformations

We try to reduce skewness by applying some transformations on the energy efficiency variable. In this sense, we analyze the square (Figure 11), the natural logarithm (Figure 12) and the reciprocal (Figure 13) of energy efficiency.
Data visualisations look promising for the natural logarithm transformation and the reciprocal. There is a reduction in skewness from 0.65 to -0.22 for the natural logarithm and 0.42 for the reciprocal. We consider both values acceptable for the data being normally distributed. Also, the Shapiro-Wilk normality test does not provide any evidence that the data is not normally distributed.
In terms of the square of energy efficiency, we abandon this transformation, as the Shapiro-Wilk test provides evidence that data is not normally distributed.

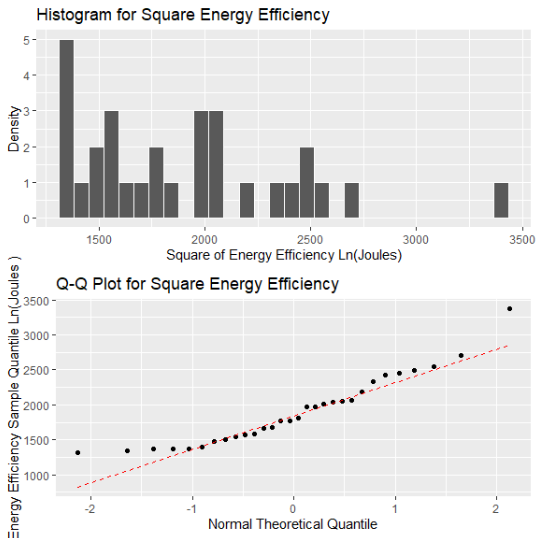


Figure 11: Square of Energy Efficiency (Wifi)

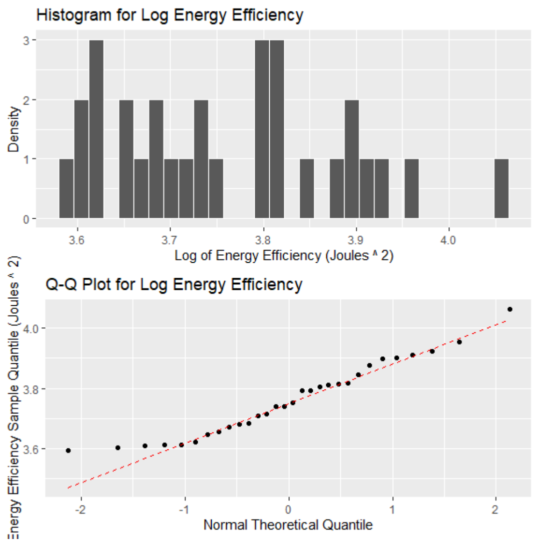


Figure 12: Natural Logarithm of Energy Efficiency (Wifi)

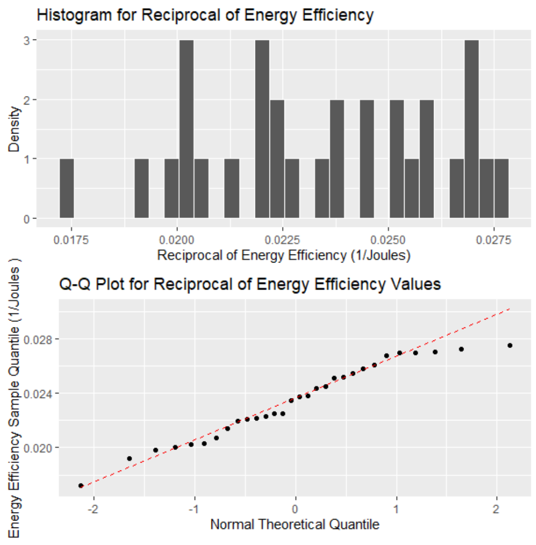


Figure 13: Reciprocal of Energy Efficiency (Wifi)

As a result, we will consider the reciprocal of energy efficiency when performing statistical tests to test the hypotheses, as this transformation proved to have the lowest skewness. Its summary is presented in Figure 14. The mean and the median are equal (both 0.23), signaling the fit to the normal distribution. We can thus use parametric tests, taking into account the observations and the results of the Shapiro-Wilk test with regards to normality.

Min	*0.017*
1st Quantile	*0.021*
Median	*0.023*
Mean	*0.023*
3rd Quantile	*0.025*
Max	*0.027*

Figure 14: Summary Reciprocal of Energy Efficiency

6.3.2 3G Data Exploration.

- **Data Insights** The figure 15 shows the distributions of three navigation timing performance levels of the energy efficiency. From the box-plot, we can see that the data of *low* and *average* levels are positively skewed (skewed right). However, the distribution of *high* level data is negatively skewed (skewed left). The performance level scatter plot also shows the same result.
  Meanwhile, the figure 16 shows different distributions of load speed performance levels. The data of three levels are all positively skewed. What's more, the data distributions in the two performance score diagrams are opposite.

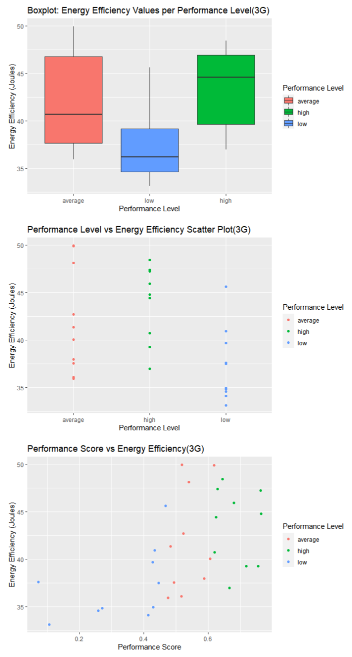


Figure 15: Navigation Timing Data Exploration

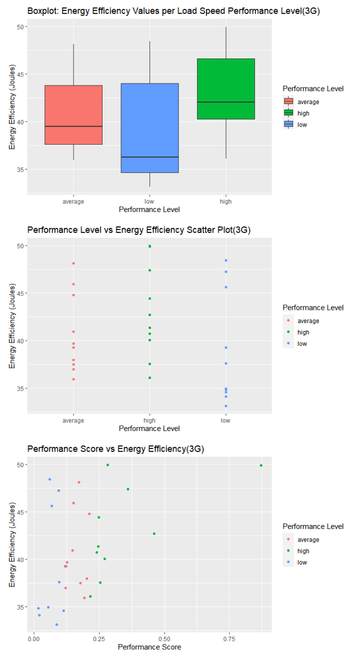


Figure 16: Load Speed Data Exploration

- **Investigating the normality of the dependent variable** Below we present some insights on the energy efficiency. Same as the wifi data, we first consider the raw measurements and analyse the fit of the sample energy efficiency to the normal distribution, by looking at the histogram and Q-Q plot in Figure 17.

The histogram clearly shows the shape and the spread of the distributions of the data. However, by given small size of data, the histogram is not that complete and some columns don't have any data. And it will become more difficult to assess normality for our experiment. From the histogram of the raw measurements, energy efficiency does not seem to fit the normal distribution. Considering this situation, the QQ-plot is a better method to provide insights regarding the normality of the data. The QQ-plot shows the whole data is fluctuate that the tail part is above the standard normal distribution, while the middle part is lower than the normal distribution. At last, we can say that the data is far from the normality. But by given the small sample size (only 30 observations), it is difficult to draw conclusions about it.

We also perform the Shapiro-Wilk normality test, suitable for small sample sizes. We obtain a p-value of 0.08, hence we cannot reject the null hypothesis that energy efficiency data is normally distributed.

We also check the symmetry of the data using box plots (Figure 18, top left) and by computing the coefficient of skewness, of 0.30. Hence, data is positively skewed, with the majority of data values less than the mean 40.9.

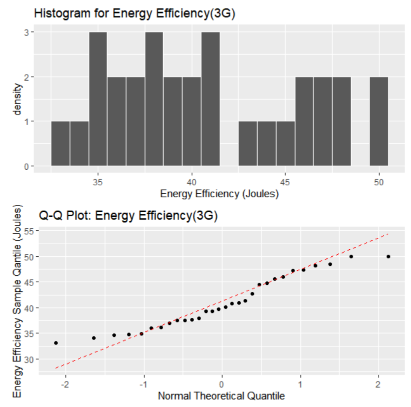


Figure 17: Raw Energy Efficiency (3G)

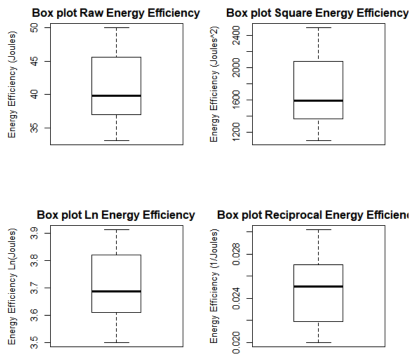


Figure 18: Box Plots Energy Efficiency Raw as Transformations

We try to reduce skewness by applying some transformations on the energy efficiency variable. In this sense, we analyze the reciprocal (Figure 19), the square (Figure 20) and the natural logarithm (Figure 21) of energy efficiency.

We get the value of skewness of 0.43, -0.03 and 0.17 in the square, the natural logarithm and the reciprocal of energy efficiency respectively. And we take the natural logarithm and reciprocal values into consideration for the data being normally distributed. Meanwhile, we get the small p-values (lower than 0.05) in the Shapiro-wilk normality tests for both transformations. So we can not reject the null hypothesis for the experiment that the data is normally distributed. In terms of the square of energy efficiency, we abandon this transformation, as the Shapiro-Wilk test provides evidence that the data is not normally distributed.

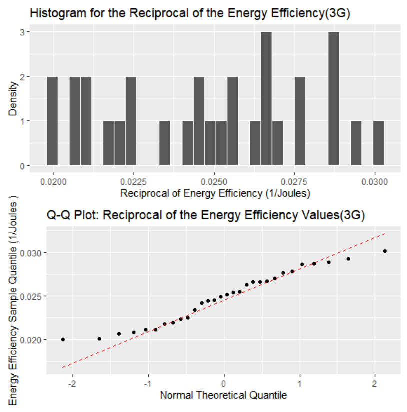


Figure 19: Reciprocal of Energy Efficiency(3G)

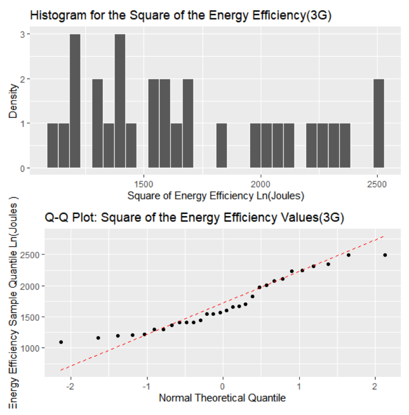


Figure 20: Square of Energy Efficiency(3G)

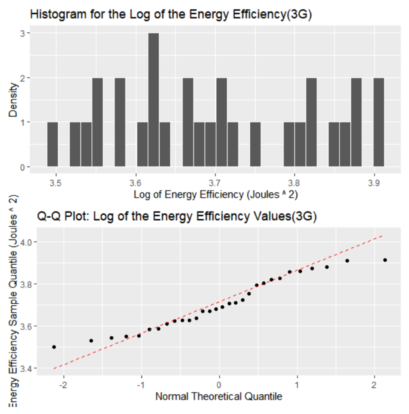


Figure 21: Log of Energy Efficiency(3G)

As a result, we will consider the reciprocal of energy efficiency when performing statistical tests to test the hypotheses, as this transformation proved to have the lowest skewness. We can thus use parametric tests, taking into account the observations and the results of the Shapiro-Wilk test with regards to normality.

6.4 Hypothesis Testing

6.4.1 Hypothesis Testing for Wifi Measurements. There are two hypotheses tested in this section, one for each performance category. In case of the data gathered over Wifi network connection, we use the reciprocal of energy efficiency to test the hypotheses stated in subsection 4.1.

Since the experiment is based on a 1 factor, > 2 treatments design, we intend to apply parametric tests for increased statistical power. To this end, we consider ANOVA to test the effect of each performance category (i.e., navigation timing and load speed) on energy efficiency.

The following assumptions must be met for the ANOVA test to be valid:

- The dependent variable (energy efficiency) is continuous
- The samples are independent
  This assumption is guaranteed by the experimental design.
- Normal distribution of the dependent variable between the independent groups
  Normality has been proven in the previous subsection.
- Normality of residuals and homoscedasticity will be tested after fitting the model

- Navigation timing
  After applying ANOVA to energy efficiency and navigation timing performance, the case factor is giving a p-value higher than the significance level. This means we cannot reject the null hypothesis, stating that the mean energy efficiency does not significantly differ among web applications having different navigation timing levels. Hence, **there is no statistical significance between energy efficiency and navigation timing over Wifi.**
  We still need to prove the following assumptions for the ANOVA test to be valid:
  - Residuals should be normally distributed
    It is dificult to draw conclusions by looking at the Q-Q plot, since the sample is fairly small. However, according to the Shapiro-Wilk test, we cannot reject the null hypothesis that energy efficiency data is normally distributed.

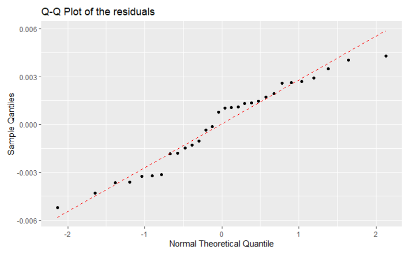


Figure 22: Q-Q Plot of the residuals

  - Homoscedasticity
    We use Levene's test with one independent variable to test the null hypothesis that the population variances are equal. The p-value is higher than the significance level, hence the differences in sample variances are likely to have occurred based on random sampling from a population with equal variances.
- Load speed
  After applying ANOVA to energy efficiency and load speed performance, the case factor is giving a p-value higher than the significance level. This means we cannot reject the null hypothesis, stating that the mean energy efficiency does not significantly differ among web applications having different load speed levels. Hence, **there is no statistical significance between energy efficiency and load speed performance over Wifi**.
  We still need to prove the following assumptions for the ANOVA test to be valid. We use the Shapiro-Wilk test for normality of residuals and Levene's test for homoscedasticity. The assumptions of ANOVA are again met.

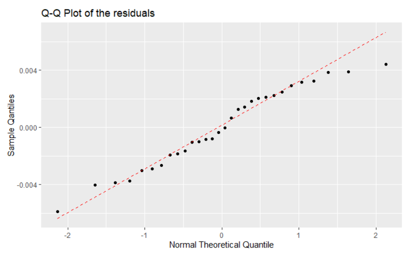


Figure 23: Q-Q Plot of the residuals

6.4.2 Hypothesis Testing for 3G Measurements. The hypothesis tests of 3G measurement data are same as the ones of wifi measurement data. There are two hypotheses tested in this section and one for each performance category (navigation timing and load speed). We also use the reciprocal of energy efficiency to test the hypotheses stated in subsection 4.1.

In this experiment, there is one factor, >2 treatments, thus we use ANOVA to test for the effect of each performance category on the mean energy efficiency. The following assumptions must be met for the ANOVA test to be valid:

- The dependent variable (energy efficiency) is continuous
- The samples are independent. - True (This assumption is guaranteed by the experimental design.)
- Normal distribution of the dependent variable between the independent groups. - True (Normality has been proven in the previous subsection.)
- Normality of residuals and homoscedasticity will be tested after fitting the model

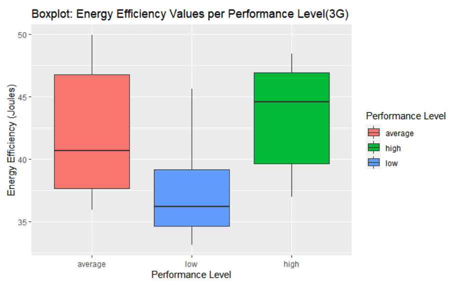


Figure 24: Energy Efficiency Values per Navigation Timing Level(3G)

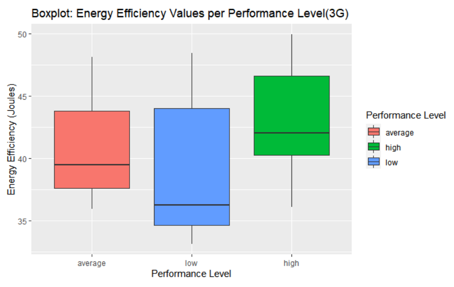


Figure 25: Energy Efficiency Values per Load Speed Level(3G)

Before we perform the ANOVA test on both performance, we also explore the data of energy efficiency to navigation timing and load speed. Figure 24 shows the distribution of three performance levels to energy efficiency. The *low* and *average* levels are positively skewed, but the *high* level is negatively skewed. Oppositely, three load speed levels are all positively skewed in figure 25.

- Navigation timing

After applying ANOVA test to the energy efficiency and navigation timing performance, we have got a small p-value that is lower than 0.05. This means we can reject the null hypothesis that the mean energy efficiency does not significantly differ among web applications having different navigation timing levels. Hence, **there is some statistical significance between energy efficiency and navigation timing over 3G**. This also means there is some difference between groups but we don't know which group is higher or lower than other groups. To find out which group is differ than others, we perform the multiple comparison by using the Tukey's test in R.

Firstly, we check whether the data variances between these groups are homogeneous. We use the function *bartlett.test()* and the p-value is 0.4 which is higher than 0.05. It indicates that there is no significant difference in the variance of these groups, that is, there is no outlier. Then we use the function *TukeyHSD()* to do the comparison between groups. In the figure 28, we can see the difference between *low* and *average* and the one between *low* and *high* are statistically significant.

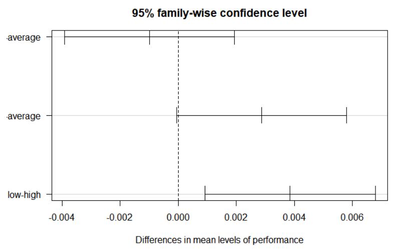


Figure 26: Difference in mean levels of performance

We also need to prove the assumption that residuals should be normally distributed for the ANOVA test to be valid. Although from the Q-Q plot, it is difficult to say whether the residuals is normally distributed or not. But through the Shapiro-wilk test, we get 0.2 of p-value that we can not reject the null hypothesis that energy efficiency data is normally distributed.

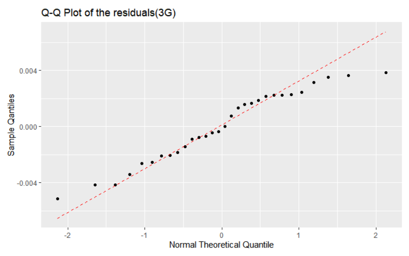


Figure 27: Difference in mean levels of performance

- Load speed

After we apply the ANOVA test to energy efficiency and load speed, we get 0.15 of p-value, meaning that we can not reject the null hypothesis that the mean energy efficiency does not significantly differ among web apps having different load speed levels. Hence, **there is no statistical significance between the variables over 3G**.

Same as the navigation timing, we also need to prove two assumptions for the ANOVA test to be valid. We use the Shapiro-Wilk test for normality of residuals and Levene's test for homoscedasticity. The p-values from both tests are higher than 0.05. This means both assumptions are valid.

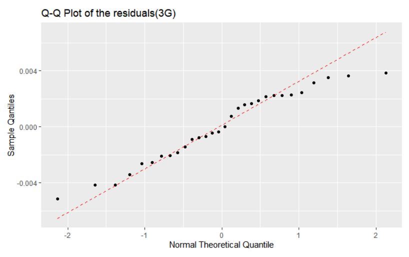


Figure 28: Difference in mean levels of performance

7 DISCUSSION

In this section, we discuss the results obtained after running the experiments and we elaborate on the implications of these results.

We measured the energy efficiency of 30 randomly sampled websites and repeated the experiment for 11 times under 2 scenarios: for WiFi Data connection and 3G Data connection.

The main research question of this research is *"To what extent does performance correlate with energy efficiency of web applications at first load?"* The null hypothesis states that the mean energy efficiency does not significantly differ among web applications having different performance metrics. More specifically, we aim to identify if the performance metrics significantly impact the energy consumption of an Android Device. Using the results obtained after analysing the WiFi and 3G data, we can partially reject the null hypothesis as we have only identified a correlation for the 3G data. The research question is thus answered by claiming that only under 3G network conditions bad performance metrics, specifically those quantifying navigation timing, can translate to an increase in energy consumption.

The different result between the network conditions (Wifi and 3G) does not necessarily imply that correlation does not exist over Wifi. It could just be that because of the fast data transfer rate, the influence of web application performance becomes significantly lower and undetectable in this experiment.

Our results cannot fully be used as an indication that slower network conditions result in bad energy performance. For this to be a true statement, further experiments using various other network conditions such as 2G, 4G or 5G and using different device types such as high end ones or low end ones should be carried out. In addition, a larger sample size should be considered with websites that are hosted in the same regions.

8 THREATS TO VALIDITY

In this section of the report we discuss the possible threats to the validity of our experiment, whether these threats affected the experiment and the mitigations we put in place to counter these effects.

8.1 Internal Validity

When running the experiments, we have noticed some issues pertaining to the testing framework that could have had an impact on the validity of our experiment data. More specifically we have identified a limitation of the Android Runner framework that would not allow us to choose a variable profiling time for the batterystats profiler. We initially set out to stop the profiling dynamically instead of choosing a fixed amount of time per website. This would have allowed us to precisely measure the energy consumed by a website until the website is completely loaded. The switch to dynamic profiling time was possible for the Perfume.js profiler but it was not possible for the batterystats profiler as it was relying on Android system calls for measuring consumed energy and these calls required a fixed amount of time to run.

For one website that was profiled during the 3G scenario, we had to increase the profiling time by 5 seconds in order to capture the Perfume.js metrics. This represents 11 runs out of 660 total runs. In order to have had a perfectly valid experiment result, we would have had to rerun the entire experiment execution for the 25 hours of data that we already generated.

8.2 External Validity

In order to examine the threats to external validity, it is first important to consider whether the sample of websites we selected to perform our analysis on was representative. We analysed a sample of 30 observations, which might not be representative enough for the entire population. It is advisable to increase the size of the samples in future experiments. The sample was selected randomly ensuring there would be no prior bias, however this ultimately resulted in the majority of websites originating from or having servers in Western Europe. This could have had some influence on the load times, since load times for these websites would inevitably be faster than those based in Russia for example. Therefore to mitigate this in the future, it would be good to include location of servers as a factor in the selection process of the candidate websites in order to make sure that this is not an external factor that would affect our results.

Additionally, in our experiment we used a mobile device that would be classified as 'low end' by modern day standards. This could be a potential threat to the validity of our experiment in terms of making the experimental environment as realistic as possible since the majority of mobile phone users would own a 'high end' device, and it is likely that programmers would be coding websites with newer technologies in mind. As a mitigation it would be good to conduct the same experiment on a high end device as well to see whether this does influence the results in any way.

To mitigate any other threats to external validity, we made the experimental environment as realistic as possible by using a real Android device instead of a simulator and by polling real websites. Furthermore the nature of our experiment assumes that it is not affected by other external factors, such as the time of day, or the day in the year on which we conduct the experiment. However, this should be taken into consideration in a future experiment, as many commercial websites scale down their servers during off-peak hours to optimize costs.

8.3 Construct Validity

To mitigate threats to the validity of our constructs we ensured that we defined these well and early on as can be seen by our GQM Tree in Figure 1. Furthermore we made sure we are analysing the appropriate variables that would allow us to make the relevant conclusions to our research question and have justified the choices for these variables throughout this report.

8.4 Conclusion Validity

To mitigate any threats to validity we made sure to choose the appropriate statistical tests to analyse the results of our experiment and come to the appropriate conclusions. Additionally, we took care not to violate any assumptions that these tests made. For example, since the tests required normalized data for correct results, we made sure to normalize the data before applying the relevant test.

9 CONCLUSIONS

The aim of this research was to ascertain whether there was any correlation between the performance of a web application and the energy efficiency on mobile devices. This experiment should raise awareness to software developers on performance, as the results show there is correlation with energy efficiency under 3G connectivity. The conclusion should not be limited to only this type of network conditions, as download speed might be an essential factor in the relevance performance has on energy efficiency. Although the network used by the device is outside of the programmer's control, this discrepancy indicates that it is advisable for the programmer to always consider optimizing for better performance in the cases where the web application is not accessed over a fast network.

There are a number of possible extensions to this experiment. Firstly, we would repeat this experiment with a number of different devices, both low-end and high-end, to decide how much of an influence the modernity of the device and the hardware has on energy efficiency as a pose to the performance of the web application. Secondly we could analyse the difference in performance over more networks such as 4G instead of only Wifi and 3G. Finally, we could improve the subject selection of the web applications. For instance, we could test the effect on energy efficiency grouped by web applications written in different programming languages, or with servers located in various distances physically from the testing location.

REFERENCES

[1] K. B. Baltzis, *Key Issues and Research Directions in Green Wireless Networking.* IGI Global, 2015.

[2] "Energy efficiency and the user experience." [Online]. Available: https://developer.apple.com/library/archive/documentation/Performance/Conceptual/power_efficiency_guidelines_osx/

[3] M. A. McNeil, S. de la Rue du Can, D. Hamza-Goodacre, and P. Roy, "Energy efficiency indicators and impact metrics," Tech. Rep., 02/2016 2016.

[4] T. Johann, M. Dick, S. Naumann, and E. Kern, "How to measure energy-efficiency of software: Metrics and measurement results," *2012 1st International Workshop on Green and Sustainable Software, GREENS 2012 - Proceedings*, 06 2012.

[5] K. Chan-Jong-Chu, T. Islam, M. M. Exposito, S. Sheombar, C. Valladares, O. Philippot, E. M. Grua, and I. Malavolta, "Investigating the correlation between performance scores and energy consumption of mobile web apps," in *Proceedings of the Evaluation and Assessment in Software Engineering*, 2020, pp. 190–199.

[6] "Lighthouse | tools for web developers | google developers." [Online]. Available: https://developers.google.com/web/tools/lighthouse

[7] "Élevez vos applications mobiles au niveau des vos ambitions business." [Online]. Available: https://greenspector.com/en/home/

[8] X. Chen and Z. Zong, "Android app energy efficiency: The impact of language, runtime, compiler, and implementation," in *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom) (BDCloud-SocialCom-SustainCom)*, 2016, pp. 485–492.

[9] L. Ying-Dar, H. Cheng-Yuan, L. Yuan-Cheng, D. Tzu-Hsiung, and C. Shun-Lee, "Booting, browsing and streaming time profiling, and bottleneck analysis on android-based systems," *Journal of Network and Computer Applications*, 2013. [Online]. Available: https://doi.org/10.1016/j.jnca.2013.02.024

[10] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Profiledroid: Multi-layer profiling of android applications," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 137–148. [Online]. Available: https://doi.org/10.1145/2348543.2348563

[11] W. Oliveira, R. Oliveira, and F. Castor, "A study on the energy consumption of android app development approaches," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, 2017, pp. 42–52.

[12] V. R. Basili, *Software Modeling and Measurement: The Goal Question Metric Paradigm.* Computer Science Technical Report Series, CS-TR-2956 (UMIACS-TR-92-96), University of Maryland, College Park, MD, September 1992.

[13] V. Le Pochat, T. Van Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen, "Tranco: A research-oriented top sites ranking hardened against manipulation," *Proceedings 2019 Network and Distributed System Security Symposium*, 2019. [Online]. Available: http://dx.doi.org/10.14722/ndss.2019.23386

[14] M. Ivano, G. Eoin, L. Cheng-Yu, V. Randy de, T. Franky, Z. Eric, P. Michael, and K. Luuk, "A framework for the automatic execution of measurement-based experiments on android devices," 2020.

[15] L. Zizzamia, "Perfume.js." [Online]. Available: https://zizzamia.github.io/perfume/

[16] "Charles proxy." [Online]. Available: https://www.charlesproxy.com/

[17] D. N. Joanes and C. A. Gill, "Comparing measures of sample skewness and kurtosis," *Journal of the Royal Statistical Society. Series D (The Statistician)*, vol. 47, no. 1, pp. 183–189, 1998. [Online]. Available: http://www.jstor.org/stable/2988433