Visualizing User Hardware Data

Dylan Perrenoud | Joseph Cook

Abstract:

This project utilizes Python, HTML, Javascript and the data gathered by Steam in the Steam Hardware Survey to create visualizations of said data such as bar charts and line graphs to document the changes in PC hardware usage over the past 16 years. The graphs are displayed in an interactive webpage at this [Github.io Page](https://hoolahups.github.io/PyProject/) and the raw code and image files can be found at the github page of the same name (<https://github.com/Hoolahups/PyProject>). Through HTML and JavaScript scripts the page has been made interactive with selectors in order to streamline user experience and make the visualization as easy on the user as possible.

Introduction:

In a world where technology is always changing, the realm of PC hardware is no stranger to constant advancements. It can be very difficult to keep up with what technology is in most demand, especially for newer users. With this in mind, we created this project with the goal of visualizing this change over time, seeing what parts are most preferred, and also as a way to visualize past trends. We were able to obtain the Steam Hardware Survey for every month since November 2008. This survey is conducted automatically to users who have Steam installed on their computers. It gathers all of the hardware data from each computer such as the graphics card model, CPU speed, amount of RAM, and many other metrics. Using this massive dataset, we can apply some data cleaning and visualization methods to easily display how the usage of these parts has been changed over time.

Methods:

Our raw dataset obtained from the survey includes over 57,000 rows of data, many of which are unnecessary for the average user. This info includes every bit of hardware inside each user’s computer as well as driver and operating system information. The first thing we want to do is filter down our data into groups such as GPU models, CPU speed, and RAM amount.

A white background with colorful text

Description automatically generated

Some challenges with this data cleaning is that some observations are the same piece of hardware, but with different names. In the example above, we combined Intel CPU speeds with AMD CPU speeds, as these are both brands for CPUs and we are interested in the speed only. We had to do many of these cleaning scripts inside the “Data-Cleaing.py” script to gives us separate datasets for each hardware piece. The entire list of hardware we thought was fit to include was CPU’s, Display Resolution, GPUs, Hard Drives, Core Count, RAM, and VRAM. Each of these parts gives us the date it was surveyed, the category (ex: GPU), the name, the percentage of people who use it, and the change of percentage from the previous month.

Given this data, we thought it would be best to show the usage over time as well as the change in usage. A problem we ran into at first is the sheer number of visualizations we would need to show, as we have about 16 years of data for each hardware part. To solve this problem, we created a function called plot\_and\_save(), which iterates over each month and saves all the plots as PNG’s.

A screenshot of a computer code

Description automatically generated

Using this function, we can write for loops for the part we want and the directory we want to save it in, and it will create all of the graphs over time for us. Below is an example of doing this for the Graphic Cards:

A computer code with many colorful text

Description automatically generated with medium confidence

A green bar graph with numbers and text

Description automatically generated

Figure 1 Percentage Graph Example

While these functions are great for a one size fits all, sometimes we had some issues graphing. The first major issue was that there were months that had NA’s for some of the parts. This would cause some of the plots to be completely blank, which would completely ruin our analysis over time. Our solution to this is included in the function above, which was to filter out any months that had any more than 2 missing values. The second problem we ran into was having a bunch of models for a certain hardware part. For example, there are many GPU’s in use today and a graph with 20 or more bars would be nonsensical. Our solution for this is also in the function above and was to take only the top 10 highest GPU’s in use. Note that we only did this in certain parts, as 12 or 13 bars is okay, but more than 20 is not.

After plotting all of these months of dataset, we are left with hundreds of plots without an effective way to interpret them over time. Our solution to this was to create a GIF object from all of our PNG’s, which show the months going from 2008 to the present day. To do this, we read the documentation on the ‘Pillow” library in Python, which would give us the ability to render these GIF’s. (These can be viewed on the interactive github.io page, as PDFs don’t take kindly to gifs)

A computer code with many colorful text

Description automatically generated with medium confidence

The code above uses ‘Image’ from the ‘Pillow” library to create these GIF’s, which show the top parts over time based on their percentage of use. We did not do this for the percent change as this would not be a very interpretable animation. With all of our figures, GIF’s, and data, we needed a way for a user to navigate it. The way we decided to do this was through a github.io page and html, which can be read about [here](https://pages.github.com/). On this page we created a menu that allows users to pick either a GIF, Percentage, or Change. Percentage and change can both be specified by the part and month, which pulls up the resulting visualization. GIF will show the over time usage for whatever part that is chosen.

After creating this, we also decided to create a line graph that shows the percentage over time of a specific part. For example, it will show the usage of the NVIDIA GeForce 6600 from the years of 2008 to now by month. It does this by grouping each dataset by name and then creating a graph of percentage vs date for each component. This will be especially useful when looking for a part to purchase to view its popularity. The interactive webpage that allows you to display all of these figures is listed below:

[Github.io Page](https://hoolahups.github.io/PyProject/)

Results:

It’s difficult to really grasp the magnitude of the visualizations in this project. With a total file size of nearly 200 mb and over 1000 graphs, it is infeasible to actually analyze all of the data and so it seems more prudent to look for interesting trends in the data. According to Steam, “Around August 2017, we started seeing larger-than-usual movement in certain stats, notably an increase in Windows 7 usage, an increase in quad-core CPU usage, as well as changes in CPU and GPU market share. This period also saw a large increase in the use of Simplified Chinese. All of these coincided with an increase in Steam usage in cybercafés in Asia, whose customers were being over counted in the survey.” We can notice this spike very easily in the line graphs of 4 physical cores:

A graph on a black background

Description automatically generated

Figure 2 A Blip in 2018 Showing the Reporting Errors

Some other applications of this could be tracking the rise and fall of several different components in the same class using the gif visualizer, and then using the percentage visualizer to pick a specific year and month to take a closer look at. There are many other applications for this visualizer too, including using your own datasets with some modification to the code in order to display other data of a similar nature, like baby names over time or perhaps car part market shares.

Conclusion/Future Work:

This is a rather rudimentary but fun example of how python can be used to easily clean data and produce images in a way that can easily interface with HTML and JavaScript. Had we tried to do this by hand, creating proper naming schemes for files and organization would have been a nightmare. Additionally, due to the nature of Python being able to run entire scripts at once, it was harder to develop than a similar package in R, but much easier to execute. Instead of running several functions like you would in R with updated Steam Hardware Survey data when it releases, you simply open 4 files, run them, and then your entire dataset is updated and ready to be displayed on the interactive viewer again. While not groundbreaking for computer science as a whole, we can take this knowledge and use it to make professional looking and interactive displays for data in the future and impress future employers or contractors with clean and easy to access visualizations of all kinds of data.

References:

<https://pillow.readthedocs.io/en/stable/>

<https://github.com/jdegene/steamHWsurvey>

<https://pages.github.com/>

ChatGPT

Copilot