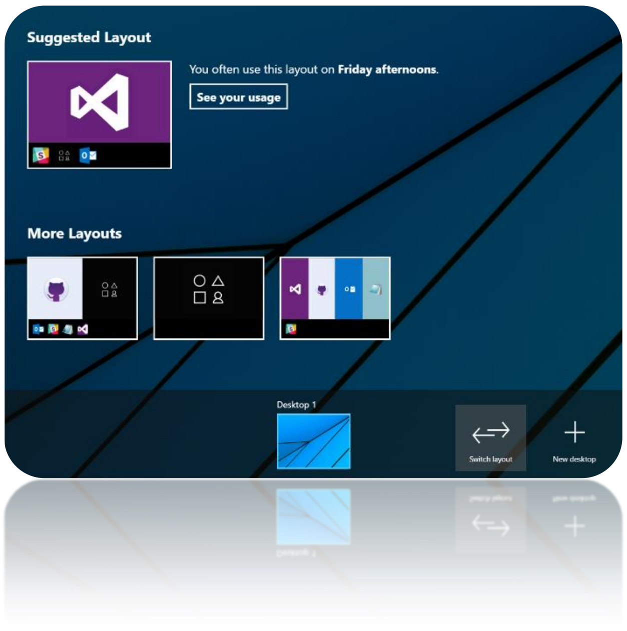
Rensselaer Polytechnic Institute

Desktop Layout Portfolio

Osvaldo Rosado & Adam Ryason



Interactive Visualization
Spring 2016

INTRODUCTION

Computers are generally competent at doing what we tell them to do, but they don't really have the intelligence to predict what we want them to do. This isn't for a lack of data collection usually. Through some advanced means of visualizing the telemetry, and preprocessing clustering techniques that we see in social media visualizations, the data our computers already collect could provide significantly more "intelligence".

Our central research question is: **Can visualizing common desktop layouts in a context-aware fashion improve computer usage?** We hypothesize that this is indeed the case when users are allowed to use these visualizations to actually change their current desktop layout, at least for the target audience of computer users who work on various "tasks" at roughly similar schedules, such as office workers and college students. The data that we would be visualizing would be collected on an individual user-specific basis as time-series data, hopefully allowing us to pick out trends and "clusters" of apps commonly used together. The goals for this project are to develop an application that will 1) Collect telemetry data based on the programs used on a computer and then 2) Create clusters of programs used together to provide layout suggestions.

Even since the early development of the GUI, it was seen that they were not very intuitive and seen as a necessary evil [Van Dam, 2000]. Van Dam talks about how interfaces should be created to match our human perceptual, cognitive, manipulation and social abilities. By creating a smart GUI, which would reduce the amount of interaction required by a user, we would be making the interaction with a computer closer to a human-human interaction. At the time of this paper, there were no public desktop telemetry software or prior work to reference to; however Microsoft has acknowledged that they are collecting this information from our desktops for their private use.

An early paper covered in class is also quite useful for guiding our project. In "Social Network Clustering and Visualization using Hierarchical Edge Bundles", Jia, Garland, and Hart speak about clustering and visualizing social networks [Jia, Garland, & Hart 2011]. We find this to be particularly useful for our use case as one can picture apps as friends in a social network, and apps used together as relationships, and duration used together as the strength of their relationship. Using the methods that they describe, we can bundle applications into groups commonly used for tasks and present these finding to the user in simple to understand ways.

We've also covered streamgraphs in class, which we've come to believe are particularly useful for depicting the kinds of time series data we'll be dealing with for this assignment. In "Stacked Graphs – Geometry & Aesthetics", Byron and Wattenberg cover how they created these types of stacked graphs and the relationship between aesthetics of data and legibility of data [Byron & Wattenberg 2008]. We think their paper can guide us to produce similar results to great effect for showing users the data our tool has collected on them, and how our tool comes to the conclusions that it does.

In "When One Isn't Enough: An Analysis of Virtual Desktop Usage Strategies and Their Implications for Design", Ringel discusses the implications of virtual desktops on desktop usage and the various organization strategies that arise from their use [Ringel 2003]. As our tool effectively allows users to pick from automatically generated sets of virtual desktops, the findings in this paper can help us determine layouts that will be more effective for users to complete tasks.

INITIAL VISUALIZATION IDEAS

Allen the Architect spends most of his day on his laptop. He relies on his laptop for work and he uses it at home to play video games and watch movies with his family. When he is at work he does not want to be distracted by his home life and vice versa. When he opens our desktop layout selector, his computer predicts that when he's on it in the morning, he is probably at work where he will be accessing his architecture software, Internet Explorer, Spotify and Photoshop. He is then presented with the layouts that he typically accesses. Once selecting a layout, it will open the applications within it. If Allen is plugged into an additional monitor, his computer will recognize this and offer to load his dual-monitor profile for his commonly used theme, where he maximizes his space for Revit and has the left monitor snapped with IE, Photoshop and Spotify.

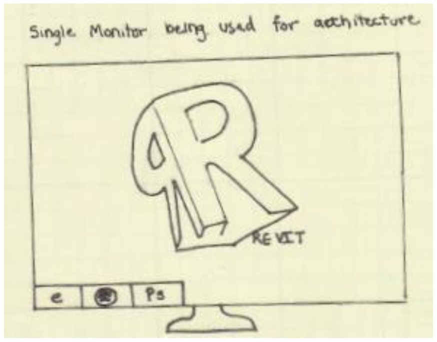


Figure 1: Initial sketch of visualization on single monitor

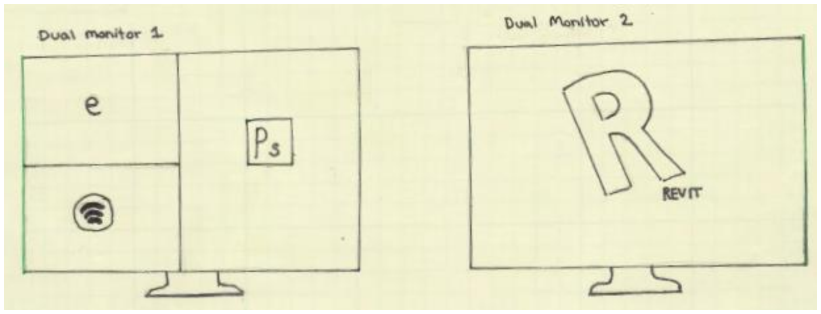


Figure 2: Initial sketch of visualization on dual-monitors

When Allen gets home and turns his computer back on, the computer predicts that he will most likely want to play video games. The layout that he is prompted with includes, StarCraft 2, Twitch.com, OBS and Spotify.

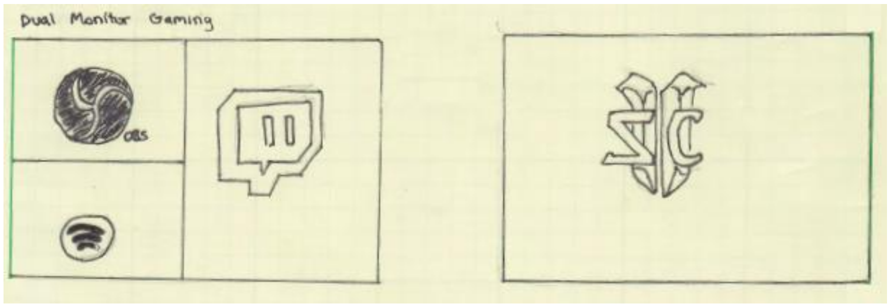


Figure 3: Initial sketch of visualization on dual monitors with a "Gaming" layout

Allen wants to know how his computer is predicting his trends, so he accesses the statistics GUI and is presented with a Streamgraph. One of the options to view is the number of days he is on a given application for different time intervals throughout the day. He could also change the view to the number of minutes he has the application active on his desktop for the time interval. We will be utilizing the algorithms discussed by Byron & Wattenberg.

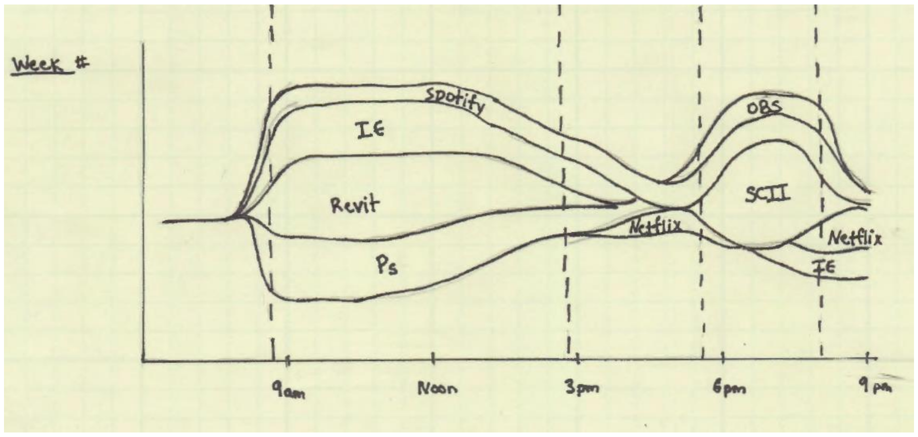


Figure 4: Initial sketch of the program usage in streamgraph

DATA COLLECTION & ANALYSIS

In order to create layouts based on the user, their application usage had to be collected. We developed a background application, *Desktop Telemetry (DT)*, which would collect data once every minute. The application collected all applications that the user had running on their computer as well as its location, dimensions, minimized/maximized state, file path, z-index, and window title. This data was then stored in an SQLite database that is local on the user's machine, however this could be stored remotely in the future. DT must be started each time the computer is turned off, but will pause when the computer is put to *Sleep* or *Hibernate*. Problems that we encountered were due to applications that were never visible but were considered running. These applications are different than background processes and we suspect that they are all "universal windows platform modern applications" (colloquially known as Metro apps) being run pre-emptively for performance reasons. This was only a problem on Windows 10.

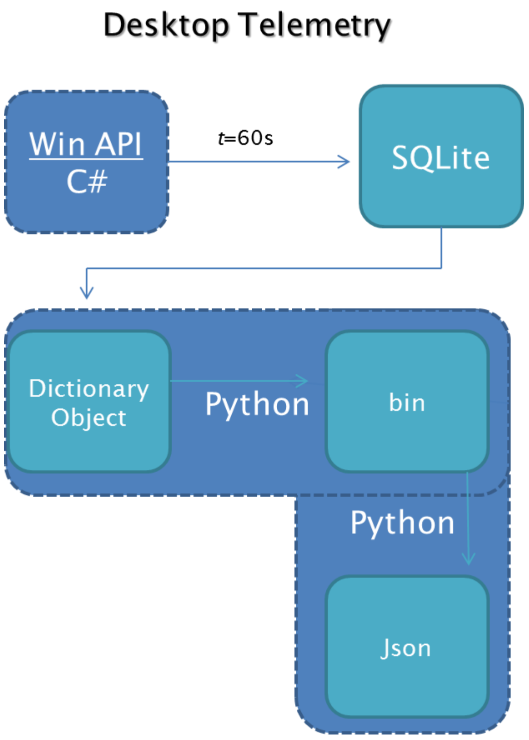


Figure 5 - Data Collection and Analysis Flow

Moving beyond mere data collection, we've implemented several Python scripts to perform analysis on the data. One primary script converts the SQLite database into a native Python dictionary, and "pickles" it for use by the other scripts. The other scripts process the data in two primary ways: to generate the clusters of programs and to generate the usage visualization. To generate the clusters of programs, all the times within a block of time were looped over to group the visible applications and the number of times these groupings appear within the time frame. These groupings are then ranked based on the number of times they appear in the time frame, with the grouping that appeared most as the most favorable group. To generate the usage visualization, the data set was divided into weeks and then subdivided into seven days and 24 hours. The usage and visible time of each program was then summed across all weeks to establish the number of seconds the app was opened during that hour across all the weeks.

In order for the algorithms to group the programs, it is suggested that DT has been running over the course of at least 7 days in order to make conclusions on the user's habits.

An interesting challenge in our data analysis was determining if programs were actually visible on screen. As we collected position, size, depth, and maximized/minimized information it was possible to calculate the visibility state accurately by recreating each scene through drawing to bitmap buffers. We implemented a hash function that would transform each application name into a specific color and drew each application to our bitmap in order of z-index. For performance reasons our bitmaps were severely resolution reduced to merely 25 pixels in width. These bitmaps were generated for each unique desktop state in our dataset, and then to determine app visibility we checked if any pixels were matching the hash for the application at hand. Although these bitmaps are not saved to disk in normal usage, we once exported them for debugging purposes (Figure 6).

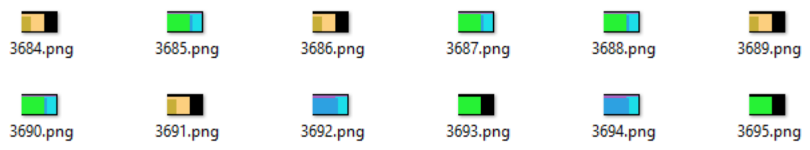


Figure 6 - Generated Bitmaps for Determining Application Visibility

For future work in data collection & analysis we would like to have all the data collection stored on a remote server where the analysis may take place. The collection of data would be anonymized so that no personal information or file information is taken from their computer, just their application usage. This would allow us to use more complex algorithms, such as machine learning, to make suggestions for the user. The next step would be to test K-means clustering on the collected data so that trends such as "What applications are started together?" or "What applications affect the trends of other applications?" can be detected.

VISUALIZATION GENERATION

There are two main parts to the visualization generation; they are the layout selection UI (Figure 7), and the usage analytics visualization (Figure 8).

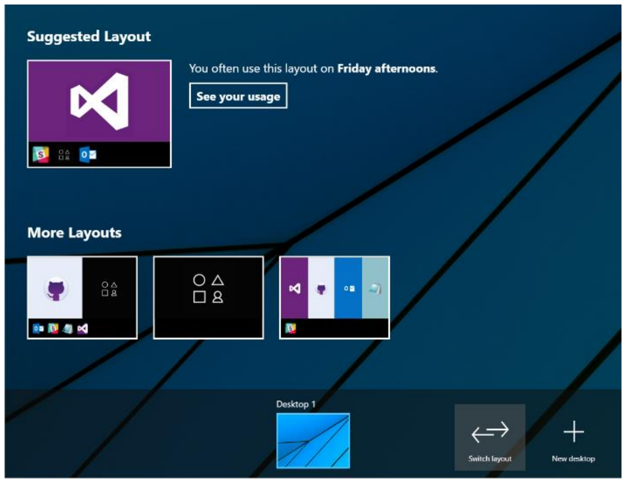


Figure 7 - The Layout Selection UI as seen when starting the application

The layout selection UI's primary purpose is to allow the user to quickly change their PC's current desktop layout to one suitable for the next task they intend to perform. It is a context-aware visualization that updates to showcase relevant suggestions to the current time. This was prototyped as a static visual in the design tool Figma, and then converted into a usable visualization in HTML5+JS+CSS. The primary invention in the layout selection UI is what we call "layout boxes". Layout boxes are the pictorial representations of proposed desktop layouts. They are graphically simple, with the upper primary section of the box representing applications visible on the computer screen and their positions on the screen, and the lower secondary section representing applications available on the taskbar but not actively visible. Our in-class "user study" provided validation that our layout boxes were easily understood visuals to represent what the program would do to the user's desktop if the layout were selected. To produce these boxes, we had to collect application icons from each app, and run script over the icons to determine the most dominant color in the image. For this color extraction task we used a library called ColorThief and for the associated icon extraction task we used the Python libraries ExtractIcon and Pillow. We believe that using dominant colors from application icons to shade in the visible application areas is an effective technique for this visualization.

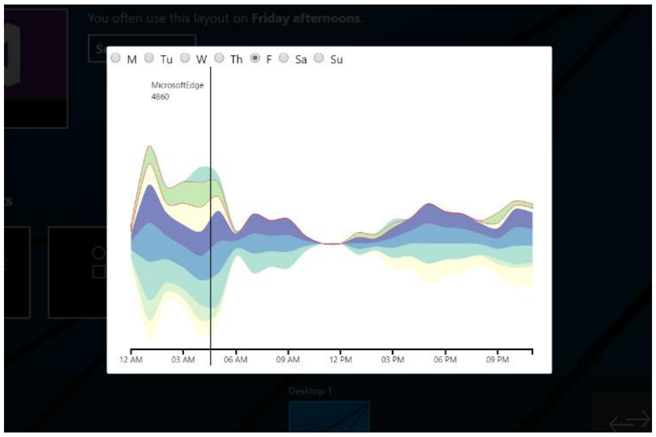


Figure 8 - The Usage Analytics Visualization seen in the application

The usage analytics visualization was prototyped using the Pandas and Pyplot libraries to generate the plots. We decided that we wanted a cleaner look for our final product that could be easily animated so we re-implemented them using D3. After testing various plot styles, such as bar charts, pie charts and time series plots, we noticed that the importance of these plots did not lie in the discrete values but in the overall trends. This led us to use a streamgraph to represent the program usage over time. Starting with the D3 code written by (Turman, 2015) we had to write our own parsing algorithms for our JSON format. We then expanded upon the streamgraph to take radio button inputs that would then dynamically change the dataset between different days of the week. For our streamgraph we chose ColorBrewer2 approved colors that were selected from the sequential dataset. We initially tested diverging and qualitative colors but felt that they contrasted too much within the application and looked unprofessional. We also updated the selection color to be red to stand out on the blue-green scheme.

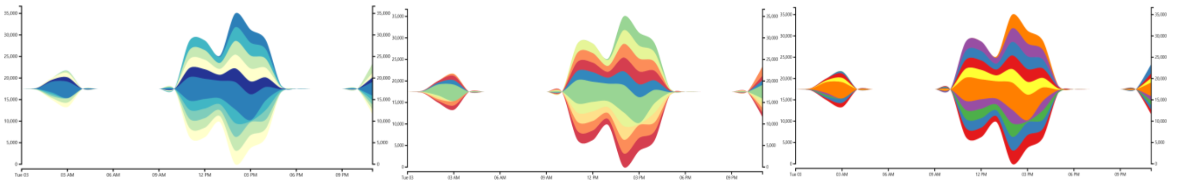


Figure 9: (left) the selected color scheme for our streamgraph. (Center) The diverging color scheme. (Right) The contrasting color scheme

When the user hovers over a layer in the streamgraph, it will highlight the selection in a red outline and fading the other layers. A tooltip will also appear that will tell them which program is selected and the total number of seconds spent with the program open for that hour across all weeks.

For future work in visualization generation, we would like to include a profile area where the user could give feedback to the application about their interests in specific programs, custom made layouts or automatic application startup. For the analytics visualization, we would like to include more interactions for the user in the graphs to help them understand their usage. Additional plot styles may also be helpful, where they would be able to visualize data for a specific program or grouping.

USER FEEDBACK

For our application demonstration, we showed the user interface for browsing and selecting program layouts as well as examples of the visualizations that users interact with to see how their usage is being tracked. We walked through the application with a group of students and then asked for comments on specific areas of the application. The areas we were looking for comment on were: if the layout boxes were effective visuals of what the application would do when the user clicked on them and if the timeline plot (see Figure 10) was effective at describing why the tool generated the suggestions it did.

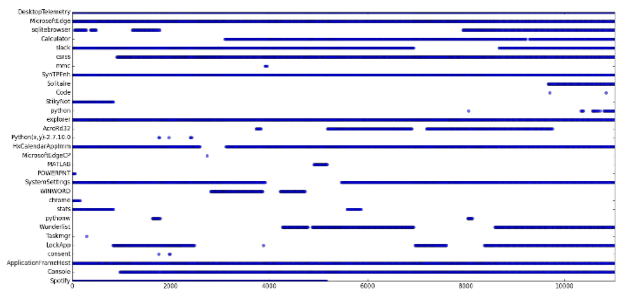
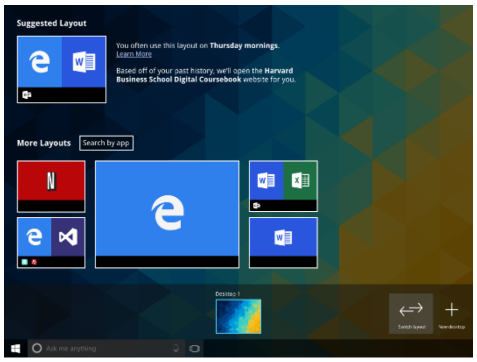


Figure 10: Timeline Plot Shown in Class for Feedback



You often use this layout on **Thursday mornings**.
Learn More

Based off of your past history, we'll open the **Harvard Business School Digital Coursebook** website for you.

Figure 11: Layout Selection Mock Shown in Class for Feedback

We received plentiful feedback on the layout selection UI. One piece of feedback was that the user interface needed to more explicitly tell (at least new) users what will happen when actions are performed. Another singular piece of feedback was that our data collection and processing technique would be very useful to generate results to appear in the "suggested app" area in the Windows 10 Start Menu. A regularly occurring feedback was that the layout boxes were intuitive but the taskbar & minimized apps were easily missed if the concept was not introduced in advance. There was also a comment that there should be some way to differentiate between multiple instances of the same application, for example if multiple Internet Explorer windows were opened.

When analyzing the proposed timeline plot many suggested for the rows to be sorted in order by most used, the range should be over a 24-hour period, and different colored bars for usage by day of week should be added. An additional request was a "Where Am I?" line on the timeline that would show users what applications they usually use at that day and time. The group did not like the initial format of times series data and recommended using a streamgraph, which we did create to replace the time series plot. It was also frequently recommended that the timeline should color code differently when apps are minimized or not visible. Some asked to be able to filter applications from timeline.

FORMAL USER STUDY

For testing our application, we want to test two aspects of it: the user-interface and the understanding of data visualization. Users will interact with our application by browsing different layouts of program combinations that they use together and selecting the one that is most suitable for what they want to accomplish. We are looking for subjects who interact with a computer very often throughout the day for work and personal reasons. To test this, the users will perform a series of tasks that will gauge their ability to identify and search the provided layouts.

The second part of the user testing would be the understanding of the data visualization. To test this, the users will perform a series of task that will gauge their ability to draw conclusions and search various plots of data. These data analysis tasks will be performed on time series plots, streamgraphs, and histogram plots.

Following the user study on the individual features, we would then run an alpha test, having a small group of people run the application on their machine for 6-weeks. The target audience would be architects, engineers, multimedia producers and computer programmers. The first week would involve the DT collecting data in order to make suggestions in the second week. We would then give the option to users for them to select a proposed layout but not require that they do so. Each week we would collect the previous week's data to see how the program is constructing groups.

CONCLUSION

The application was successful collecting the usage data on 4 machines (1 Windows 7, 1 Windows 8, and 2 Windows 10). It was then capable of grouping together applications that are frequently opened together and proposing suggested layouts in a clean and user-friendly fashion. We hope to continue developing the specific clustering algorithms as well as adding additional functionality to the application. We would like to properly assemble all the tools into a single program that can be installed on Windows computers.

We have a lot more data that we would like to take advantage of such as the location and size of the applications being opened. This would allow us to make multi-monitor desktops which we had initially wanted to implement but did not due to time constraints.

We ran into challenges primarily when implementing the streamgraph, once when we had to parse our JSON object to work with the D3 streamgraph and then again when we wanted the streamgraph to switch between days. Both the situations were unique in that no code was available that already accomplished this. Another problem that we had was time, as the application developed, we continued to get more ideas that would have been great to implement as they would have enriched the user-experience but we made sure to create a polished product before adding new features.

APPENDIX I: CONTRIBUTIONS

Adam drew storyboards for the desired layouts. He then wrote code for the initial PowerShell script to collect opened programs. Following that he worked on parsing and analyzing the SQLite data in Python using Pandas & PyPlot to debug the data collection. He then experimented with D3 to create the final stream graphs and add the additional interactions to the visualization.

Osvaldo expanded on the initial PowerShell data collection by creating a C# application that collected all necessary data in the background every minute and stored it into an SQLite database file. He also developed the mockups for the final user interface and implemented the primary UI. In addition to the primary UI, Osvaldo improved on the existing Python data collection to determine bundles of apps used together and rank them, as well as visibility testing and time series reporting, and exporting all of this processed data to JSON for the web interface.

BIBLIOGRAPHY

A. van Dam, "Beyond WIMP," *IEEE Computer Graphics and Applications*, vol. 20, issue 1, pp. 50-51, 2000.
L. Byron, M. Wattenberg, "Stacked Graphs – Geometry & Aesthetics", *IEEE TVCG*, 2008

Jia, Yuntao, Michael Garland, and John C. Hart. "Social network clustering and visualization using hierarchical edge bundles." *Computer Graphics Forum*. Vol. 30. No. 8. Blackwell Publishing Ltd, 2011.

Ringel, Meredith. "When one isn't enough: an analysis of virtual desktop usage strategies and their implications for design." *CHI'03 extended abstracts on Human factors in computing systems*. ACM, 2003.

"Privacy Statements for Windows 10 Technical Preview - Microsoft Windows." *Windows.microsoft.com*. N.p., Jan. 2015. Web. 06 May 2016.

Bostock, Mike. "D3.js - Data-Driven Documents." *D3.js - Data-Driven Documents*. N.p., n.d. Web. 07 May 2016.

Matplotlib Development Group. "Introduction." *Matplotlib: Python Plotting — Matplotlib 1.5.1 Documentation*. N.p., n.d. Web. 07 May 2016.

NUMFocus. "Python Data Analysis Library¶." *Python Data Analysis Library — Pandas: Python Data Analysis Library*. N.p., n.d. Web. 07 May 2016.

Dhakar, Lokesh. "Lokesh/color-thief." *GitHub*. N.p., n.d. Web. 07 May 2016.

Mandaga, Fadhil. "Firodj/extract-icon-py." *GitHub*. N.p., n.d. Web. 07 May 2016.

Pillow Contributors. "Pillow: The Friendly PIL Fork." *Pillow: The Friendly PIL Fork*. N.p., n.d. Web. 07 May 2016.