Sometime in June

Summary:

- Got HWB's script working on my computer
- Downloaded some data from HP-WREN website
- Created some kml files with the script
- Played around with google earth
- Edited hvb-atc.pl to change the color of the line extending to the ground to change based on altitude

Disclaimer: this was written a while after I actually did these things. My starting place was the script that HWB sent over. After downloading perl, I copied it from the email and made a new perl script. I did some googling to learn some perl basics, but my dad was a huge help in understanding the script. I needed to change it in order to make it work with my file structure, but (as far as I can remember,) that was all. For example, windows can't have colons in a file name, so I changed the script and the data I downloaded to use underscores instead. At this point I knew the important parts of the script and what I could change, mainly the end time variable. I didn't understand the structure of the KML file, which is why I started playing around with google earth.

First, I focused on the time slider. I played with the "window" and how big I could make it before my computer locked up. I noticed I could make the camera follow a path which I could use as a cinematic to capture certain points of interest. I looked at all the attributes of the data points which helped me understand how the program read attributes such as altitude from the raw data and wrote them into the kml format. I changed the color and icons in google earth, but not in the script. Finally, I just looked at the planes fly around during different time periods and noticed different speeds, different altitudes, and some airports.

The first big accomplishment (in my opinion) was changing the color of the line to ground. This happened about a week after I first started playing with google earth, and now, I was really dissecting HWB's script. Dad suggested that I should make a new kml style for each color I wanted. I picked three colors, so I made the styles ToGroundLow, ToGroundMid, and ToGroundHigh. These styles hold attributes like line color, width, and fill color. Next, I made a function (subroutine) which returns a string with the style title. It uses if and elsif statements that evaluate the altitude of each plane. I had a funny problem with this part because I was using the $ALT variable which comes directly from the split function that reads the raw data, when I should have been using the $altitude variable which comes from the $alt variable after a hash is applied to it. I still don't really understand those hashes, but I found this issue by printing both variables as well as the style each aircraft was assigned and seeing which altitude variable matched the style. I decided to leave this debugging print command in because it looks cool. After I learned that kml uses AABBRRGG format for color instead of RRGGBBAA, I got the colors looking right and it worked!

July 8 *The day I began this document

Summary:

- Duane setup a unix file server which I got set up on
- Learned basic git commands
- Slight change to make hvb-atc.pl simpler to run

Dad made a file server for me to handle the large files. I connected to it with terminal from my windows machine using ssh as well as file explorer using windows file sharing. Now I can edit the script using notepad++ on my computer, download the data and run it through the script through the terminal, and open the kml file in google earth on my windows machine. He also taught me the git commands init, add, commit, status, and log.

The simple change that I made was to make the program work with the file tree on the server. I also made it so that the only argument passed is a date such as 20200201, and then added hours as another argument. This isn't great for other people that want to use my program because they will have to edit the script to change the output and input paths, but it makes my life easier because the filepath is hardcoded and I don't have to type it out. As always, I was surprised that this simple changed made me learn so much about perl syntax and different ways to code the same thing. After debugging and testing that everything works, I committed the script and began writing this document.

July 12

Summary:

- Made new script count.pl

I wrote a script with the goal to count how many unique airplanes flew in range of the sensor every day. In its current state, this program takes a range of dates and reads the raw data for the date range (all raw data must be downloaded, but only the 1:0:1 file). It then returns how many different airplanes were counted on each day, as well as a total for the date range. I've started downloading more raw data to begin comparisons between covid stages. My aim for this sort of sub-project is to have a graph which displays airplane count over a long period of time, but for that to happen, I will either need to download lots more data, or have the script work without downloading the entire file. Also was curious, what is with test1234 in the data files? Also also, I think this script really helped me understand hashes in perl.

July 13

Summary:

- Debugged hvb-atc.pl for no reason
- Fixed dates with count.pl and added day of week to output

While I was at work today, I figured that a short term goal of mine should be to download two months of data (one during covid, one before) and stuff like number of unique airplanes, avg. altitude, number of planes landed(?), etc. I'll probably duplicate count.pl and edit it so that It analyzes altitude. Also I want to make count.pl output to a file that can be read by a graphing program. My edits with count.pl today were to fix the dates. Dad suggested I use POSIX to handle dates. I take the date from STDIN, turn it into unix timestamp format, add 1 day worth of seconds to that number, push it onto the dates array, turn the unix timestamp back into yyyymmdd format, then prints it at the end along with the day of the week that it corresponds to.

I started debugging the hvb-atc.pl script when I tried running it on the data file from the first day. It used a new name for $site, which I accounted for, but the kml output file was like 3kb large. Something was wrong with the endtime, so I played around with it and added print statements to debug it. Eventually dad told me it was probably a timezone problem and I gave up on it and just used a newer data file. However, I now have a line counter for how many lines the script reads in the IC and ID files, as well as a counter of what unix time stamp the script is on, and it also prints the start and end time. I also had an issue with google earth where it always would end on the date 2020/02/02. This just happened because I accidentally saved the data to "my places" in google earth, so once I deleted it, it was fixed.

July 14

Summary:

- Made count.pl able to analyze attitude (put 'a' at the end of cmd line)
- Made count.pl output to a text file in addition to the terminal

So for the altitude, the script will open the 3:1:0 file and read the altitude from each line. Each time it adds it to a variable to get the total, which at the end is divided by the total number of lines to get the average altitude. It also checks for the smallest altitude (which is almost always negative) and the largest (which is usually 100000+ ft). That stuff is returned from the subroutine and printed like the count.

For the second bullet point, I just added an open command for an output text file and had it print the same things to the file, but separated by commas. Hopefully I can plug this file directly into some graphing software (I was thinking google sheets, but dad might have something better).

While doing all this I've been downloading lots of data. As of writing this, I have all of February 2020, and I just started on December 2019. I plan to use these for my during-covid and pre-covid sources (although it might help to use something later like march or April for during covid). While looking at the results from February 2020, I discovered something very strange. The last half of the month has the same mins, maxes, and counts, though the average altitudes do change. I confirmed the kml files were different, so it might be that I have some bug in my program, although I'd be surprised.

July 27

Summary

- Added script “getalts.pl”
- Added script “getdistribution.pl”
- Added directory in results “altdist”
- Changed the format of result files
- Added script “altdistro.gnuplot” (inside altdist/)
- Added makefile in altdist/

The new script getalts does a similar thing to the count script. It reads the 3:1:0 file, but it just returns the altitudes from each line. It prints this list of altitude values to a file called [date]-alts.dat. The next new script, getdistribution, will read from this file. It will return the distribution for the set of altitudes in a variable bin size (I started with 1000 ft). This return file is called [date].dat, and it’s inside of the new directory altdist. The final new script, altsidstro.gnuplot, reads this file. Dad basically wrote the gnuplot script for me, but all it does is graph the distribution. To make this simpler (somewhat), dad showed me how to make a “makefile” which would run all three scripts in order after being given a date. At this point, I can use the makefile to pick any date I have and see a graph of the altitude distribution of that day. Besides that stuff, dad also suggested I don’t hardcode file directories and names in my scripts as, despite how convenient it is for me, it makes it hard to run on other machines and is generally a bad practice. I can already tell from the three different distributions that I looked at that there are some interesting patterns to be found, and I know that I should use bounding boxes for the maximums in my original count program.

Earlier this week, I edited hvb-act so that the path is now colored and the line to ground is gone, and I aligned the output of count.pl in the command line so it looks nice. I didn’t get very much work done this week, despite solving all of the challenges from the previous meeting, really just because of work. Speaking of which, I have to get up in 6 hours for a morning shift, so I’d better run.

August 12

Summary

- Added convert command in makefile to copy the pngs of graphs to gifs.
- Added gifsicle command in makefile to animate all .gif files
- Began removing hard-coded directories

I’ll admit, I’ve been pretty bad at keeping up with this document and with my git commits. A lot of work is done late at night and I just forget to document it. That being said, I expect to be doing lots more typing in the next few weeks as I conclude this project.

So me and dad made the graphs use a fixed x and y axis. This way we can combine them in an animation showing the altitude distribution over time. We then added commands to the makefile that would copy the pngs of the graphs to gif files, and then another command would make an animated gif from all of them. Overall, the makefile now reads altitude data from raw files, then gets the distribution of that data, then graphs it, then changers the graph format, then animates them all together. To me, it’s very confusing keeping track of all the scripts and the file locations, so a makefile is really nice. Speaking of file locations, it’s shocking how much hardcoding of directories I did. It’s *super* convenient, but it’s impossible for it to run on a different computer. So, I’ve began going through all my scripts and the

makefile to remove any hardcoded directories and making them command line arguments. I've also began to plan my readme and my report.

August 15

Summary

- Changed the altitude distribution script to read the average altitude for every unique hardware ID
- Created 'better_histo.pl' which creates a stacked histogram of unique airplane count over time

After meeting last Friday, it was suggested to me that my graph could be more meaningful if I count unique airplanes instead of each altitude transmission, given that the transmission rate could be inconsistent, and planes flying different trajectories would transmit different amounts of times, therefore being weighted differently. Me and dad decided that I should use each planes average altitude over the whole day. While I don't fully understand this part, dad helped me set up referenced hashes (I think that's what they're called) so that each altitude value for each plane was stored, then averaged at the end. This script now takes a lot longer to run because of the Statistics module, but I do think the data is more meaningful.

Following this idea, I also received the suggestion to make a count of airplanes over time. Dad helped me a lot with the same referenced hashes in this script, but it basically prints a time column and a count of airplanes at certain altitude levels. The time and altitude bins can be adjusted, but the .gnuplot file will have to be changed to fix the labeling on the altitude bins. This graph shows lots of data, and I think it will be very useful. Just like the old graph, this graph can be created by running the makefile in the /betterhisto directory.

August 18

Summary

- Finished report
- Wrote reproduction instructions

At this point, my project is mostly over. I've produced what I hope is meaningful data, analyzed said data, and answered a question. My report compares two average days before and during covid. My reproduction instructions have a simple step-by-step guide to make some data and they also go into detail describing every script. If I were to write it over again, I wouldn't write as much. I'm definitely happy with where I've gotten in this project. There's probably more I could do and more scripts I could write, but I still got a lot out of working with this data.

August 23

Summary

- Published project to github
- Revised report

I edited my report after receiving some feedback from the meeting, and I've now published the project onto github. I've been told this project will be a good factor in interviews to display my passion for programming and data analysis. At this point, I'm very happy to have successfully found patterns and conclusions from this heap of air traffic data. Thank you to my dad and everyone in the Caida group that helped me with this project and giving me this amazing opportunity.