A Sub-Group Comparison Tool

Megan Tabbutt

CS 765 Final Project 12/15/21

1 Introduction

Here I describe a tool I built as part of the final project for CS 765. The tool is a prototype attempting to solve and explore the sub grouper problem. That is: we want to find and validate specific sub groups in large data sets as well as find which are interesting. Specifically, we want to be able to tackle this problem from the visual angle. This is accomplished both through graphics and tables. The prototype is fairly robust and fully functional, albeit somewhat limited in generality at the moment. Two data sets were used; one to build the tool, and the other to validate and evaluate the performance of it. All in all, it worked wonderfully and I got some very interesting results out of the data in the end. Read on to see what they were!

1.1 Background and Motivation

The problem of sub group comparison is well known, especially in the context of Data Science, and is ubiquitous. In the current state of an ever increasing amount of data to be analyzed and in the time of Big Data; the problem of finding interesting subgroups or finding valid subgroups becomes untenable by hand. You may have some prior knowledge and/or expertise that helps reduce the possible choices, but still, most data sets are too large to try every possible combination. Consider a simple data set with 100,000 entries each representing one person in a city. Each entry has an age represented as an integer from 0 to 100, a height in centimeters, from 0 to 200cm, and a categorical variable denoting weight class (underweight, perfect, overweight, or obese). Let's say you want to compare height and age, but want to make a cut on weight class either to include one, all or any combination in between. The number of possible plots I could make then for height vs age would be (4 choose 4) + (4 choose 3) + (4 choose 2) + (4 choose 1) = 15.

Fifteen different plots to analyze to look for specific trends is already pushing the bounds of what a person is going to reasonably be able to do and get a meaningful outcome and that was for a very modest amount of choices: a single categorical variable, with only 4 options. This not even to mention if you want to cut on one of the other two variables that have 100 and 200 choices. You could of course make some obvious scaling decisions. Is anyone going to be 1cm tall? No. If you select the sample of people between 198 and 200cm, that will be incredibly small. So you might be able to scale some of these variables back based on prior knowledge, but still the problem is extremely poor in scaling.

1.2 Proposed Solution

So, what can we do? We want a tool that will do some of this heavy lifting for us. It should be able to help us make decisions about which subgroups are a) valid to choose, and b) interesting to choose. This tool should help enable us to use prior knowledge about the data set or not. The tool will start in a data exploratory phase where we can find correlations between variables and summary statistics for columns. Next we can start looking at sub groups of that data and exploring whether they were valid choices and interesting choices. Finally, given all of that knowledge we can make multivariate visualizations to compare the interesting sub groups we found in the previous steps. The detailed implementation of the code and specific steps with examples are all explored more in the following section. There are two tutorial notebooks available as well to help with this.

2 Implementation

The tool was created as a Python class that could be imported into your local environment. All of the examples here as well as the tutorials provided in the GitHub repository are using Jupyter Notebooks. One feature of the code is that the dependencies are all very standard Python libraries readily available. While the tool is quite complex in nature and has many different features, there is still more generalization that could be implemented in the future. This will be discussed more in the Limitations and Future Work section.

The tool is broken down into three different concepts of how the user may want to explore and analyze their data. First, is data exploration looking at correlations, summary statistics, and clustering. Secondly, you can compare subgroups based on summary statistics and correlations. Finally, two special plotting routines are provided to compare higher-dimensional data. In the next three sections we will provide examples of all the features of the code.

The first data set is used in the development of the tool and will be discussed here in the implementation section. It is the American Community Survey's Public Use Micro-Data Set (PUMS) from the US Census. Specifically I used the 5-year set ending in 2019 considering only the state of Wisconsin, which has 300,000 entries. All of the plots shown in the following sections, in addition to more can be found in the first tutorial notebook in the Github Repository. There will be a second data set and tutorial notebook that I used as validation of the tool presented and discussed in the evaluation section. This data set was completely new to me, so I had no prior knowledge about it. I used the tool to explore and discover interesting stories.

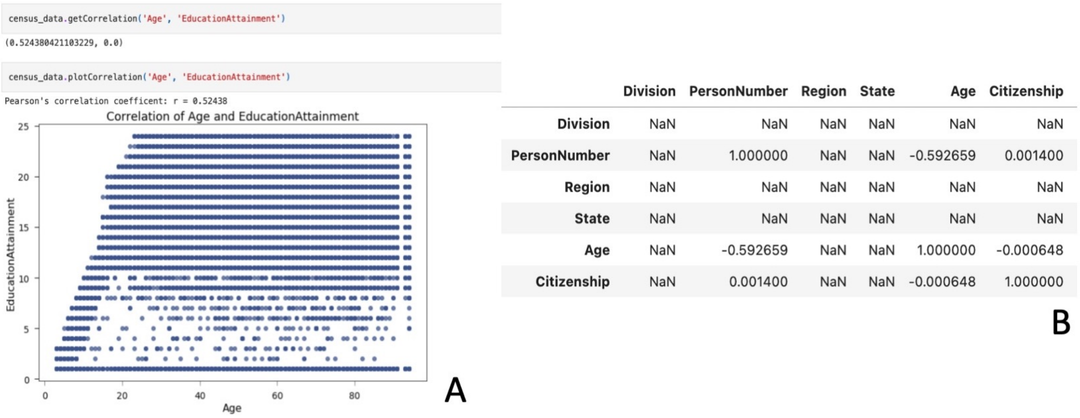
2.1 Part 1: Data Exploration

2.1.1 Correlations

The first use that we can imagine for this tool is a data exploration exercise. Perhaps you want to gain some intuition for the data and get a general overview. Since the implementation is based off a class structure, when you create an instance of the class, you need to provide an argument to the method that is the path to a csv which can be read by Pandas. Specifically, the program assumes that the second row of this are the column headers and the first row are the data types (Categorical, Ordinal, Interval, or Ratio). The specification of the data types is important for the plotting routines later as well as the subgroup comparison methods. Here we assume that the user of the program will do some pre-processing on the data. Since we are viewing this program as a tool that would be used by a data scientist as part of their work flow we think that is reasonable.

One of the first things that you might want to look at are correlations. The program can calculate and plot the Pearson's Correlation Coefficient for the data, as seen in Figure 1a. You can also explore all correlations that are possible between all pairs of variables in the data set. This will be provided as an NxN matrix where only the lower or upper triangle is unique see Figure 1b. From this matrix of all possible correlations you can either pick out the largest (in magnitude), or all above some threshold. For instance, in development we found a high, non trivial correlation of

0.97, but the two variables were ”Time Of Work Departure” and ”Time Of Work Arrival”. This of course makes sense, most commutes are going to be an average length of time. While this makes sense it might not be interesting from a data analysis perspective, so we can instead lower the threshold to 0.8. Then we find some more interesting pairs such as ”Age” and ”Marital Status” which also makes sense but might be more meaningful. We also pull out more trivial examples such as ”Wages” and ”Income” being correlated. Here we can see that this data exploration step can also be really useful in identifying redundant information in the data set. Perhaps both columns are not necessary if they are representing the same information and you could choose only to consider one.



```
census_data.getCorrelation('Age', 'EducationAttainment')
```

```
(0.524380421103229, 0.0)
```

```
census_data.plotCorrelation('Age', 'EducationAttainment')
```

```
Pearson's correlation coefficent: r = 0.52438
```

	Division	PersonNumber	Region	State	Age	Citizenship
Division	NaN	NaN	NaN	NaN	NaN	NaN
PersonNumber	NaN	1.000000	NaN	NaN	-0.592659	0.001400
Region	NaN	NaN	NaN	NaN	NaN	NaN
State	NaN	NaN	NaN	NaN	NaN	NaN
Age	NaN	-0.592659	NaN	NaN	1.000000	-0.000648
Citizenship	NaN	0.001400	NaN	NaN	-0.000648	1.000000

Figure 1: A) A Pearson’s correlation coefficient calculation along with the visualization of that correlation. B) A subsection of the full NxN matrix of all possible correlation pairs. Note that only the lower or upper triangle is unique.

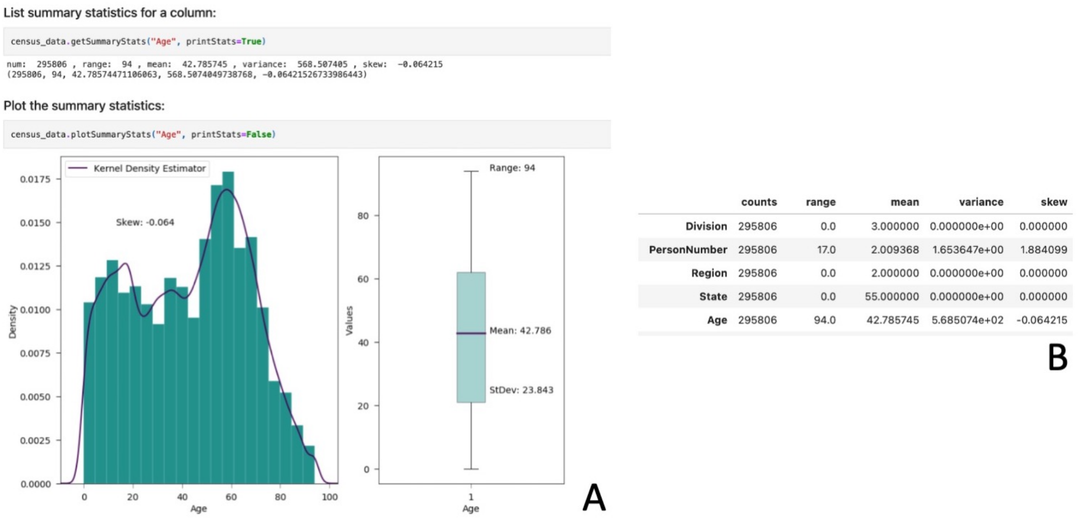
2.1.2 Summary/Distribution Statistics

The next thing that you might want to look at in the data set are the summary or distribution statistics for various columns. This is especially important in non-complete data sets and for columns that don’t have values for every entry. For example in PUMS there is a column for ”Year of Naturalization”, which is blank for most people. If I want to use a sub sample of the data based on that column, I want to know about its counts relative to the whole set. The first element to look at is to pull and plot the summary statistics for a specific column in the data set as seen in Figure 2a. Note that we are also plotting the Kernel Density Estimator to allow for a smoothing visual. This also helps identify modality of the data set, irregardless of binning choices. The statistics that are available to pull are: counts, range, mean, variance, and skew.

You can also generate a list of all the summary statistics for all columns as seen in Figure 2b. This is particularly useful as an overview of the data set. Once you have an overview for the data you might want to compare two columns against each other. I have a plotting routine for this as well, seen in Figure 3.

2.1.3 Clustering

The last thing that you might want to do in terms of data exploration is to look at the clustering of the columns based on these statistics. This will also aid in a point mentioned during the correlation section, that there might be columns that are very similar, and thus redundant depending on what



List summary statistics for a column:

```
census_data.getSummaryStats("Age", printStats=True)
```

```
num:  295806 , range:  94 , mean:  42.785745 , variance:  568.507405 , skew:  -0.064215
(295806, 94, 42.78574471106063, 568.5074049738768, -0.06421526733986443)
```

Plot the summary statistics:

```
census_data.plotSummaryStats("Age", printStats=False)
```

	counts	range	mean	variance	skew
Division	295806	0.0	3.000000	0.000000e+00	0.000000
PersonNumber	295806	17.0	2.009368	1.653647e+00	1.884099
Region	295806	0.0	2.000000	0.000000e+00	0.000000
State	295806	0.0	55.000000	0.000000e+00	0.000000
Age	295806	94.0	42.785745	5.685074e+02	-0.064215

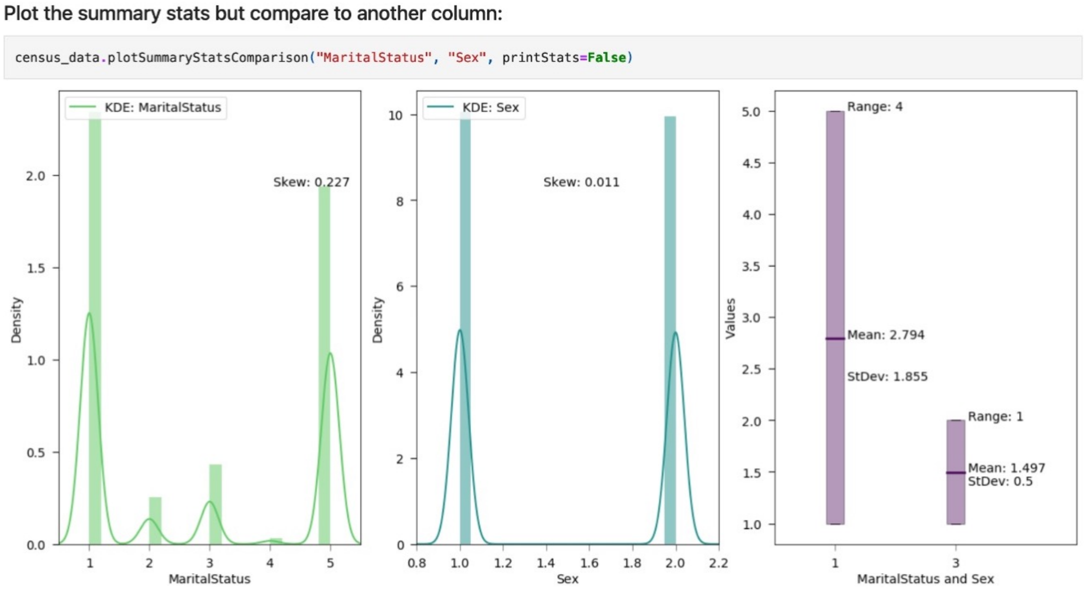
Figure 2: A) Summary or Descriptive statistics for a single column as well as the visual including the Kernel Density Estimator. B) A portion of the table listing the summary statistics for all columns in the data set. This can be particularly useful in understanding the data set and validating future graphing choices for selections.

analysis you are doing. For the clustering function I am using a Gaussian Mixture Model based on the summary statistics of each column. The user is able to specify the number of clusters. Some interesting patterns come out. For instance, in Figure 4 you can see one of our clusters was ”Year of Naturalization” and ”English Speaking” which makes sense. However, ”Wages” and ”Total Income” were clustered separately even when I varied cluster number. Despite being similar distributions, the range and variance are very large compared to some of the other columns since it is not normalized. Normalization is something that may be beneficial to your analysis and an example of normalizing the data and then proceeding through all the previous steps is shown in the first tutorial notebook.

Along with normalizing the data set (an example output can be seen in Figure 5), you can also pre-select a subset by trimming the data set manually before loading into the class instance. Finally, you may want to compare two columns based on the clustering information, which would follow the same routine as Figure 3. There is an example of this in the first tutorial as well.

2.2 Part 2: Comparing and Exploring Sub Groups

A lot of the smaller functions of the tool were aimed at the data exploration part and while that is certainly a huge part of an analysis undertaking, the next two parts are the more interesting aspects of the code as they start to drill into the sub group exploration problem that was presented at the start and is at the heart of this project. The first aspect is to compare sub groups. I wanted a method that will start cutting the data set on various subgroups and looking at the output for comparison. While a full tool would want to be able to make any cuts on any of the columns, this prototype will focus on categorical variables. In Figure 6 we can see an example of all the selections possible on our data set. Each categorical variable is broken up by it’s possible values, for instance ”Sex” could take on ”1” or ”2”. We then identify an independent variable ”Age” and plot out an overview of how the summary statistics for ”Age” would change if we selected ”Sex” = 1.0, 2.0, etc. The importance of this is that we can decide what are valid cuts to make by comparing the



Plot the summary stats but compare to another column:

```
census_data.plotSummaryStatsComparison("MaritalStatus", "Sex", printStats=False)
```

Figure 3: The summary statistics comparison for two different columns. While the skew was small for both, its clear that one is more skewed than the other, and that while Sex is populated pretty fairly, Marital Status is not.

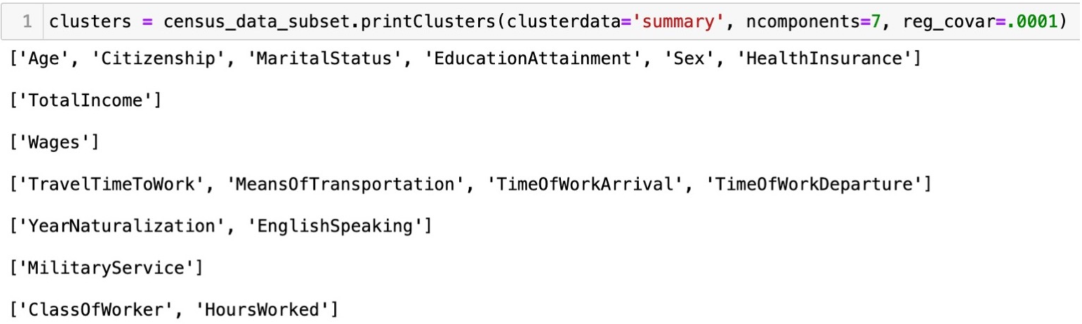
summary statistics of the cut to the data set as a whole (marked by the black dashed line). For instance, cutting on either ”Sex” value, leaves a large number of counts and a similarly skewed, ranged, and varied data set. The same is not true, however, ”Military Service” = 1. This chart is also a great look up chart after making some of the final plotting choices to explain why some sub sets may be more noisy, or have other less desirable features. You may also select a subset of columns to try, an example of this is shown later.

We can now start exploring sub groups in our data set. I chose to pull out the sub group with ”Class of Worker” = 1. Which based on the skew and mean looked like a fair assumption. I then plotted the correlation between ”Age” and ”Education Attainment” as seen in Figure 7. The left plot is for the full data set and the right only for people who are an ”Employee of a private for-profit company”. What we can see is that the correlation disappears in the right plot meaning that it was coming from the other part of the data that we cut out.

There are seemingly endless iterations of this type that you could preform in order to look at the data. Some of that could be better automated by the tool in future versions and this is discussed more later. However, as is, the tool still does a great job at enabling good choice making for simple subgroups and exploring whether certain sub groups are valid (through summary statistics) and interesting (through correlations).

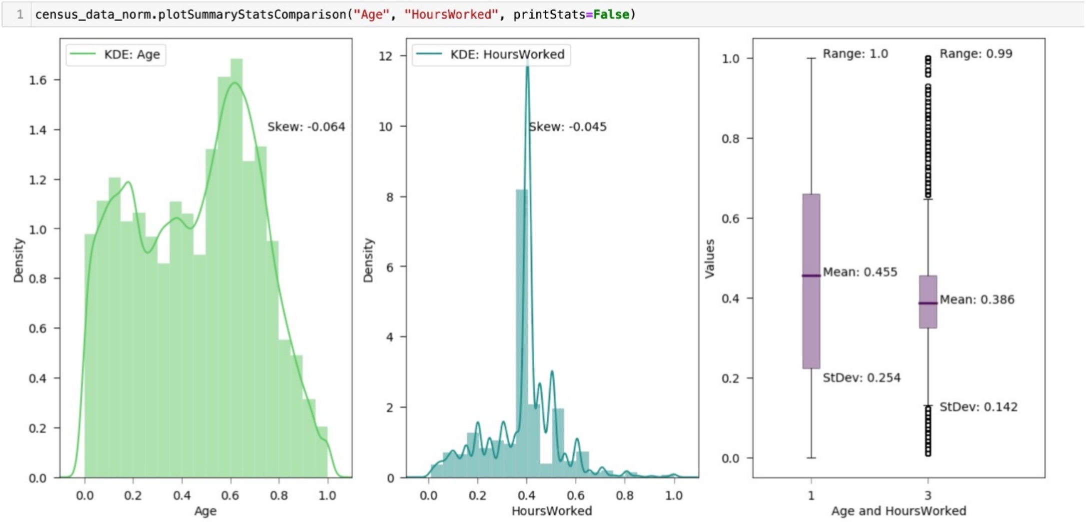
2.3 Part 3: Plotting the Data and Comparing the Subgroups

The most interesting part, in my opinion, is to use the two multi-variate plotting routines to compare the sub groups that we found interesting in the previous section. Two special plotting routines are provided, and more will be made in the future. The first routine can be seen in Figure **??**. The x variable should be a quasi-continuous variable. The main y variable should also be quasi-continuous. The second y variable of interest can be any type (categorical, ordinal, interval, ratio). However, caution should be used with un-ordered data as the color bar might imply some



```
clusters = census_data_subset.printClusters(clusterdata='summary', ncomponents=7, reg_covar=.0001)
['Age', 'Citizenship', 'MaritalStatus', 'EducationAttainment', 'Sex', 'HealthInsurance']

['TotalIncome']

['Wages']

['TravelTimeToWork', 'MeansOfTransportation', 'TimeOfWorkArrival', 'TimeOfWorkDeparture']

['YearNaturalization', 'EnglishSpeaking']

['MilitaryService']

['ClassOfWorker', 'HoursWorked']
```

Figure 4: An example output of a Gaussian Mixture Modelling clustering analysis done on our data set with 7 clusters.

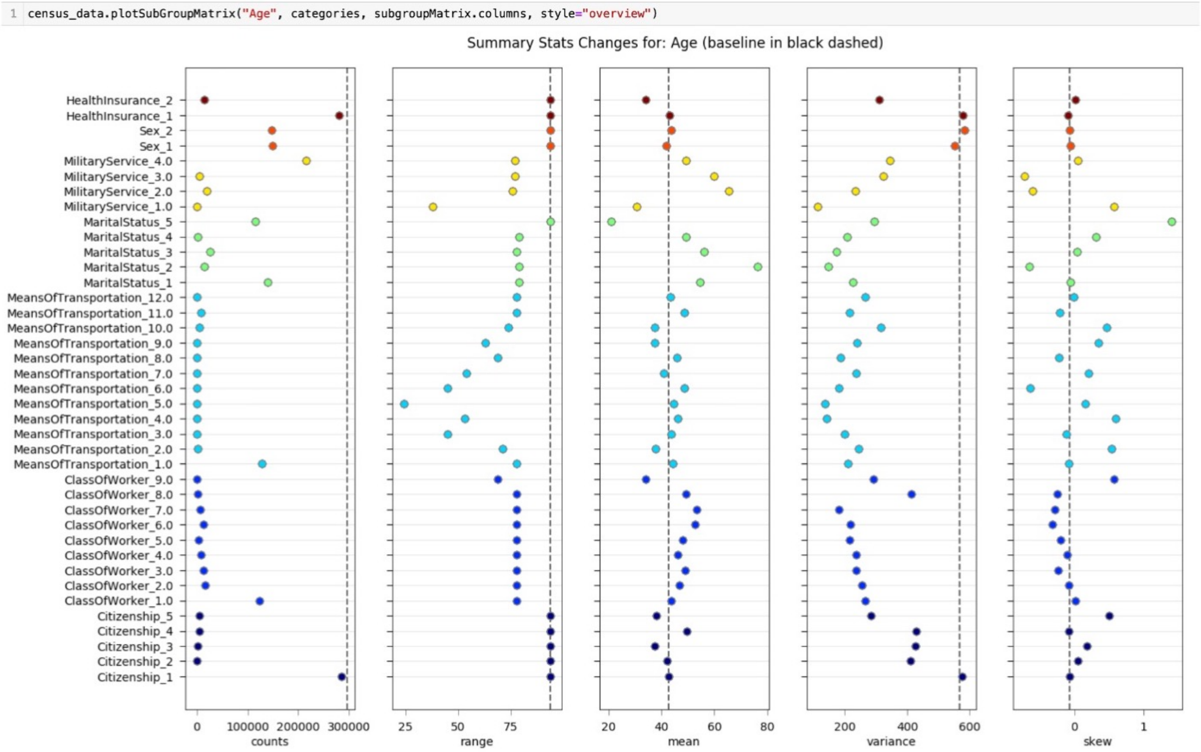


```
census_data_norm.plotSummaryStatsComparison("Age", "HoursWorked", printStats=False)
```

Figure 5: An example of a normalized data set, while this obscures the axes a bit more, it makes the clustering more valid.

ordering. In Figure 8 we can see some very interesting trends. First, you can pick out the three major shifts of working: morning, afternoon, and night. Secondly, when you compare across plots, it's clear that if you are an employee of a private company you are more likely to be coming in on those specific times since the clustering is tighter and there are fewer points in the in-between times. However, there doesn't seem to be too high of a trend with education attainment.

The second multivariate plotting routine considers the average of the main y variable in bins of the x variable. Here the first y variable could be anything you want as long as an average makes sense, that is most likely interval or ratio. The x variable could be anything from categorical on. The second y variable it best in categorical form and only a handful. While you could plot more, the colors start to blend together as well as the trends past about 5-6 categories. In this second plot I chose the sub groups of having or not having health insurance, and then plotted the educational attainment vs age for both males and females. An interesting trend can be seen in those with health insurance, that the females are on average attaining the same or higher levels. This trend disappears in those without health insurance, however, if we go back to our summary statistics from Figure 6 we know that that sample isn't as statistically valid. It has very low counts, and



```
census_data.plotSubGroupMatrix("Age", categories, subgroupMatrix.columns, style="overview")
```

Figure 6: After choosing the variable column of interest, "Age", we can cut the data set according to the possible values in the categorical columns to see how the summary statistics change when compared to the uncut data set (black dashed lines).

higher variance when compared to those with health insurance. So, it might be that the trend still exists but is washed out by noise.

Here we have explored the functionality of the tool in full and seen most of the features that it has to offer. All of the functions are represented in the first tutorial notebook, and can be seen in more detail there. We were able to effectively explore the data set, make informed choices about sub groups, find interesting sub groups, and validate them. We were also able to make many plots comparing those sub groups and find trends and stories in the data. Next we will turn to the evaluation section where we will explore another data set to test the tool and see how it does on a data set for which we have no prior knowledge of.

3 Evaluation

The evaluation of the tool and how well it addressed the problem at hand was done in two parts. First I tested the tool and the actual implementation of the code on a completely new data set that I had not seen before. Secondly, I took a more abstract set of evaluation criteria, laid out in Munzner to determine our success. Where there were shortcomings I will mention it.

3.1 Testing an Unfamiliar Data Set

The data set that was used was the American Time Usage Data set provided to us from the first design challenge in class. I have never worked with this set before, and the only pre-processing that

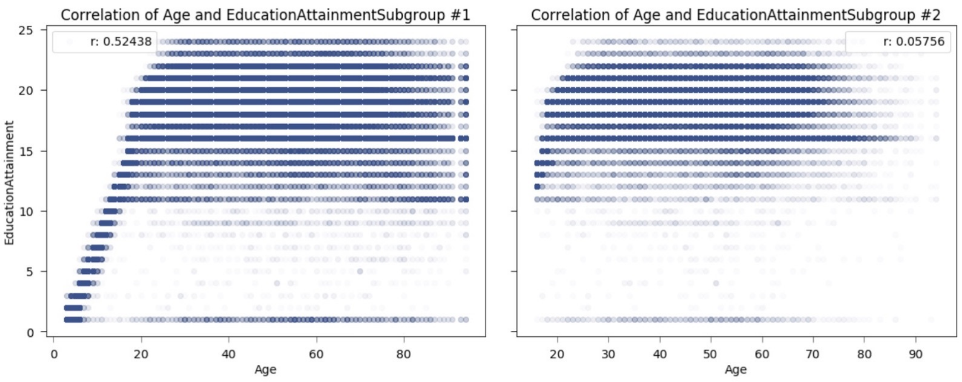
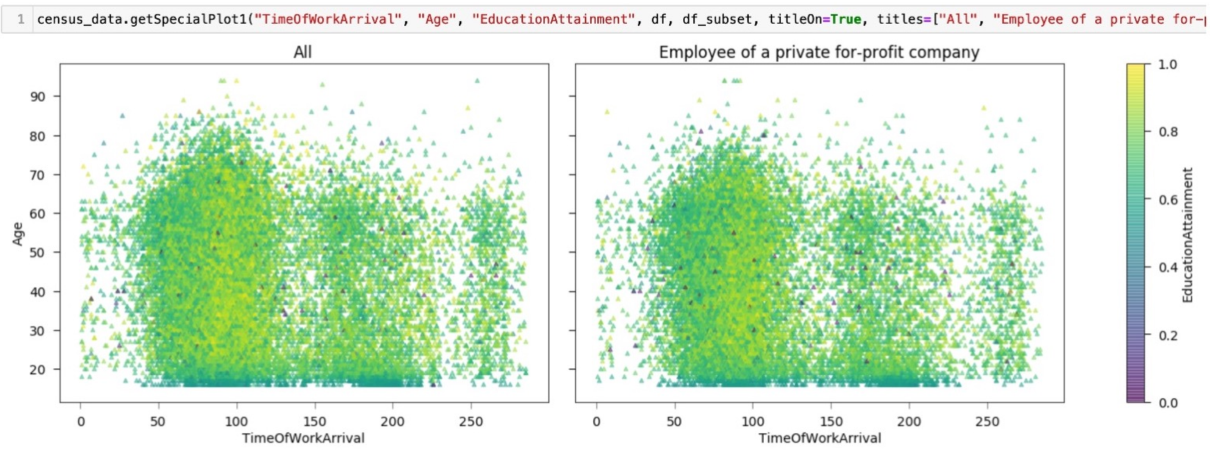


Figure 7: Comparing Education Attainment for the full data set (left) and the sub group of people who are an ”Employee of a private for-profit company”. The correlation disappears in the second plot, meaning that it is tied heavily to the set of data that is not in our cut.



```
census_data.getSpecialPlot1("TimeOfWorkArrival", "Age", "EducationAttainment", df, df_subset, titleOn=True, titles=["All", "Employee of a private for-
```

Figure 8: Comparing Education Attainment for the full data set (left) and the sub group of people who are an ”Employee of a private for-profit company”. The three main working shifts can clearly be seen in the clustering.

was done was reading the column descriptions so that I could rename them more intuitively and assign data types for each column. I followed a similar exploration order as before.

I first explored correlations, looking for relatively large ones. In particular I found that the amount of time people spent working was negatively correlated with how much time they spent socializing as seen in Figure 10. I didn’t assume this a priori, but it makes sense. Another interesting correlation was between education and education years, it seems that those columns are tracing the same idea but in different manners, they are mostly redundant.

I next examined some of the summary statistics and distributions of columns as seen in Figure 11. What was interesting here was that I tried to normalize the data set by dividing the time columns by 365 to get hours spent on activities per day. However the double peaks in the working distribution almost certainly should correspond to zero hours and about eight hours. The second peak is quite low though, about 1.5 hours. This tells us that either something is wrong with the data or more likely I misunderstood the normalization and perhaps the data was only for a couple months, not the whole year.

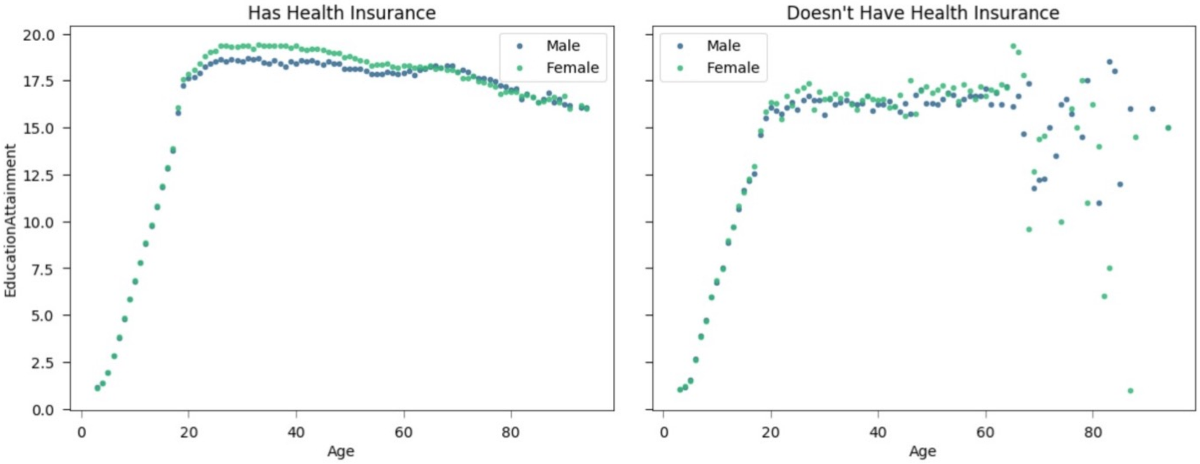


Figure 9: Comparing Education attainment vs age for males and females. The left plot is for those with health insurance and the right plot is for those without health insurance. While it seems like the clear trend in the left plot that goes away in the right, there are actually much lower counts and higher variance in the right sub group meaning that the trend might be washed out in the noise.

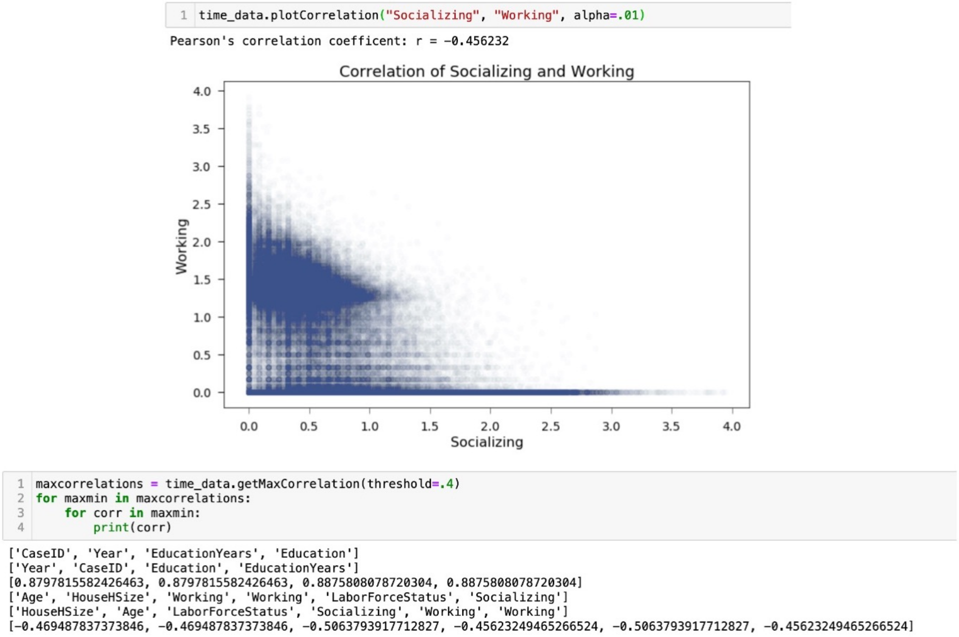
I next wanted to examine sub group selection but only for a few categories, ”Labor force status”, ”Sex”, and ”Region” as seen in Figure 12. Both ”Sex” and ”Region” seem to be pretty fairly distributed among the choices, as one would expect, but ”Labor Force Status” is not. This I found interesting and wanted to explore some more.

I first explored how the correlation of time spent on sports/exercise vs age changed between those that were currently working (left) and those not in the work force (right) as seen in Figure 13. The results are not surprising. First we can see that most people who are not in the work force are children and those of retirement age. We can also see a negative correlation between exercise and age in both but especially in the the right plot.

Finally, I wanted to make two really nice multi-variate summary plots based on what I had learned about the sub groups in this data set. The first plot shows the correlation between time spent on house hold care and age as seen in Figure 14. The color bar is denoting Labor Force Status in the following order from 0 to 1: Not in labor force, unemployed - looking, unemployed - laid off, employed - absent, and employed - working. The first feature that jumps out is the same as above, the yellow is concentrated on the wings because those are the people not in the work force. Interestingly, both have a bit of a central peak around the parenting age which makes sense, and the females appear to be higher on average which one might also assume.

Finally, I used the second special plotting routine to look at the correlation between time spent on sports/exercise and age for both males and females split but the labor force status as seen in Figure 15. Interestingly, most of the high outlier points seem to be unemployed persons. You can also see a steep downhill trend with age early on across all labor categories and for both sexes. The males also have a higher average than females.

Overall, I would say that the tool worked splendidly. I was able to explore the data, get a good intuition for it. Find interesting and valid subgroups and make some really nice visualizations that showed the multivariate trends I was seeing in the data. Next I will evaluate the tool more holistically.



```
time_data.plotCorrelation("Socializing", "Working", alpha=.01)
```

```
Pearson's correlation coefficent: r = -0.456232
```

```
maxcorrelations = time_data.getMaxCorrelation(threshold=.4)
for maxmin in maxcorrelations:
    for corr in maxmin:
        print(corr)
```

```
['CaseID', 'Year', 'EducationYears', 'Education']
['Year', 'CaseID', 'Education', 'EducationYears']
[0.8797815582426463, 0.8797815582426463, 0.8875808078720304, 0.8875808078720304]
['Age', 'HouseHSize', 'Working', 'Working', 'LaborForceStatus', 'Socializing']
['HouseHSize', 'Age', 'LaborForceStatus', 'Socializing', 'Working', 'Working']
[-0.469487837373846, -0.469487837373846, -0.5063793917712827, -0.45623249465266524, -0.5063793917712827, -0.45623249465266524]
```

Figure 10: The correlation between time spent socializing and working is negative and significant, which seems intuitive. Other large correlations are listed.

3.2 The Munzner Criteria

Munzner defines four different encompassing levels of evaluation: domain situation, data/task abstraction, visual encoding/interaction idiom, and algorithm. I have already evaluated the algorithm in the above section, it worked magnificently on a completely different data set.

For me the domain situation was that I wanted to target data scientists or enthusiasts who wanted another tool available to them (in Python) to explore their data set. I think in the end the user is even more general, since the tool is so easy to deploy in Python and requires no special packages, even a very elementary Python user could utilize it. This could be a bit to its detriment however, as it's a prototype at this stage it lacks the flexibility that a true data scientist might want. This would be a great area of improvement for future work.

The task was to create a tool that can find and evaluate interesting and valid subgroups. I think in general this tool does do that task well. I also think that it would be possible to come up with more specific tasks that this tool can not handle in it's current state. It can not, for instance, bin anything but categorical variables on the x-axis for the second special plotting routine. These are specific tasks and features that need to be added in future work. Another task that is not addressed in this prototype is an automatic method for validating a sub group. The user needs to choose that by hand. While that may be fine in some situations and applications, it won't scale well to big data, but would be easy enough to implement a solution in the future.

Finally, we can consider the visual encoding. If we limit ourselves to the tasks that the tool can currently accomplish I think the visual encoding worked very well for the later special plotting routines. Where the prototype might come up a bit short is on the correlations. There is a lot of data saturation in color for the large data sets with only a few possible values. A new design solution to improve this could be good.

Overall, I think that the design of the prototype especially considering the time constraints, is well executed and already applicable to many people. However, I think that there are many places

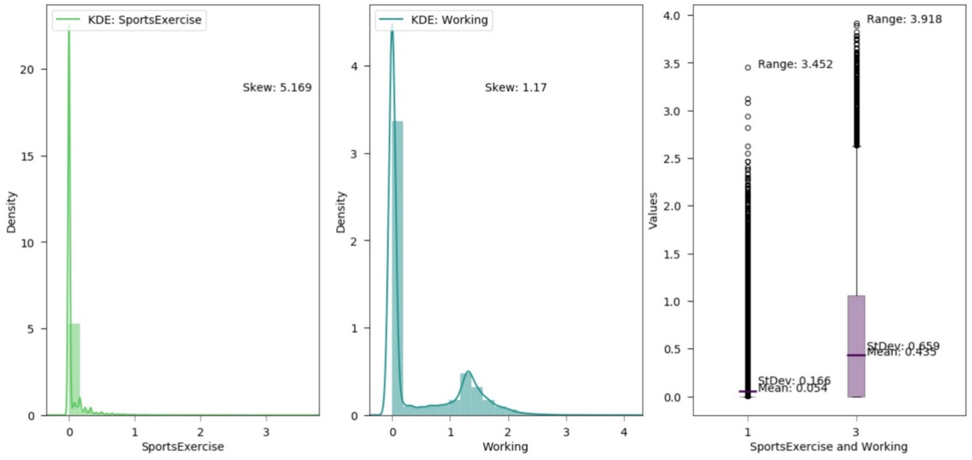
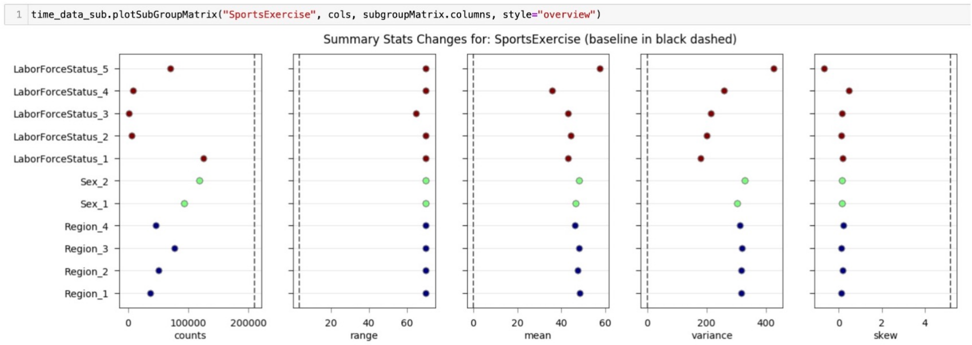


Figure 11: The summary and descriptive statistics for two different columns: time spent working, and time spent on exercise/sports.



```
time_data_sub.plotSubGroupMatrix("SportsExercise", cols, subgroupMatrix.columns, style="overview")
```

Figure 12: A comparison of how the summary statistics for time spent on sports/exercise change when various sub groups are selected.

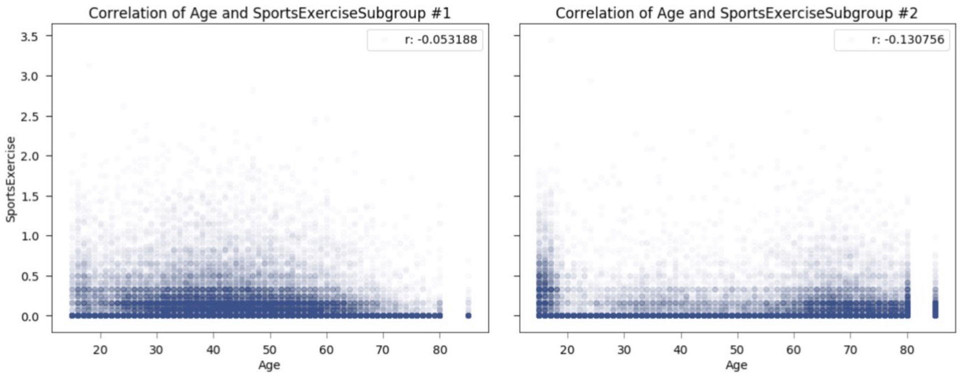


Figure 13: The correlation between time spent on sports/exercise and age for those currently working (left) and those who are not in the work force (right).

to go from here and much future work that could be done to improve it and turn it into a great general purpose Python tool for both the Data Scientist and average user.

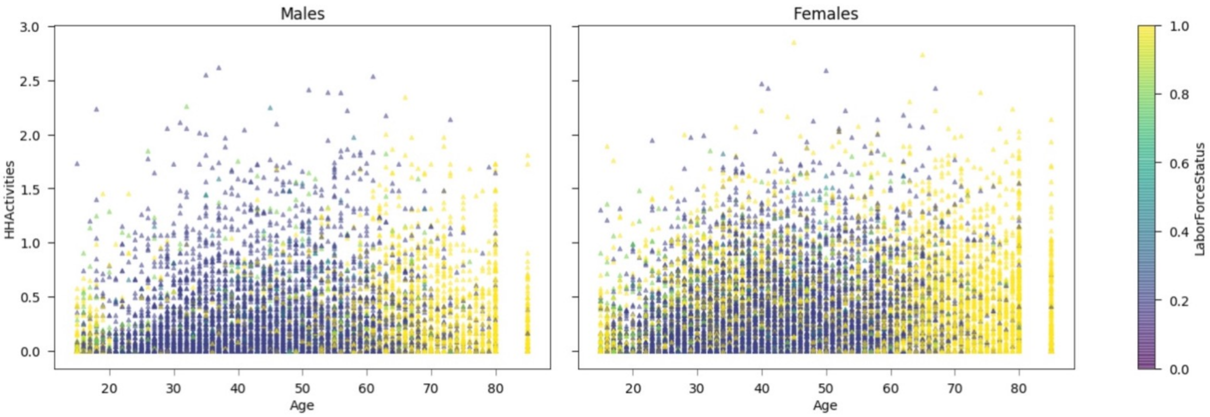


Figure 14: The correlation between time spent on House Hold activities and age, for both males and females. Colored by Labor force status as follows from 0 to 1: Not in labor force, unemployed - looking, unemployed - laid off, employed - absent, and employed - working.

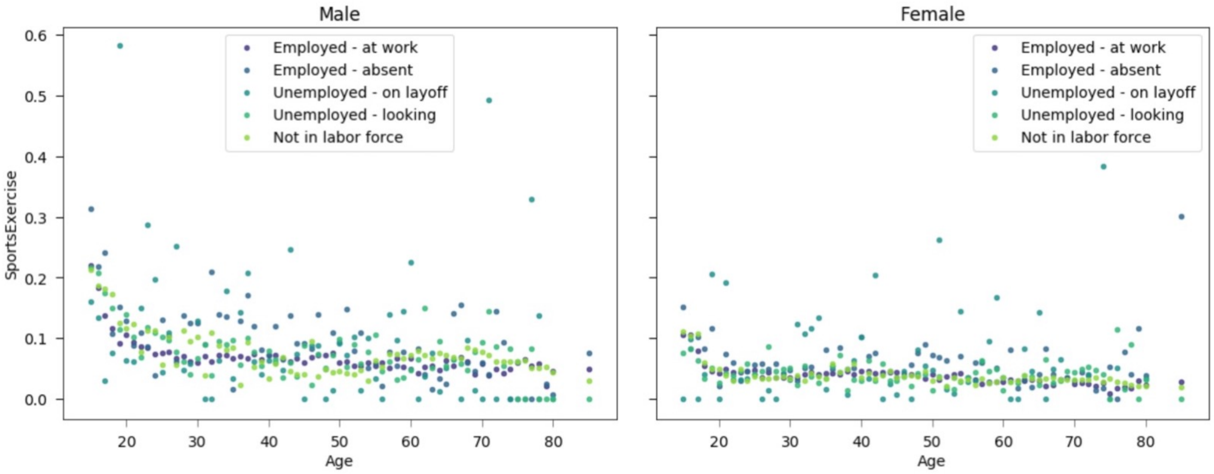


Figure 15: The correlation between time spent on sports/exercise and age for males and females, categorized by their labor force status.

4 Limitations and Future Work

Some of the limitations and future work have already been mentioned, but most of it can be summarized as the tool needs more generality and flexibility. There are many functions, such as the sub group enumeration method that are limited in scope right now and should be expanded in future work. This method in particular should include a way to specify cuts on non-categorical data types. It would also be great to include multiple cuts. Initially, I had designed the system to make a large table of many cuts behind the scenes and only return the interesting and valid solutions, but that became untenable quickly and would take more sophisticated methods to optimize. In summary, there are many edge cases that could be tied up.

There are a few plots that could be improved on still, such as the correlation plot. This plot saturates quickly for variables with few choices as can be seen in this paper. There are also plots with automated annotations that overlap in some places, this could be cleaned up. Finally, the special plotting routines could be expanded in number and variety of type of plot as well as the types of data that are allowed as input. But overall, it is a sparkling prototype that does the fundamental job that I set out of for it to do, while still having much room to grow into a very powerful tool in the future.

5 Conclusion

In closing, while I think that there are endless ways to improve this prototype and turn it into a very powerful data exploration and sub group validation tool, it does work as a prototype well. The user can explore their data, visualize the descriptive and summary statistics of columns as well as compare them. They can cut the data set on categorical variables and compare the statistics of the sub groups. You can cluster the results and look for interesting combinations that way. And finally, you can make very nice multi-variate comparison plots to look for trends and stories in the data. Overall, the tool works, and it works well as a prototype.