**Flight Picker**

**George Perez**

**CSCI 431, Final Project**

**April 22, 2019**

[**1. ALGORITHM**](#_m4oydyr7r5tr) **2**

[1.1 Best of Each Category](#_bz9ht137rljv) 2

[1.2 Best Based on User Selection](#_melbk5sdewsz) 3

[1.3 Best When Combining Categories](#_pj7gghv908yd) 3

[**2. PROGRAMMING**](#_or5zfdgi24jl) **4**

[**3. INTERMEDIATE FILES**](#_12waddl2njrt) **5**

[**4. OS, PYTHON VERSION, GUI**](#_yei7024h6vdt) **5**

[**5. AUTOMATIC RESULTS**](#_vad2kfgmg1h5) **6**

[5.1 Output](#_2nub7s8nj359) 6

[5.2 Testing Evidence](#_624n1dl6va1m) 7

[**6. MANUAL RESULTS**](#_rk3gi85wumq8) **7**

[6.1 Output](#_zceihv3c7eqh) 7

[6.2 Testing Evidence](#_g6k3fv6zj497) 8

[**7. TESTING AND VALIDATION**](#_a9dlc47c684c) **8**

[**8. ACKNOWLEDGEMENTS**](#_16hmgjf23uy2) **8**

# 1. ALGORITHM

To find the best flights overall, Flight Picker employs an algorithm that uses the data found in the rows of the data file. Each row is given a number of ranking points, and the rows that have the most points will be ranked higher than others. Each row is not indicative of a single flight, so a weight must be determined so that each row will be equal as any other row in determining its overall rank. To determine this weight, the flights performed for that row is divided by the number of seats. This essentially converts the row to an equivalent row of one flight performed. The weight for the row is multiplied by the chance of success to determine the row’s rank in the overall list.

To determine the chance of success, the program will take the flights performed divided by flights scheduled. This is done to determine the overall chance of success that flight has of performing. If there are more flights performed than scheduled, the chance of success is capped at one hundred percent to avoid skewing the results. Flight Picker uses this new list of ranked flights as the base ranking system for both Automatic and Manual mode. Additionally, all flights involved were all either from North Carolina or South Carolina, or had a destination of North Carolina or South Carolina.

## 1.1 Best of Each Category

In context of Automatic mode, Flight Picker chooses the best options for each category and displays them for each criterion.

These criteria, following specifications, are:

* Distance (in miles)
* Carrier
* Origin City
* Destination City
* Aircraft
* State Origin
* State Destination
* Month

The options for these criteria were determined by iterating through the rows and getting the unique options available.

To determine the best options for each criterion, Flight Picker uses the ranked flight list determined at the start of the program. The program will iterate through the flight list and categorize each flight as belonging to a specific option. With the new categories, the best options for each criterion are determined. For each option, its average rank is calculated by taking the rank for all rows associated with that option, and dividing that sum by the number of those rows. These newly ranked options are displayed under a tab for each criterion, in descending order. Additionally, the user may click on any individual option and view the rows associated with that option; these are also ranked in descending order.

## 1.2 Best Based on User Selection

The user may select certain criterion to filter by using Manual mode. This mode uses the same ranked flight list as Automatic mode. To filter out the rows, the user is presented with a menu that contains drop down boxes, with the names of each option. These drop down boxes are categorized under each individual criterion and can consist of any combination, further discussed in Section 1.3. When the user selects an option, the rows that match that option will be displayed to the user.

## **1.3** **Best When Combining Categories**

Combining categories pertains to both Automatic and Manual mode. For Automatic mode, the program will display the best options for a smaller pool of flights. These smaller pools of flights will be derived from the first criterion in the combination. The initial combinations of flights are: Month + Carrier, Distance + Aircraft, and Origin State + Carrier. The best flights for the second criterion are determined from the smaller pool of flights, which are then displayed to the user similar to the single criterion.

The design for combinations in Manual mode is as follows: when the user selects an option in more than one criterion, the program will go through the ranked flight list and match all options the user selected with rows in the list, if any exist. Manual mode will allow for any and all combinations of options, regardless of whether or not rows matching those options exist.

# **2. PROGRAMMING**

This section will outline the steps taken to program Flight Picker and the order of priorities taken when developing the program, along with certain specifics about libraries and design.

The initial priority was to determine the design of the algorithm that matched specifications. However, much of the initial development was focused on design of the GUI, data retrieval, and data alteration. This was due to misconceptions about the specifications needed, but once the issues were resolved, the focus shifted back to working on the algorithm. The initial data retrieval tool used in Flight Picker is pandas, without which the program would not get the .csv rows from the project data. However, once the data is initially retrieved from the .csv, pandas is no longer needed as the rest of the data alteration work is done natively in Python.

As more time was spent on development, the main aspect of focus was on preparing the output for the two modes and the selection of options; the design of the GUI was tailored around what was needed at the time to display the required data and options. Additionally, there was much focus placed on designing modules and functions to have a high degree of modularity and organization. Once the overall design of the algorithm was complete, finishing touches were applied to the GUI and organization of code.

The GUI was designed to deliver information about the flights in a compact manner, while ensuring the user is presented with all options without having to scroll or navigate multiple windows.

# 3. INTERMEDIATE FILES

The original data file provided for the program, ProjectData.csv, was altered to remove unnecessary rows and columns. Any row with zero flights scheduled were removed, as the user would have had no opportunity to take a flight in that row. Additionally, the columns that were removed were either not part of the tested criteria, or had no part in neither the score nor success percentage calculation.

There were intermediate files created for the execution of the program, the first group of which being smaller data files to test for consistency and accurate calculations. These files took rows from the larger file, and followed the same format. Once the program was deemed to be functional and accurate, the smaller files were no longer used and instead were replaced with the larger data file.

When the options for the criteria were determined, they are written to .dat files in the data/options folder. These files were created to reduce the load time of the options, and are only written to when the user selects the option to update the data. Each time the program loads, the data files are loaded as the options for the criteria.

Within the data file for the program, there is a column for the aircraft used by the flights taking place from that row. These aircraft columns were displayed as codes within the data file. To make the program more user friendly and readable, the codes were translated using another data file, AircraftNames.csv, that matched the codes with the names of the aircraft. These list of names were returned and written to the option .dat files, and by extension, were provided to the user as the options for aircraft.

# 4. OS, PYTHON VERSION, GUI

The operating system primarily used for development of this project is Linux Mint 19.1 Cinnamon, version 4.0.8. This was selected because it was effective in developing the project and contained efficient methods of retrieving the packages and GUI needed.

Python version 3.6.7 is being used, and was selected due to its writability and library support.

The GUI employed for this project was primarily Tkinter version 8.6. This was chosen due to its prevalent use in the Python development community, and as such, has plenty of documentation and support.

# 5. AUTOMATIC RESULTS

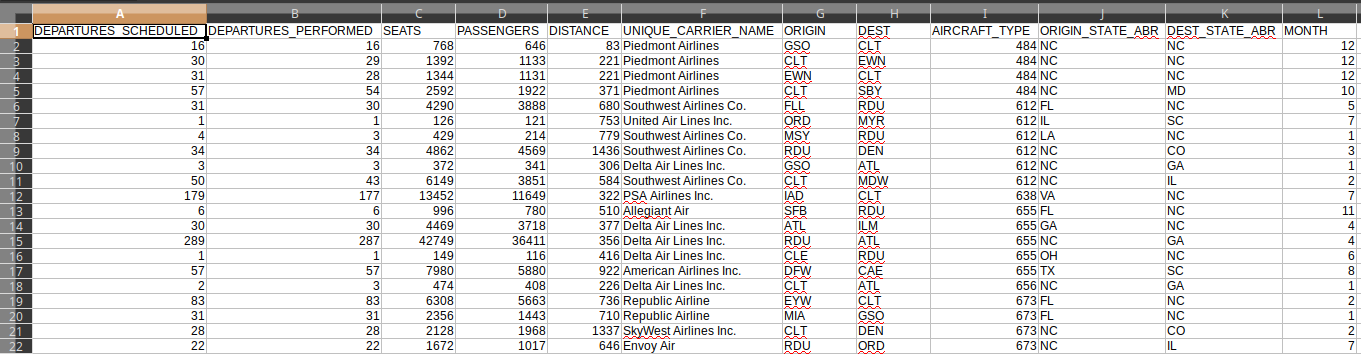
The results of Automatic mode are outlined in this section, along with supporting tests from smaller data files.

## 5.1 Output

Utilizing the algorithm outlined in Section 1, the output for Automatic mode is displayed below.

|  |  |  |
| --- | --- | --- |
| **Criterion** | **Best Option** | **Average Score** |
| **Distance** | 1500-1999 | 166.39 |
| **Carrier** | Frontier Airlines Inc. | 182.64 |
| **Origin City** | ISP | 223.71 |
| **Destination City** | ISP | 223.71 |
| **Aircraft** | A330-3 | 287.81 |
| **Origin State** | PR | 185.3 |
| **Destination State** | ID | 187.0 |
| **Month** | August | 109.95 |
| **Month + Carrier** | Frontier Airlines Inc. | 184.44 |
| **Distance + Aircraft** | A330-3 | 288.39 |
| **Carrier + Origin State** | RI | 218.59 |

## **5.2 Testing Evidence**

One of the smaller test files used for Automatic mode is displayed below:

The results from Flight Picker using the data file above are:

|  |  |  |
| --- | --- | --- |
| **Criterion** | **Best Option** | **Average Score** |
| **Distance** | 400-499 | 149.0 |
| **Carrier** | Allegiant Air | 166.0 |
| **Origin City** | SFB | 166.0 |
| **Destination City** | ILM | 148.97 |
| **Aircraft** | MD-90 | 158.0 |
| **Origin State** | OH | 149.0 |
| **Destination State** | GA | 143.31 |
| **Month** | November | 166.0 |
| **Month + Carrier** | Allegiant Air | 166.0 |
| **Distance + Aircraft** | MD-80 | 149.0 |
| **Carrier + Origin State** | FL | 166.0 |

## 

# 6. MANUAL RESULTS

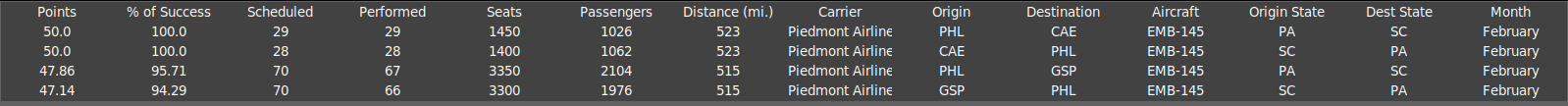
For this section, specific options have been selected from the criteria and their flights will be displayed. As mentioned in Section 1.2 and Section 1.3, the user may select any combination of criterion and their respective options.

## 6.1 Output

Two different scenarios are output below, with different combinations.

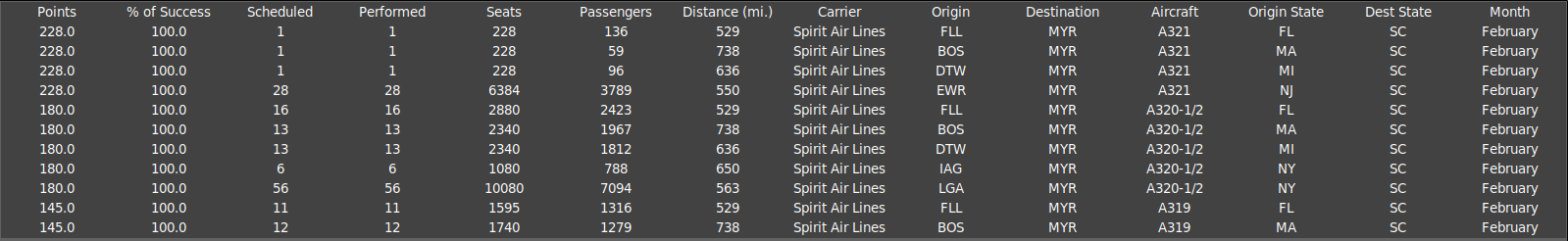
The first scenario has the options selected of:

* Distance: 500 - 999
* Carrier: Piedmont Airlines
* Origin City: None
* Destination City: None
* Aircraft: EMB-145
* Origin State: None
* Destination State: None
* Month: February

These selected options will output four different rows, all matching the options.

The second scenario has the options selected of:

* Distance: 500-999
* Carrier: Spirit Air Lines
* Origin City: None
* Destination City: MYR
* Aircraft: None
* Origin State: None
* Destination State: SC
* Month: February

These options will yield the eleven flights below, which match the options:

# 7. TESTING AND VALIDATION

To ensure the correctness of the program outside of debugging, there were numerous tests done to validate the results of both Automatic and Manual mode.

## 

Automatic mode was tested by running through smaller sets of data and calculating the average score for each option in each criterion. Essentially, the algorithm for Automatic mode was applied and the results compared to what was being displayed. Additionally, each option was tested to ensure the rows matching that option were displayed in the menu accompanying the results.

For Manual mode, the testing involved simply running the program against different criteria and options, and ensuring the rows displayed matched the options the user selected. Many different combinations were tested, and a random button was added to test random combinations of Distance, Carrier, and Month; adding more options to the pool would have drastically decreased the chance of finding any matching rows.

The edge case of an empty data file for either ProjectData.csv and/or AircraftNames.csv was tested, along with the possibility that ProjectData.csv contains only one row. In the event of the former, Flight Picker will give a popup window to the user alerting them of this fact, and will not continue until data is provided. In the event of the latter, the program will simply display data for one single row, meaning all results from Automatic mode will be tailored to that row alone; since the results from Manual mode are dependent on the options within the row, only that row will be shown and can be filtered as normal.

If the data files of ProjectData.csv and/or AircraftNames.csv is not found in the data/ directory, the program will alert the user of this fact and will cease execution until the files are provided.

# 8. ACKNOWLEDGEMENTS

Of note, there are a few acknowledgements to be made regarding recommendations and basic tips.

Ira Daniel Branham made the suggestion of using Tkinter and pandas, which allowed the development to continue at a smoother pace, as less time would have been spent researching these options.

Cashton Christensen gave a recommendation to include a Reset Options button, which was promptly implemented in the design of the program and made testing much faster.

Stack Overflow was instrumental in furthering the understanding of errors in Tkinter and pandas, and deserves an honorable mention.