COSC480 Project Report (ID Number: 31779222)

Introduction

Online gaming has become considerably more popular over the last few years (Sauter and Draschkow, 2017). As a result, several studies by researchers have suggested that excessive online gaming is associated with negative mental health outcomes such as lower quality of life and increased anxiety (Sauter and Draschkow, 2017). Sauter and Draschkow (2017) did a survey on the League of Legends subreddit with about 13,000 online gamers as participants, a significantly larger sample size compared to previous studies on online gaming.

Sauter and Draschkow (2017) survey and data set consisted of questionnaire results about hours of gaming per week, narcissism, general anxiety, social phobia, quality of life, and other demographic information. My project's primary aim was to calculate the correlations between hours of gaming per week (as an independent variable) and narcissism, general anxiety, social phobia and quality of life (as dependent variables). My project also aimed to describe summary statistics for variables within the data set, conduct multiple linear regression, and produce graphs.

Method and Code

Jupyter Notebook was chosen as the integrated development environment for Python, Anaconda was also used.

To run the program the *GamingStudy_data.csv* by Sauter and Draschkow (2017) is uploaded. Then the program is run producing summary statistics, correlations and p-values, histograms, bar graphs, scatterplots, and multiple linear regression statistics.

Checking and cleaning data

The program initially uploads and checks the data by printing the first few lines to ensure the data set was uploaded correctly. The data set was cleaned by removing participants with *NA* values for variables of interest (when social phobia, quality of life, general anxiety, narcissism, and hours gaming per week questions were not completed/incomplete) and two participants were removed who stated they engaged in online gaming over 168 hours per week (because this is not possible).

Three new pandas dataframes were created to simplify my code in two separate functions (*quantitative_cleaning* and *qualitative_cleaning*). A dataframe was created for qualitative variables (e.g. type of game, gender, education, gaming platform etc…) for statistical analysis, another dataframe was created for graph data for qualitative variables. Another dataframe was created for quantitative variables (e.g. age, number of hours gaming and streaming per week, anxiety questionnaire scores, etc...).

The code was written so all participants who gamed for over 168 hours per week and all *NA* values (for variables of interest) were removed instead of removing specific participants based on specific hours of gaming/ID numbers. This was done to simplify my code and make it more efficient by making it more usable for different datasets. The code would remove all participants with over 168 hours gaming per week and *NA* values (for variables of interest) regardless of the data set used. Initially my code had “index_hours = data[data['Hours'] == 8000].index data.drop(index_hours, inplace=True)” to remove one participant with over 168 hours gaming per week, a problem with this code was that it would only remove participant(s) with 8000 hours gaming per week, so I changed my code so I would remove all participants with over 168 hours of gaming “data = data[~(data['Hours'] > 168)]”.

Calculating summary statistics, correlations, and graphs

A for loop and accumulator pattern were used to print definitions of variables (from a list) and generate the summary statistics, correlations, and graphs. This was done to avoid unnecessary repetitive code. When the code was initially written, each independent variable had its own block of code to produce each summary statistic, correlation, and graph. For example, with the initial qualitative summary statistics code, as shown in *Figure 1* (before definitions were added).

```
def qualitative_summary(data):
    '''Frequency of all qualitative variables (excluding highest league,reference,accept)'''
    '''GADE qualitative statistics (how difficult has gaming made their 'work, take care of things at home, or get along with other people').'''
    gade_count = data.groupby('GADE')['S. No.'].count()
    print(gade_count)
    print()

    game_count = data.groupby('Game')['S. No.'].count()
    print(game_count)
    print()

    platform_count = data.groupby('Platform')['S. No.'].count()
    print(platform_count)
    print()

    earnings_count = data.groupby('earnings')['S. No.'].count()
    print(earnings_count)
    print()

    whyplay_count = data.groupby('whyplay')['S. No.'].count()
    print(whyplay_count)
    print()

    league_count = data.groupby('League')['S. No.'].count()
    print(league_count)
    print()

    gender_count = data.groupby('Gender')['S. No.'].count()
    print(gender_count)
    print()

    work_count = data.groupby('Work')['S. No.'].count()
    print(work_count)
    print()

    degree_count = data.groupby('Degree')['S. No.'].count()
    print(degree_count)
    print()

    residence_count = data.groupby('Residence')['S. No.'].count()
    print(residence_count)
    print()

    birthplace_count = data.groupby('Birthplace')['S. No.'].count()
    print(birthplace_count)
    print()

    playstyle_count = data.groupby('Playstyle')['S. No.'].count()
    print(playstyle_count)
    print()
```

Figure 1 – Qualitative summary statistics long

To remove unnecessary repetition of code for loops, the accumulator pattern (and lists of column headings when only some variables in the dataframe were used) were used to generate summary statistics, graphs, and correlations. For example, with the updated

qualitative summary statistics code as shown in *Figure 2* (definitions included) the code is shorter and simpler than the previous qualitative summary code, and it will work with a different set of variables as well unlike my code before when for loops were not used (from *Figure 1*).

```
def qualitative_summary(qualitative_data):
    '''Frequency of all qualitative variables (excluding highest league,reference

    print('\n' * 3)
    print('Qualitative variables summary')
    print()

    descriptions_qualitative = ["S. No counts number of participants", "GADE is a
    , "Game is game the participant plays 'most regularly at the moment'", "Platf
    , "Earnings is whether the participant makes money from gaming or gaming is a
    , "League asks participants what league a player is in if the participants pl
    , "Gender summary statistics", "Work asks particpants for their current work
    , "Degree asks participants for their highest educational degree", "Residence
    , "Birthplace asks participants their country of birth", "Playstyle asks part
    i = 0

    list_qualitative_data = qualitative_data.columns.tolist()

    for qualitative_variable in list_qualitative_data:
        print(descriptions_qualitative[i])
        print(qualitative_data.groupby(qualitative_variable)['S. No.'].count())
        print()
        i += 1
```

Figure 2 – Qualitative summary statistics short

Multiple linear regression

The *multi_linear* function called a separate function *multi_linear_production* to generate multiple linear regressions. This avoided having to unnecessarily repeat the for loop that used the linear_model method (from sklearn) to generate multiple linear regressions. As shown in *Figure 3*.

```python
def multi_linear(quantitative_data):
    '''calculating multiple linear regression'''
    print('\n' * 3)
    print('Multiple linear regression outputs:')
    print()

    quantitative_data = quantitative_data[['Age', 'Hours', 'streams', 'Narcissism', 'GAD_T', 'SWL_T', 'SPIN_T']].dropna()
    x = quantitative_data[['Age', 'Hours', 'streams', 'Narcissism', 'GAD_T', 'SWL_T']]
    y = quantitative_data['SPIN_T']
    list_y = quantitative_data[['SPIN_T']].columns.tolist()
    multi_linear_production(x,y,list_y)

    x = quantitative_data[['Age', 'Hours', 'streams', 'Narcissism', 'GAD_T','SPIN_T']]
    y = quantitative_data['SWL_T']
    list_y = quantitative_data[['SWL_T']].columns.tolist()
    multi_linear_production(x,y,list_y)

    x = quantitative_data[['Age', 'Hours', 'streams', 'Narcissism', 'SWL_T','SPIN_T']]
    y = quantitative_data['GAD_T']
    list_y = quantitative_data[['GAD_T']].columns.tolist()
    multi_linear_production(x,y,list_y)

    x = quantitative_data[['Age', 'Hours', 'streams', 'GAD_T', 'SWL_T','SPIN_T']]
    y = quantitative_data['Narcissism']
    list_y = quantitative_data[['Narcissism']].columns.tolist()
    multi_linear_production(x,y,list_y)
```

```python
def multi_linear_production(x,y,list_y):
    '''using sk_learn linear model for multiple linear regression and print output'''
    list_x = x.columns.tolist()
    i = 0
    multi_linear = linear_model.LinearRegression()
    multi_linear.fit(x,y)
    for variable in list_x:
        print(f"As {variable} increases by one, {list_y[0]} changes by {multi_linear.coef_[i]:.3f} (3 d.p).")
        i += 1
    print('\n' * 3)
```

Figure 3 – Multiple linear regression code

Main function

A main function was used to call other functions. This was to make the code more readable and easier to identify bugs when they occurred.

Discussion and Steps for Future Improvement

Data cleaning

Further data cleaning could have been done to the qualitative variables. This is because some of the questions within the data set asked participants open ended questions where participants could write their own responses. As a result, these open-ended written responses were difficult to analyze because it was difficult to do meaningful statistical analysis on these

unique written answers. For example, in the question “Is playing this game your hobby or do you make money from it?”, participants could choose from three forced choice answers or write their own response (Sauter and Draschkow, 2017). If all the written response answers (responses not from forced choice options) were counted together and named as a new variable like ‘other’ this would have made statistical analysis with qualitative variables significantly easier.

Graphing

Some of the graphs are difficult to read, this could be improved upon. For example, with the histograms relating to country of residence and birthplace, they had many independent variables (countries) making each country difficult to read as shown in *Figure 4*. In the future I could create a new dataframe during the data cleaning phase with modified country of residence and birthplace dataframe columns. These new columns could combine all the less common countries of residence and birthplaces into an ‘other’ country variable and keep the most common countries in their own separate columns. This would have made it easier to interpret the country of residence and birthplaces graphs.

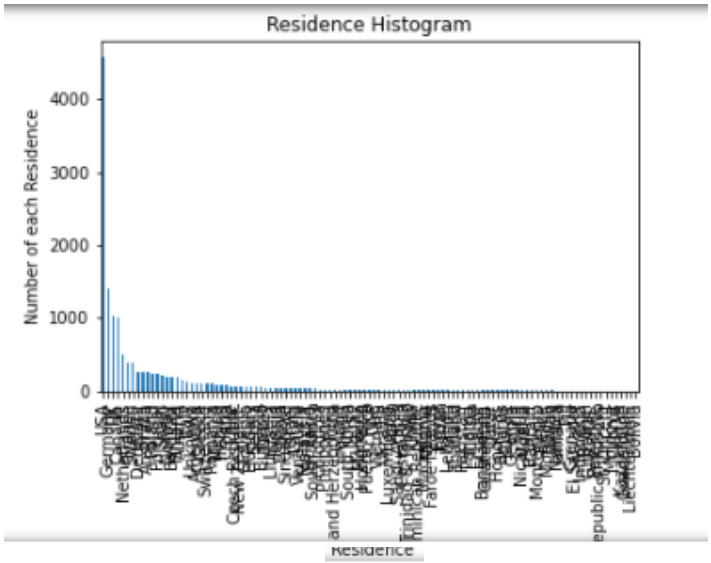

Figure 4 – Residence Histogram

All the scatterplots are one colour, in the future it may be useful to have different colours to represent some categorical variables like gender using matplotlib for example. This would be beneficial because it would not only show the relationship between two quantitative variables (e.g. hours of gaming per week and social phobia levels), it would also show more useful information about a third categorical variable such as gender which could be differentiated using different colours.

Some of the outliers could be removed to produce graphs that are easier to interpret. This is because for some of the graphs the distribution of the independent variable is difficult to see due to outliers. For example, in the age summary statistic histogram a participant at the age of 63 right skewed the age histogram making it difficult to see the range of ages between 18 to 22, as shown in *Figure 5*. In the future the histograms with a clear right skew such as age, hours gaming and streams per week could have participants' whose independent variable is considered an outlier removed and shown in a separate graph to show a clearer distribution of the independent variable as well.

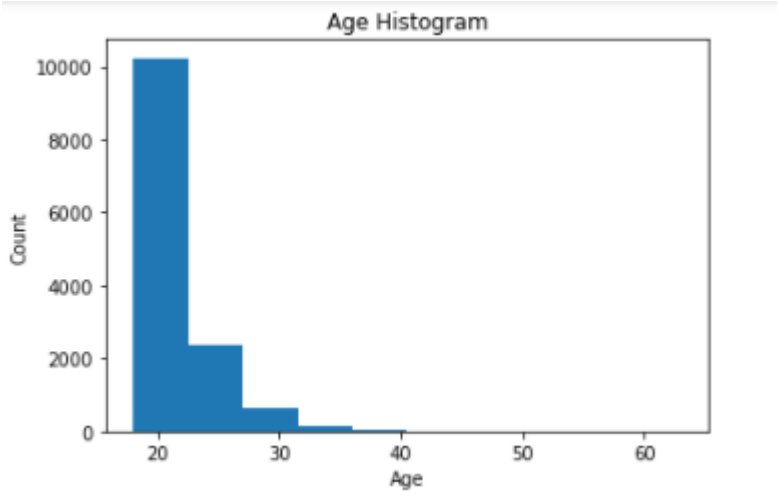

Figure 5 – Age Histogram

The output for the program code could be easier for the user to read. In the future, the program could produce the output on an interactive dashboard instead of having all the output produced at once in one large output at the end of the program. This would make it easier for a user running the program to be able to find the summary statistics, correlations statistics, graphs and multiple linear regression statistics more easily. A Python framework such as Dash could be used to create a dashboard that is more user friendly than the program's current output (Castillo, 2021).

Conclusion

This COSC480 project can generate summary statistics, correlations, graphs, and multiple linear regression statistics about online gaming, social phobia, general anxiety, quality of life and narcissism. For loops were frequently used to produce the statistics and graphs to make the code more readable and shorter. However, the output of statistics, correlations and graphs could be generated in a more user-friendly format to make it easier to read, and the graphs could also be modified to be easier to interpret. The COSC480 project has been a good learning experience for myself using Python to analyze a real-world data set.

References

Castillo, D. (2021). *Develop Data Visualization Interfaces in Python With Dash – Real Python*. RealPython.com. Retrieved 3 June 2021, from https://realPython.com/Python-dash/#what-is-dash.

Sauter, M., & Draschkow, D. (2017). Are gamers sad and isolated? A database about the anxiety, life satisfaction and social phobia of over 13000 participants.