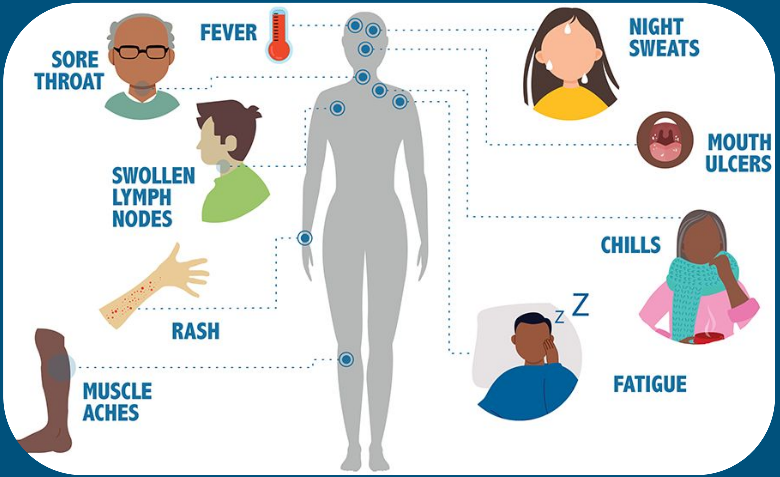
Interactive Modeling of Human Immunodeficiency Virus in the United States

Tyler Audino, Tikiri Ekanayake, Arlen Gyden, and Quinn Yuan

What is HIV?

- Human Immunodeficiency Virus
  - The last stage of HIV is called Acquired Immune Deficiency Syndrome (AIDS)
    - Severely damages the immune system
- HIV can be transmitted in various ways, most of them involving the exchange of body fluids from an infected person into the bloodstream of an uninfected person.



https://wwwnc.cdc.gov/travel/diseases/hiv

Purpose

- This tool aims to predict at-risk communities in need of further health & wellness education and resources—specifically in regards to HIV—using public health databases
- Provides visual aids in the form of maps and plots, as well as statistical aids through ANOVA tests and T-Tests.
- Data from AIDSVu - “State New Diagnoses” 2008-2020
  - Datasets: https://aidsvu.org/resources/#/datasets
  - Data methods: https://aidsvu.org/data-methods/data-methods-statecounty/

Program Outline

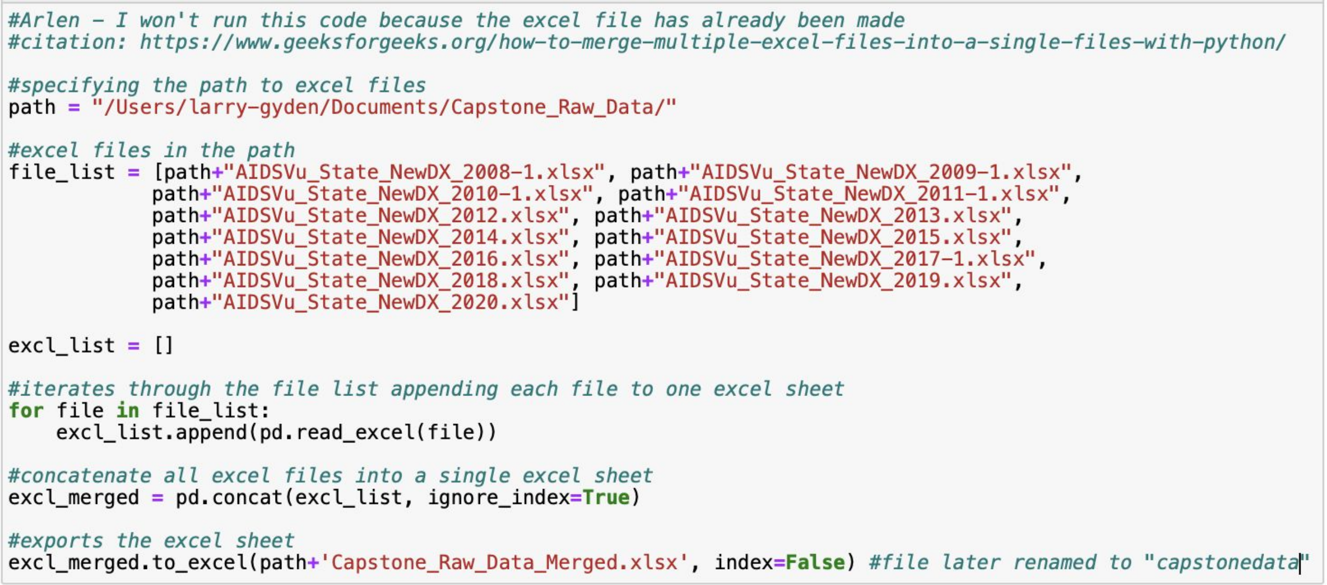
- Data Concatenating
  - Used to merge multiple excel files (multiple years of data) into one sheet
- Interactive Choropleth Map
  - Takes up to 2 user parameters and generates a choropleth map of that specific data
    - Sex, Gender, Age, or Transmission Type
  - Useful for data visualization across states
- ANOVA
  - Statistical test used to determine if multiple independent variables have a statistically significant impact on data
- T-Test
  - Statistical test used to determine if there is significant difference between two groups

Data Limitations and Considerations

- Data is collected by individual states, not nationally
    - Inconsistencies of available data between states
    - Cases are not per capita
        - More populous states will generally have higher cases
- ANOVA cannot compare individual parameters
    - Data was not available
    - Ex: ages 13-24 and 25-34 | Female gender and Asian race
- Originally planned to do data by counties
    - Data by counties was largely lacking in many states
    - Decided to switch to state data
        - More widely available and mostly complete
- dataskip() function
    - Created to clean data when data is missing for certain parameters
    - Some states were missing data for more specific parameters

Data Concatenating

- Combines the separate excel files from the database into one file that can be converted into pandas

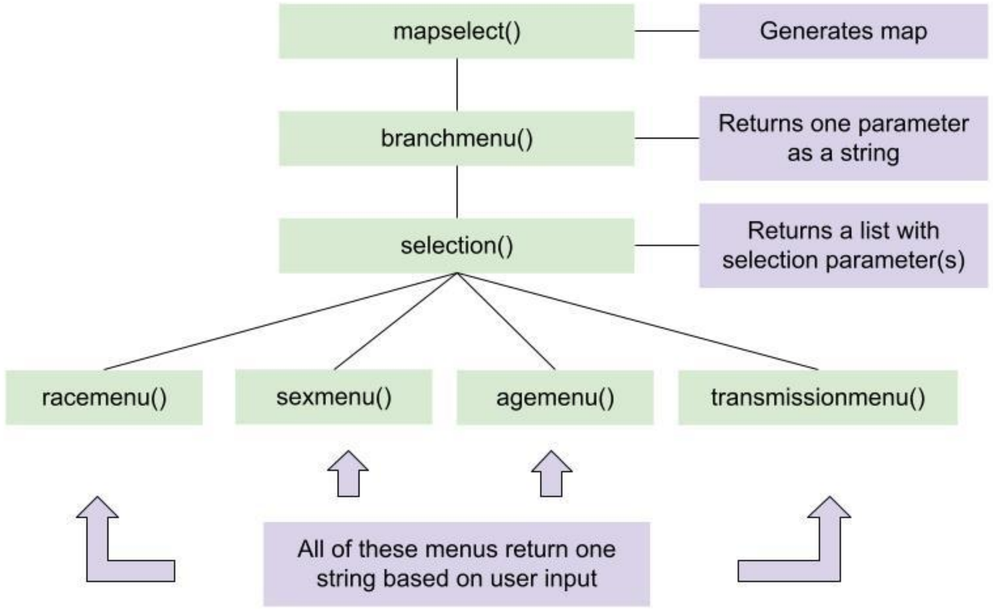


```
#Arlen - I won't run this code because the excel file has already been made
#citation: https://www.geeksforgeeks.org/how-to-merge-multiple-excel-files-into-a-single-files-with-python/

#specifying the path to excel files
path = "/Users/larry-gyden/Documents/Capstone_Raw_Data/"

#excel files in the path
file_list = [path+"AIDSVu_State_NewDX_2008-1.xlsx", path+"AIDSVu_State_NewDX_2009-1.xlsx",
             path+"AIDSVu_State_NewDX_2010-1.xlsx", path+"AIDSVu_State_NewDX_2011-1.xlsx",
             path+"AIDSVu_State_NewDX_2012.xlsx", path+"AIDSVu_State_NewDX_2013.xlsx",
             path+"AIDSVu_State_NewDX_2014.xlsx", path+"AIDSVu_State_NewDX_2015.xlsx",
             path+"AIDSVu_State_NewDX_2016.xlsx", path+"AIDSVu_State_NewDX_2017-1.xlsx",
             path+"AIDSVu_State_NewDX_2018.xlsx", path+"AIDSVu_State_NewDX_2019.xlsx",
             path+"AIDSVu_State_NewDX_2020.xlsx"]

excl_list = []

#iterates through the file list appending each file to one excel sheet
for file in file_list:
    excl_list.append(pd.read_excel(file))

#concatenate all excel files into a single excel sheet
excl_merged = pd.concat(excl_list, ignore_index=True)

#exports the excel sheet
excl_merged.to_excel(path+'Capstone_Raw_Data_Merged.xlsx', index=False) #file later renamed to "capstonedata"
```

Choropleth Map

- mapselect() - Main Function
  - Asks user what parameters they would like, by calling branchmenu() function
    - branchmenu() calls selection(), which then calls individual menu functions (racemenu(), agemenu(), sexmenu(), and transmissionmenu()) based on what parameter(s) the user wants
      - selection() returns a list of parameter names as strings
    - branchmenu() asks if user wants to look at cases or rates
      - Uses a list of all header names, and narrows down to header that contains the key-words provided by the user parameters
    - branchmenu() returns a single string of the name of the column header for the data the user wants to look at
  - Mapselect cleans data column for the selected data, then generates the map using the maps class

Functions Overview



mapselect()

Generates map

branchmenu()

Returns one parameter as a string

selection()

Returns a list with selection parameter(s)

racemenu()

sexmenu()

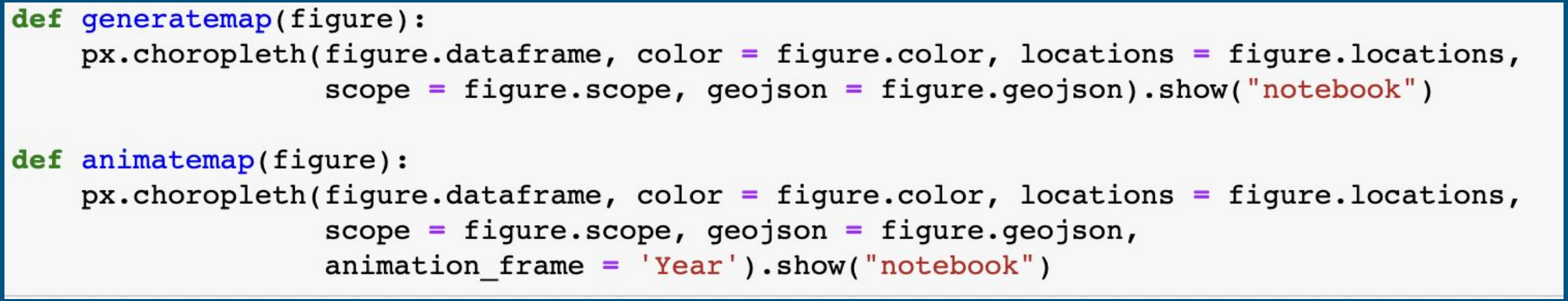
agemenu()

transmissionmenu()

All of these menus return one string based on user input

Choropleth Map

- Uses plotly.express package
- Requires a GeoJSON or JSON file of the area to be mapped
  - JSON File Used: https://www.kaggle.com/datasets/pompelmo/usa-states-geojson?resource=download
- px.choropleth(dataframe, color, locations, scope, geojson) generates the map

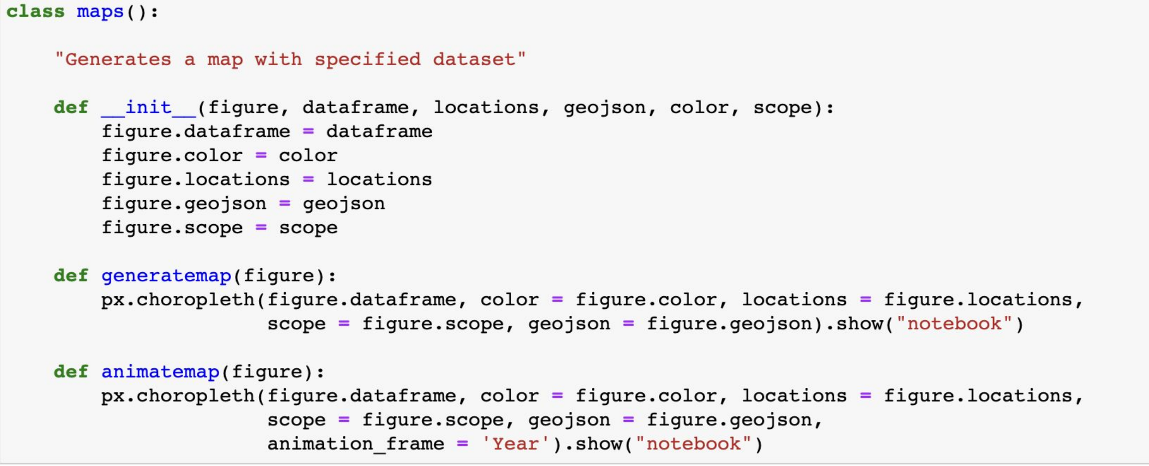


```
def generatemap(figure):
    px.choropleth(figure.dataframe, color = figure.color, locations = figure.locations,
                  scope = figure.scope, geojson = figure.geojson).show("notebook")

def animatemap(figure):
    px.choropleth(figure.dataframe, color = figure.color, locations = figure.locations,
                  scope = figure.scope, geojson = figure.geojson,
                  animation_frame = 'Year').show("notebook")
```

Choropleth Map

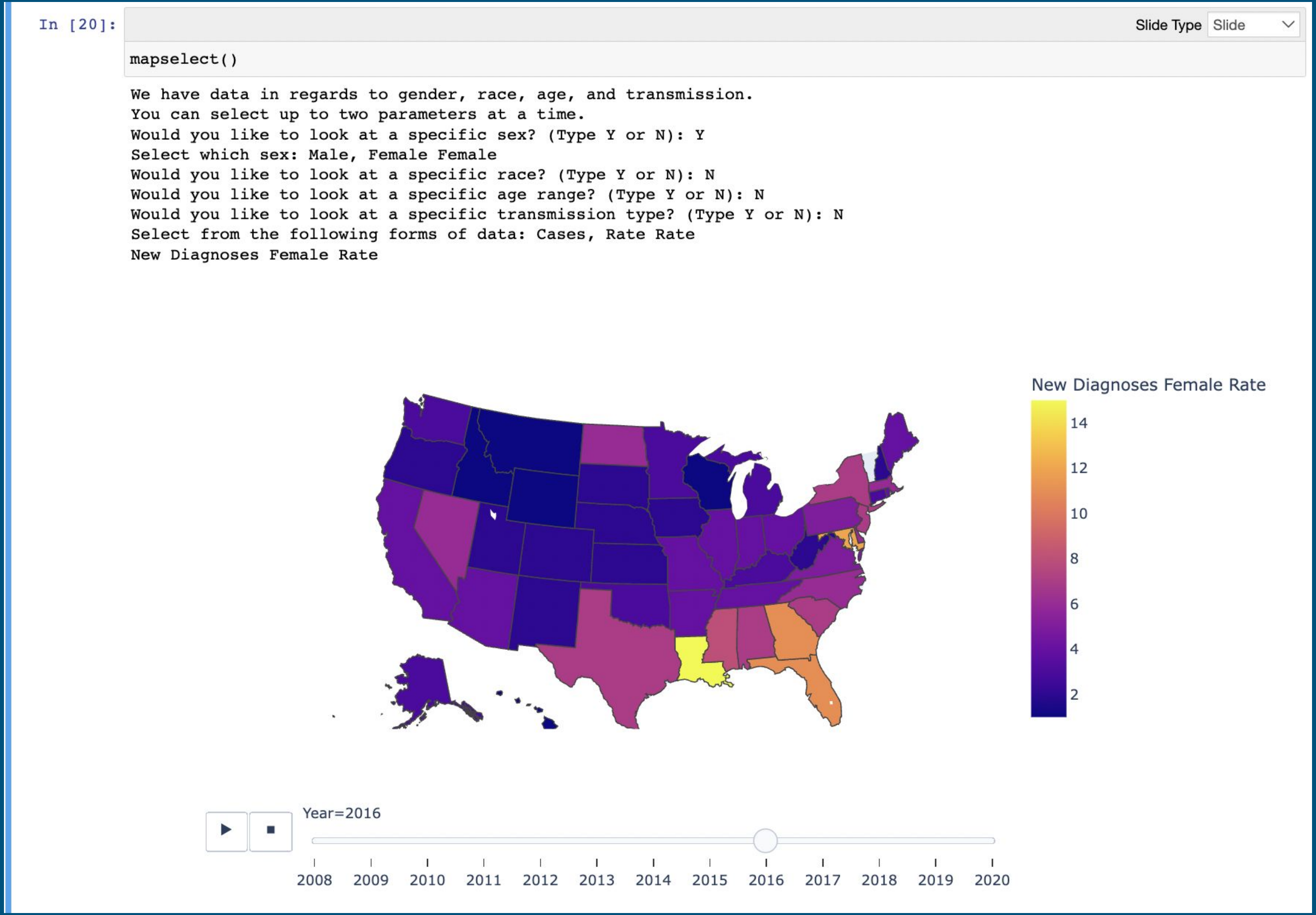
Map Class:

- Develops the individual maps with user-provided parameters
- figure.color must be a string of the name of the column of data to be displayed



```
class maps():

    "Generates a map with specified dataset"

    def __init__(figure, dataframe, locations, geojson, color, scope):
        figure.dataframe = dataframe
        figure.color = color
        figure.locations = locations
        figure.geojson = geojson
        figure.scope = scope

    def generatemap(figure):
        px.choropleth(figure.dataframe, color = figure.color, locations = figure.locations,
                      scope = figure.scope, geojson = figure.geojson).show("notebook")

    def animatemap(figure):
        px.choropleth(figure.dataframe, color = figure.color, locations = figure.locations,
                      scope = figure.scope, geojson = figure.geojson,
                      animation_frame = 'Year').show("notebook")
```

Example Output

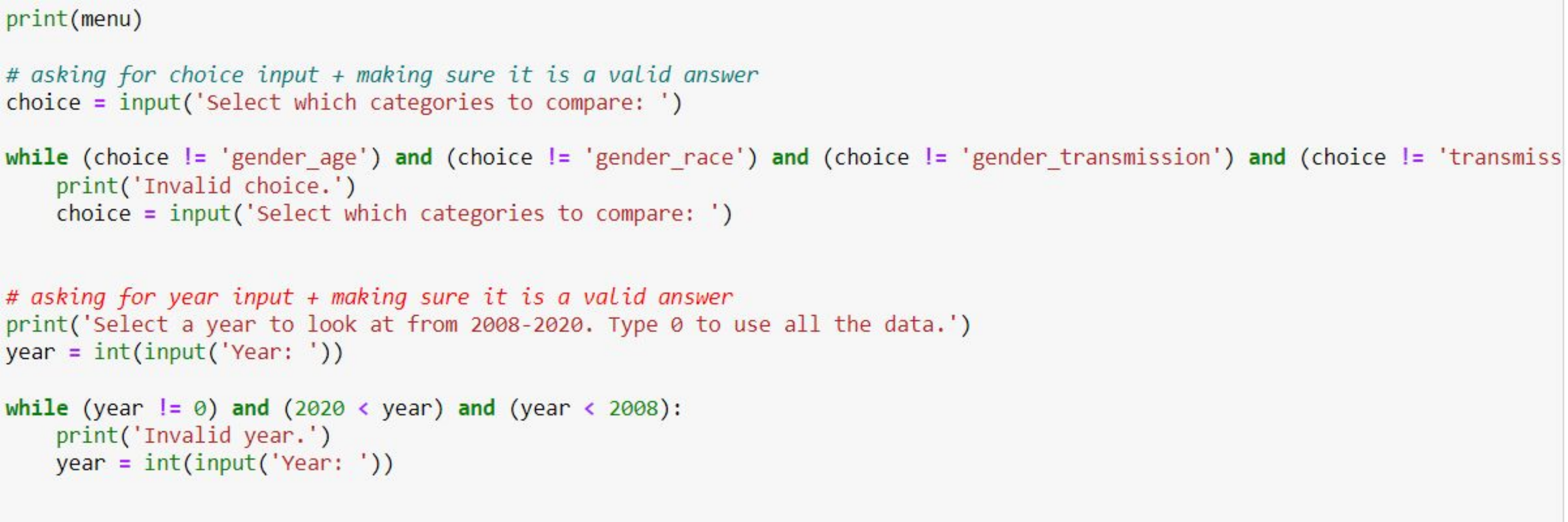


```
mapselect()
```

```
We have data in regards to gender, race, age, and transmission.
You can select up to two parameters at a time.
Would you like to look at a specific sex? (Type Y or N): Y
Select which sex: Male, Female Female
Would you like to look at a specific race? (Type Y or N): N
Would you like to look at a specific age range? (Type Y or N): N
Would you like to look at a specific transmission type? (Type Y or N): N
Select from the following forms of data: Cases, Rate Rate
New Diagnoses Female Rate
```

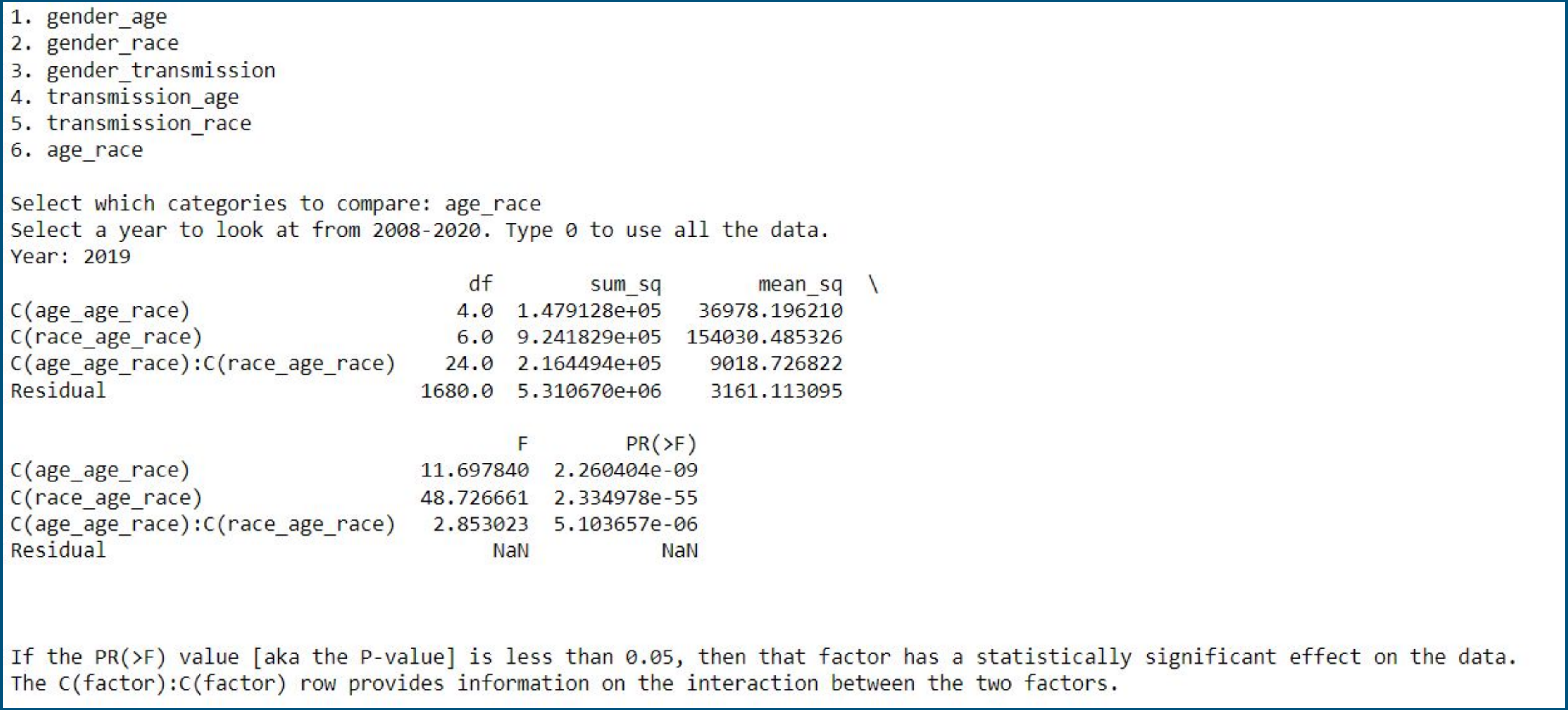
ANOVA

- Testing if there is statistical significance in the interaction between two different variables



```
print(menu)

# asking for choice input + making sure it is a valid answer
choice = input('Select which categories to compare: ')

while (choice != 'gender_age') and (choice != 'gender_race') and (choice != 'gender_transmission') and (choice != 'transmiss
    print('Invalid choice.')
    choice = input('Select which categories to compare: ')


# asking for year input + making sure it is a valid answer
print('Select a year to look at from 2008-2020. Type 0 to use all the data.')
year = int(input('Year: '))

while (year != 0) and (2020 < year) and (year < 2008):
    print('Invalid year.')
    year = int(input('Year: '))
```

Example Output

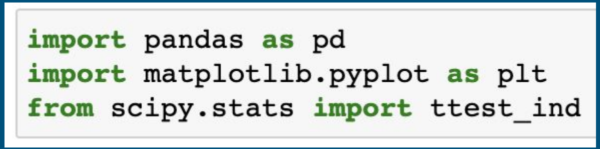


```
1. gender_age
2. gender_race
3. gender_transmission
4. transmission_age
5. transmission_race
6. age_race

Select which categories to compare: age_race
Select a year to look at from 2008-2020. Type 0 to use all the data.
Year: 2019
                                      df        sum_sq        mean_sq  \
C(age_age_race)                      4.0  1.479128e+05   36978.196210
C(race_age_race)                     6.0  9.241829e+05  154030.485326
C(age_age_race):C(race_age_race)    24.0  2.164494e+05    9018.726822
Residual                          1680.0  5.310670e+06    3161.113095

                                         F        PR(>F)
C(age_age_race)                  11.697840  2.260404e-09
C(race_age_race)                 48.726661  2.334978e-55
C(age_age_race):C(race_age_race)  2.853023  5.103657e-06
Residual                               NaN           NaN



If the PR(>F) value [aka the P-value] is less than 0.05, then that factor has a statistically significant effect on the data.
The C(factor):C(factor) row provides information on the interaction between the two factors.
```

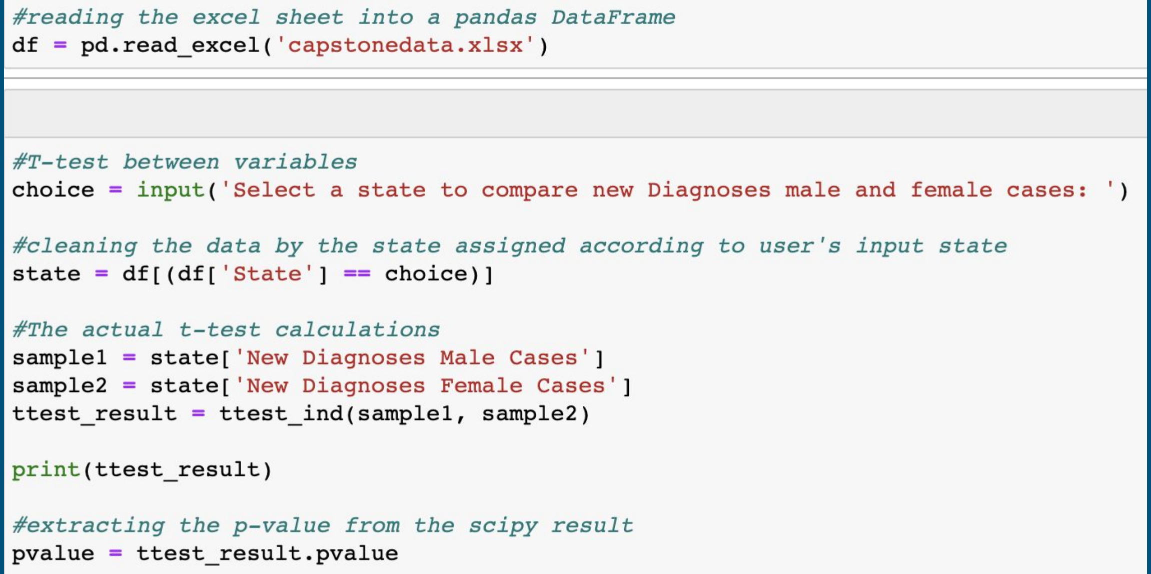
T-Test

- T-test is a parametric statistical test often used in hypothesis testing to compare the difference between means of two groups

Necessary packages to generate t-test in python



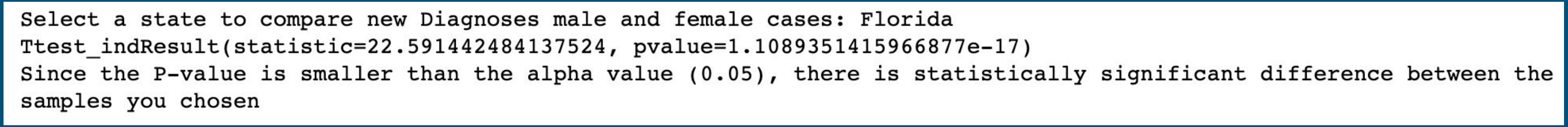
```python
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import ttest_ind
```



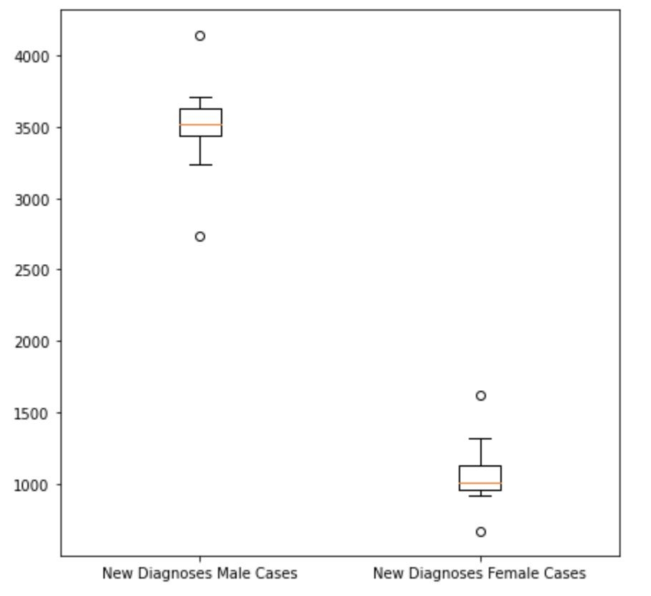
```python
#reading the excel sheet into a pandas DataFrame
df = pd.read_excel('capstonedata.xlsx')
```

```python
#T-test between variables
choice = input('Select a state to compare new Diagnoses male and female cases: ')

#cleaning the data by the state assigned according to user's input state
state = df[(df['State'] == choice)]

#The actual t-test calculations
sample1 = state['New Diagnoses Male Cases']
sample2 = state['New Diagnoses Female Cases']
ttest_result = ttest_ind(sample1, sample2)

print(ttest_result)

#extracting the p-value from the scipy result
pvalue = ttest_result.pvalue
```

Example Output



```
Select a state to compare new Diagnoses male and female cases: Florida
Ttest_indResult(statistic=22.591442484137524, pvalue=1.1089351415966877e-17)
Since the P-value is smaller than the alpha value (0.05), there is statistically significant difference between the
samples you chosen
```



THANK YOU!