Video Game Demo

We will be examining data from a Tidy Tuesday repository. The repository can be accessed from the following link: https://github.com/rfordatascience/tidytuesday/tree/master/data/2021/2021-03-16. This dataset contains a time series of video games and their average number of players, peak players and gain in players over time.

With this data I would like to be able to predict the number of players for any game and any month of a year.

```
library(reticulate)
use_virtualenv("cse6040")
```

Let's first begin by reading in our data with the code below. The data can be read directly from the rfordatascience Github repo or downloaded then read in.

```
import pandas as pd
video_game_tbl = pd.read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/maste
```

Now that we have our data, let's examine different characteristics of the data. Below are three different ways to view the data.

Method 1 uses the head() method. This will display the first few rows of data. This is almost identical to the R head function.

```
print(video_game_tbl.head())
```

```
                        gamename  year  ...     peak  avg_peak_perc
0  Counter-Strike: Global Offensive  2021  ...  1123485       65.9567%
1                          Dota 2  2021  ...   651615       62.1275%
2     PLAYERUNKNOWN'S BATTLEGROUNDS  2021  ...   447390       44.4707%
3                     Apex Legends  2021  ...   196799       61.4752%
4                             Rust  2021  ...   224276       52.4988%

[5 rows x 7 columns]
```

Next is the iloc() method from pandas. It behaves similar to the head method but slices based on index of either row or column. We will examine the first 5 rows with the code below.

```
print(video_game_tbl.iloc[:5,])
```

```
                        gamename  year  ...     peak  avg_peak_perc
0  Counter-Strike: Global Offensive  2021  ...  1123485       65.9567%
1                          Dota 2  2021  ...   651615       62.1275%
2     PLAYERUNKNOWN'S BATTLEGROUNDS  2021  ...   447390       44.4707%
```

```
3                    Apex Legends  2021  ...   196799       61.4752%
4                            Rust  2021  ...   224276       52.4988%

[5 rows x 7 columns]
```

Finally, if you are an R user, you will appreciate the info() method. This method is very similar to the glimpse() function found within R's tidyverse suite of packages. The info method will come in handy to quickly see columns and types.

```
video_game_tbl.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 83631 entries, 0 to 83630
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   gamename       83631 non-null  object
 1   year           83631 non-null  int64
 2   month          83631 non-null  object
 3   avg            83631 non-null  float64
 4   gain           82373 non-null  float64
 5   peak           83631 non-null  int64
 6   avg_peak_perc  83631 non-null  object
dtypes: float64(2), int64(2), object(3)
memory usage: 4.5+ MB
```

So we can see that we have game information by year and month. Below is a more detailed breakdown of each field from the github repository.

variable	class	description
gamename	character	Name of video games
year	double	Year of measure
month	character	Month of measure
avg	double	Avg. # of players at the same time
gain	double	Gain (or loss) difference in avg compared to prior month (NA = 1st month)
peak	double	Highest # of players at the same time
avg_peak_perc	character	Share of the average in the maximum value (avg/peak) in %

Now that we understand a little more about our structure, let's go ahead and examine any potential missing data within our data. Below we will use isnull() and sum() to figure this out.

```
video_game_tbl.isnull().sum()
```

```
gamename            0
year                0
month               0
avg                 0
gain             1258
peak                0
avg_peak_perc       0
dtype: int64
```

It looks like we do have some missing values, all from the gain column. If we look back at our data overview, we will see that there are going to be NA's within this column for the first month in the data because it is calculate a percentage increase/decrease.

Let's examine the data a little more to see if there are any NA values that need to be cleaned up. If I were to be doing this in my job, I would check to see if there are 1258 unique games. If there are, then that would explain all of the gain missing values, otherwise there may be duplicate names or some actual missing data! Let's implement this below.

```
video_game_tbl['gamename'].nunique()
```

```
1258
```

Voila! There are 1258 unique games in our dataset. This absolutely would explain the missing values in gain.

In all likelihood, we will not need to worry about the first month of each game. Let's go ahead and remove those missing values so they do not become pesky down the road.

```
video_game_clean_tbl = video_game_tbl.dropna()
```

Let's go ahead and confirm that we removed a total of 1258 rows and nothing more.

```
video_game_tbl.shape[0] - video_game_clean_tbl.shape[0]
```

```
1258
```

Let's take a quick look at what years are covered in our data with the code below.

```
video_game_clean_tbl['year'].unique()
```

```
array([2021, 2020, 2019, 2018, 2017, 2016, 2015, 2014, 2013, 2012],
      dtype=int64)
```

Cool, we have data going back from 2012 up through 2021. This means we have data through a portion of the pandemic. Let's move on and examine a few games but we will come back to the pandemic call-out.

First let's get the total number of players regardless of time or game.

```
video_game_clean_tbl['avg'].sum()
```

```
226176920.60000002
```

Holy moly. The total averaged value is 226 million. Could you imagine what this number looks like overall? That is a lot of screen-time!