project

April 15, 2023

1 Initial Imports

```
[16]: import pandas as pd
      import numpy as np
      import matplotlib.pyplot as plt
      import seaborn as sns

      from sklearn.manifold import TSNE
```

2 Dataset Selection

```
[21]: # This is just to get the latest dated dataset without changing the string␣
        ↪manually
      import os
      latest_data_dir: list = [file for file in os.listdir() if file.
        ↪startswith("df_steam")]

      # Import and visualize
      df_main: pd.DataFrame = pd.read_csv(latest_data_dir[0], na_values=['None'])
      df_main = df_main.dropna()

      print(df_main.shape)
      df_main.head()
```

```
(1000, 20)
```

```
[21]:      appid                             name
      0      570                           Dota 2  \
      1      730  Counter-Strike: Global Offensive
      2  1172470                      Apex Legends
      3   578080               PUBG: BATTLEGROUNDS
      4  1063730                         New World

                                developer        publisher  score_rank  positive
      0                                 Valve            Valve         NaN   1620812  \
      1  Valve, Hidden Path Entertainment            Valve         NaN   6317781
      2             Respawn Entertainment  Electronic Arts         NaN    523297
```

```
3                    KRAFTON, Inc.    KRAFTON, Inc.         NaN   1237022
4                     Amazon Games     Amazon Games         NaN    177176

   negative  userscore                      owners  average_forever
0    341671          0  200,000,000 .. 500,000,000            38131  \
1    810361          0   50,000,000 .. 100,000,000            30485
2    109790          0   50,000,000 .. 100,000,000             7186
3    927771          0   50,000,000 .. 100,000,000            22055
4     76296          0   50,000,000 .. 100,000,000             7386

   average_2weeks  median_forever  median_2weeks  price  initialprice
0            1313             879            730      0             0  \
1             759            5864            286      0             0
2             809             813            419      0             0
3             691            6340            188      0             0
4             338            3225            167   3999          3999

   discount      ccu                                          languages
0         0   520119  English, Bulgarian, Czech, Danish, Dutch, Finn…  \
1         0  1223076  English, Czech, Danish, Dutch, Finnish, French…
2         0   417985  English, French, Italian, German, Spanish - Sp…
3         0   381064  English, Korean, Simplified Chinese, French, G…
4         0    23870  English, French, Italian, German, Spanish - Sp…

                                               genre
0                       Action;Free to Play;Strategy  \
1                                Action;Free to Play
2                      Action;Adventure;Free to Play
3  Action;Adventure;Free to Play;Massively Multip…
4         Action;Adventure;Massively Multiplayer;RPG

                                                tags
0  Free to Play;MOBA;Multiplayer;Strategy;e-sport…
1  FPS;Shooter;Multiplayer;Competitive;Action;Tea…
2  Free to Play;Multiplayer;Battle Royale;Shooter…
3  Survival;Shooter;Battle Royale;Multiplayer;FPS…
4  Massively Multiplayer;Open World;MMORPG;RPG;Ad…
```

[47]:
```
df_genres = pd.read_csv("genres.csv")

print(df_genres.shape)
df_genres.head()
```

```
(18, 1)
```

[47]:
```
                genre
0              Action
1           Adventure
```

```
2  Animation & Modeling
3      Audio Production
4                Casual
```

[48]:
```
df_tags = pd.read_csv("tags.csv")

print(df_tags.shape)
df_tags.head()
```

```
(394, 1)
```

[48]:
```
          tag
0       1980s
1      1990's
2        2.5D
3          2D
4  2D Fighter
```

3 Introduction

[ ]:
```

```

[ ]:
```

```

4 Data Cleaning & Manipulation

4.1 Cleaning

4.1.1 Dropping irrelevant columns & unplayed games

We start off with some simpler procedures. First we remove the data columns that we deem irrelevant to the analysis, which are the following, with reasons:

Dropped column	Reason
`developer`	Too many unique values and contains special characters
`publisher`	Too many unique values and contains special characters
`score_rank`	All `NaN`
`userscore`	Almost all 0
`discount`	Too dependent on current time (arguably `ccu` too, but we keep it because it is more interesting)
`initialprice`	Often matches `price` and is more of a historical data

Dropped column	Reason
`languages`	Personal choice, though an alternative analysis on English / non-English games may be interesting

We also drop any unplayed or 'barely-played' games. For this analysis we set this rule to be games with 10 or less concurrent users at time of data collection.

[359]:
```
# Cleaning: Drop data that is likely not useful
drops = ["developer", "publisher", "score_rank", "userscore", "discount",␣
 ↪"initialprice", "languages"]
df_steam = df_main.drop(drops, axis=1)

# Only games with more than 10 concurrent users
df_steam = df_steam[df_steam["ccu"] > 10]

print(df_steam.shape)
df_steam.head()
```

```
(903, 13)
```

[359]:
```
     appid                              name  positive  negative  \
0      570                            Dota 2   1620812    341671
1      730  Counter-Strike: Global Offensive   6317781    810361
2  1172470                      Apex Legends    523297    109790
3   578080               PUBG: BATTLEGROUNDS   1237022    927771
4  1063730                         New World    177176     76296

                        owners  average_forever  average_2weeks  \
0  200,000,000 .. 500,000,000            38131            1313
1   50,000,000 .. 100,000,000            30485             759
2   50,000,000 .. 100,000,000             7186             809
3   50,000,000 .. 100,000,000            22055             691
4   50,000,000 .. 100,000,000             7386             338

   median_forever  median_2weeks  price      ccu  \
0             879            730      0   520119
1            5864            286      0  1223076
2             813            419      0   417985
3            6340            188      0   381064
4            3225            167   3999    23870

                                    genre
0             Action;Free to Play;Strategy  \
1                      Action;Free to Play
2            Action;Adventure;Free to Play
```

```
3  Action;Adventure;Free to Play;Massively Multip…
4         Action;Adventure;Massively Multiplayer;RPG

                                                tags
0  Free to Play;MOBA;Multiplayer;Strategy;e-sport…
1  FPS;Shooter;Multiplayer;Competitive;Action;Tea…
2  Free to Play;Multiplayer;Battle Royale;Shooter…
3  Survival;Shooter;Battle Royale;Multiplayer;FPS…
4  Massively Multiplayer;Open World;MMORPG;RPG;Ad…
```

[330]:
```
import warnings
warnings.filterwarnings("ignore", 'This pattern is interpreted as a regular␣
 ↪expression, and has match groups')

tag_counts = []
for tag in df_tags['tag'].unique():
    tag_counts.append({"tag": tag, "count": df_main["tags"].str.contains(tag).
 ↪fillna(False).to_numpy().sum()})

df_tag_counts = pd.DataFrame(tag_counts).sort_values(by = "count", ascending =␣
 ↪False)
df_tag_counts.head(10)
```

[330]:
```
              tag  count
309  Singleplayer    826
14         Action    800
220   Multiplayer    723
20      Adventure    643
72          Co-op    493
35    Atmospheric    458
242    Open World    424
304       Shooter    363
133  First-Person    362
317    Soundtrack    355
```

[352]:
```
df_tag_counts['tag'].iloc[11:21]
```

[352]:
```
273             RPG
172           Indie
328        Strategy
327      Story Rich
344    Third Person
125             FPS
241    Online Co-Op
308      Simulation
335        Survival
297         Sandbox
```

```
Name: tag, dtype: object
```

[355]:
```
chosen_tags = df_tag_counts['tag'].iloc[11:21]

df_assignments = pd.DataFrame(columns = ['appid', 'tag'])
for tag in chosen_tags:
    indexes = df_main["tags"].str.contains(tag).fillna(False).to_list()
    appids = df_main['appid'][indexes]

    df_temp = pd.DataFrame({'appid': appids, 'tag': tag})
    df_assignments = (
        df_assignments.merge(df_temp, on='appid', how = 'outer', indicator=True)
        .query("_merge != 'both'")
        .drop('_merge', axis = 1)
    )
    df_assignments['tag'] = df_assignments['tag_x'].
 ↪combine_first(df_assignments['tag_y'])
    df_assignments = df_assignments.drop(['tag_x', 'tag_y'], axis = 1)

print(df_assignments.shape)
df_assignments.head()
```

```
(515, 2)
```

[355]:
```
    appid  tag
0  247080  RPG
1  372000  RPG
2  434650  RPG
4  788100  RPG
5   34270  RPG
```

[360]:
```
(
    df_steam
    .merge(df_assignments, on='appid', how='inner')
    .drop(['genre', 'tags'], axis=1)
)
```

[360]:
```
      appid                              name  positive  negative  \
0       730  Counter-Strike: Global Offensive   6317781    810361
1       550                       Left 4 Dead 2    710112     18055
2    230410                           Warframe    460909     71760
3    105600                           Terraria   1090163     24836
4      4000                         Garry's Mod    914868     31472
..      …                                   …         …         …
457  446150                           GUNS UP!      4269      1632
458   47410                Stronghold Kingdoms      5709      3134
459  505460                            Foxhole     24687      5461
460  640590  The LEGO NINJAGO Movie Video Game      4923       815
```

```
461  236090              Dust: An Elysian Tail     17004       789

                        owners  average_forever  average_2weeks
0    50,000,000 .. 100,000,000            30485             759  \
1     20,000,000 .. 50,000,000             1994             262
2     20,000,000 .. 50,000,000            10749            1356
3     20,000,000 .. 50,000,000             6524             794
4     20,000,000 .. 50,000,000             9596             516
..                         …                …               …
457      1,000,000 .. 2,000,000             798               0
458      1,000,000 .. 2,000,000            2568               0
459      1,000,000 .. 2,000,000            4201             340
460      1,000,000 .. 2,000,000            1014              16
461      1,000,000 .. 2,000,000             523               0

     median_forever  median_2weeks  price      ccu           tag
0              5864            286      0  1223076  Online Co-Op
1               512            106    999    24587      Survival
2               404            764      0    48454  Online Co-Op
3              1871            373    999    41773       Sandbox
4              1310             96    999    27618       Sandbox
..              …              …    …      …            …
457             169              0      0       89      Strategy
458             129              0      0      554       Sandbox
459             510            623   2999     2384       Sandbox
460              82             16   1999       64  Online Co-Op
461             343              0   1499       14    Story Rich

[462 rows x 12 columns]
```

[4]:
```
# Manipulation

# df_steam["log_ccu"] = np.log(df_steam["ccu"])
df_steam["prop_review"] = df_steam["positive"] / (df_steam["positive"] +␣
 ↪df_steam["negative"])

print(df_steam.shape)
df_steam.head()
```

```
(903, 13)
```

[4]:
```
                              name                          developer
0  Counter-Strike: Global Offensive  Valve, Hidden Path Entertainment  \
1                           Dota 2                            Valve
2                     Apex Legends             Respawn Entertainment
3              PUBG: BATTLEGROUNDS                     KRAFTON, Inc.
4                    Path of Exile                Grinding Gear Games
```

```
            publisher  positive  negative                       owners  
0               Valve   6317781    810361   50,000,000 .. 100,000,000  \
1               Valve   1620812    341671  200,000,000 .. 500,000,000   
2     Electronic Arts    523297    109790   50,000,000 .. 100,000,000   
3       KRAFTON, Inc.   1237022    927771   50,000,000 .. 100,000,000   
4  Grinding Gear Games    175016     24204    20,000,000 .. 50,000,000   

   average_forever  average_2weeks  median_forever  median_2weeks  price  
0            30485             759            5864            286      0  \
1            38131            1313             879            730      0   
2             7186             809             813            419      0   
3            22055             691            6340            188      0   
4             9636            1443             331           1187      0   

       ccu  prop_review  
0  1223076     0.886315  
1   520119     0.825899  
2   417985     0.826580  
3   381064     0.571427  
4   123844     0.878506  
```

5 Descriptive Analytics

[7]:
```
df_numerical = df_steam.select_dtypes("number").drop(["ccu", "positive",␣
 ↪"negative"], axis = 1)
df_numerical["prop_review"] = df_numerical["prop_review"].fillna(0)

df_numerical.head()
```

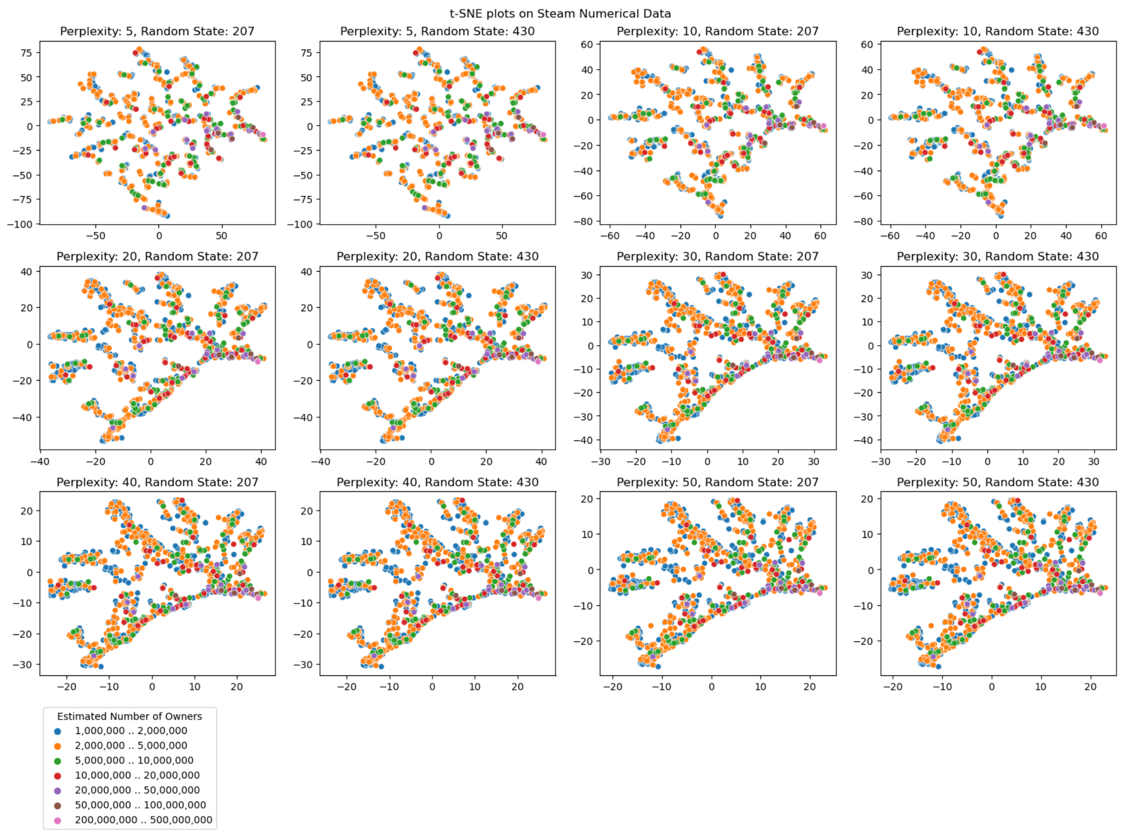
[7]:
```
   average_forever  average_2weeks  median_forever  median_2weeks  price  
0            30485             759            5864            286      0  \
1            38131            1313             879            730      0   
2             7186             809             813            419      0   
3            22055             691            6340            188      0   
4             9636            1443             331           1187      0   

   prop_review  
0     0.886315  
1     0.825899  
2     0.826580  
3     0.571427  
4     0.878506  
```

[8]:
```
# Store TSNE data to memory to save time trying plots out
# (RIP my RAM)
tsne_memo = {}
```

```
perplexities: list = [5, 10, 20, 30, 40, 50]
random_states: list = [207, 430]

for i in range(12):
    ppx = perplexities[np.floor_divide(i, 2)]
    rs = random_states[np.remainder(i, 2)]
    tsne_memo[i] = (
        TSNE(perplexity=ppx, random_state=rs)
        .fit_transform(df_numerical)
    )
```

[9]:
```
# tl;dr The order the 'owners' category should be in, in order of number␣
↪instead of string sort
owners_order = np.argsort([int(s.split(' .. ')[0].replace(',', '')) for s in␣
↪df_steam["owners"]])
```

[10]:
```
fig, axs = plt.subplots(3, 4, figsize = (16, 10))

plt.suptitle("t-SNE plots on Steam Numerical Data")

for i, ax in enumerate(axs.flatten()):
    ppx = perplexities[np.floor_divide(i, 2)]
    rs = random_states[np.remainder(i, 2)]

    sns.scatterplot(
        ax = ax,
        x = tsne_memo[i][:, 0][owners_order],
        y = tsne_memo[i][:, 1][owners_order],
        hue = df_steam["owners"].iloc[owners_order]
    )

    handles, labels = ax.get_legend_handles_labels()
    ax.legend().remove()
    ax.set_title("Perplexity: {}, Random State: {}".format(ppx, rs))

fig.legend(handles, labels, bbox_to_anchor = (0.2, 0), title = "Estimated␣
↪Number of Owners")
fig.tight_layout()
plt.show()

# del(tsne_memo)
```



t-SNE plots on Steam Numerical Data
Perplexity: 5, Random State: 207
Perplexity: 5, Random State: 430
Perplexity: 10, Random State: 207
Perplexity: 10, Random State: 430
Perplexity: 20, Random State: 207
Perplexity: 20, Random State: 430
Perplexity: 30, Random State: 207
Perplexity: 30, Random State: 430
Perplexity: 40, Random State: 207
Perplexity: 40, Random State: 430
Perplexity: 50, Random State: 207
Perplexity: 50, Random State: 430
Estimated Number of Owners
1,000,000 .. 2,000,000
2,000,000 .. 5,000,000
5,000,000 .. 10,000,000
10,000,000 .. 20,000,000
20,000,000 .. 50,000,000
50,000,000 .. 100,000,000
200,000,000 .. 500,000,000

[ ]: