Importing Relevant Library

```
import pandas as pd
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import os
```

```
from google.colab import files
uploaded=files.upload()
```

Choose Files pro.csv

- **pro.csv**(application/vnd.ms-excel) - 1278462 bytes, last modified: 11/16/2020 - 100% done

```
Saving pro.csv to pro.csv
```

```
import io
```

```
pro_df= pd.read_csv(io.BytesIO(uploaded['pro.csv']))
```

```
from google.colab import files
uploaded= files.upload()
```

Choose Files pro.csv

- **pro.csv**(application/vnd.ms-excel) - 1278462 bytes, last modified: 11/16/2020 - 100% done

```
Saving pro.csv to pro (1).csv
```

```
pr_df= pd.read_csv(io.BytesIO(uploaded['pro.csv']))
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
pro_df=pd.read_csv("pro.csv")
pro_df
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBN
0	2f262dd9795e60	78437bcd91d40e	d5db3a49eb2955	0	0	0.0	
1	a32847cf5bf34b	85b7ce5a12e10b	65223f05c7fdb4	0	0	163.2	
2	1b1900a9990396	edf80d6523380a	1cadec4534f30a	0	3	278.7	
3	f589dd03b60bf2	804ab5e5585558	c4a5676dc91604	0	0	191.9	
4	c23c4cc5b78b35	b3e2cd169ed920	cd595700a01bfa	0	0	100.0	
...	...	...	...	...	...	...	...

2) Check the datatype of all the columns.

```
pro_df.dtypes
```

```
Id                 object
groupId            object
matchId            object
assists             int64
boosts              int64
damageDealt       float64
DBNOs               int64
headshotKills       int64
heals               int64
killPlace           int64
killPoints          int64
kills               int64
killStreaks         int64
longestKill       float64
matchDuration       int64
matchType          object
maxPlace            int64
```



```
rideDistance      float64
roadKills           int64
swimDistance      float64
teamKills           int64
vehicleDestroys     int64
walkDistance      float64
weaponsAcquired     int64
winPoints           int64
winPlacePerc      float64
dtype: object
```

3) Find the Summary of all the numerical columns and write your findings about it.

```
pro_df.describe()
```

ankPoints	revives	rideDistance	roadKills	swimDistance	teamKills	vehicleDe
00.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.
89.699600	0.160200	600.693584	0.004200	4.385917	0.024400	0.
38.535034	0.454045	1524.915601	0.074719	30.889620	0.171486	0.
-1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
-1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
44.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.
00.000000	0.000000	0.000575	0.000000	0.000000	0.000000	0.
35.000000	5.000000	28780.000000	3.000000	971.200000	3.000000	2.

4) The average person kills how many players?

```
pro_df['Id'].nunique()
```

```
10000
```

```
pro_df['kills'].mean()
```

```
0.9134
```

5) 99% of people have how many kills?

```
pro_df['kills'].quantile(0.99)
```


6) The most kills ever recorded are how much?

```
pro_df['kills'].max()
```

```
35
```

7) Print all the columns of the dataframe.

```
pro_df.columns
```

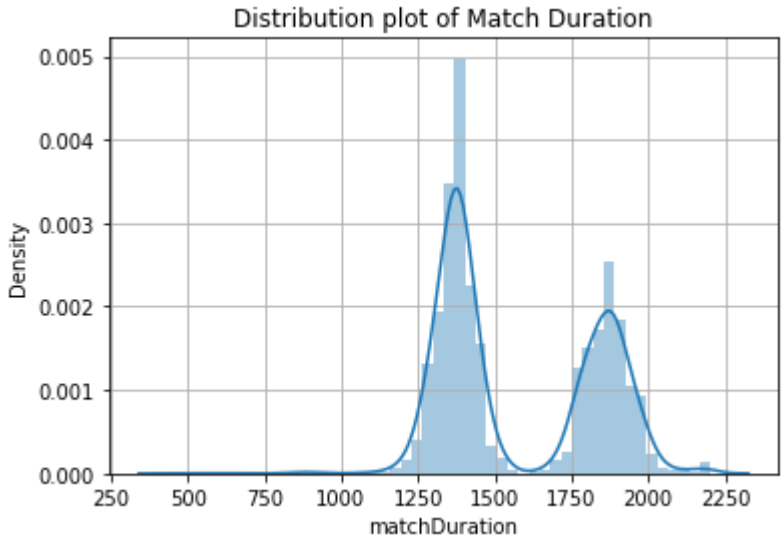
```
Index(['Id', 'groupId', 'matchId', 'assists', 'boosts', 'damageDealt', 'DBNOs',
       'headshotKills', 'heals', 'killPlace', 'killPoints', 'kills',
```

```
       'killStreaks', 'longestKill', 'matchDuration', 'matchType', 'maxPlace',
       'numGroups', 'rankPoints', 'revives', 'rideDistance', 'roadKills',
       'swimDistance', 'teamKills', 'vehicleDestroys', 'walkDistance',
       'weaponsAcquired', 'winPoints', 'winPlacePerc'],
      dtype='object')
```

8) Comment on distribution of the match's duration.Use seaborn.

```
sns.distplot(pro_df['matchDuration'], bins=50)
plt.grid()
plt.title('Distribution plot of Match Duration');
```

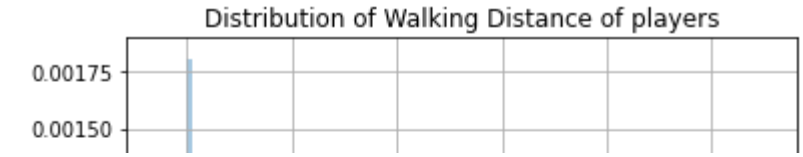
```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
```



9) Comment on distribution of the walk distance Use seaborn


```
sns.distplot(pro_df['walkDistance'], bins=100)
plt.grid()
plt.title('Distribution of Walking Distance of players');
```

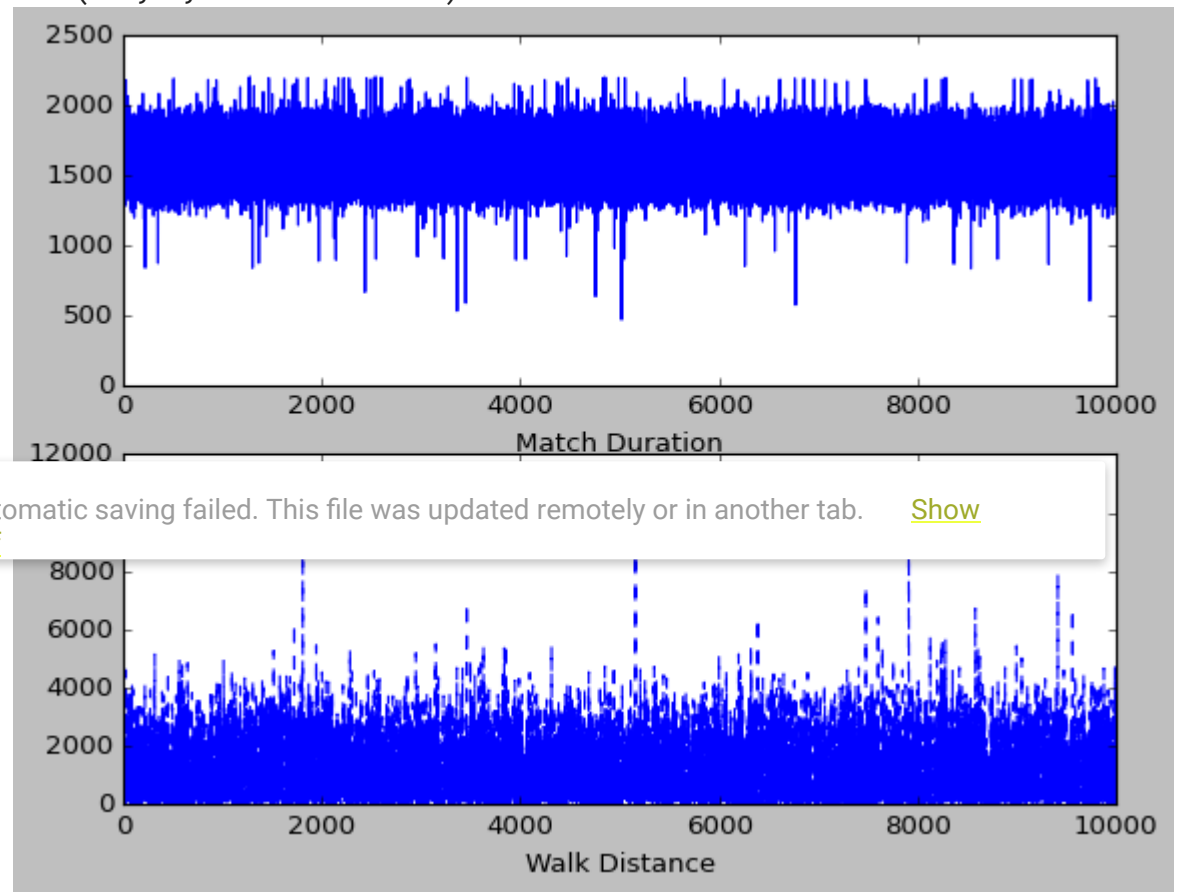
```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
```



10) Plot distribution of the match's duration vs walk distance one below the other.


```
%matplotlib inline
plt.style.use('classic')
plt.figure()
plt.subplot(2,1,1)
plt.plot(pro_df["matchDuration"],"-")
plt.xlabel("Match Duration")
plt.subplot(2,1,2)
plt.plot(pro_df["walkDistance"],"--")
plt.xlabel("Walk Distance")
```

```
Text(0.5, 0, 'Walk Distance')
```

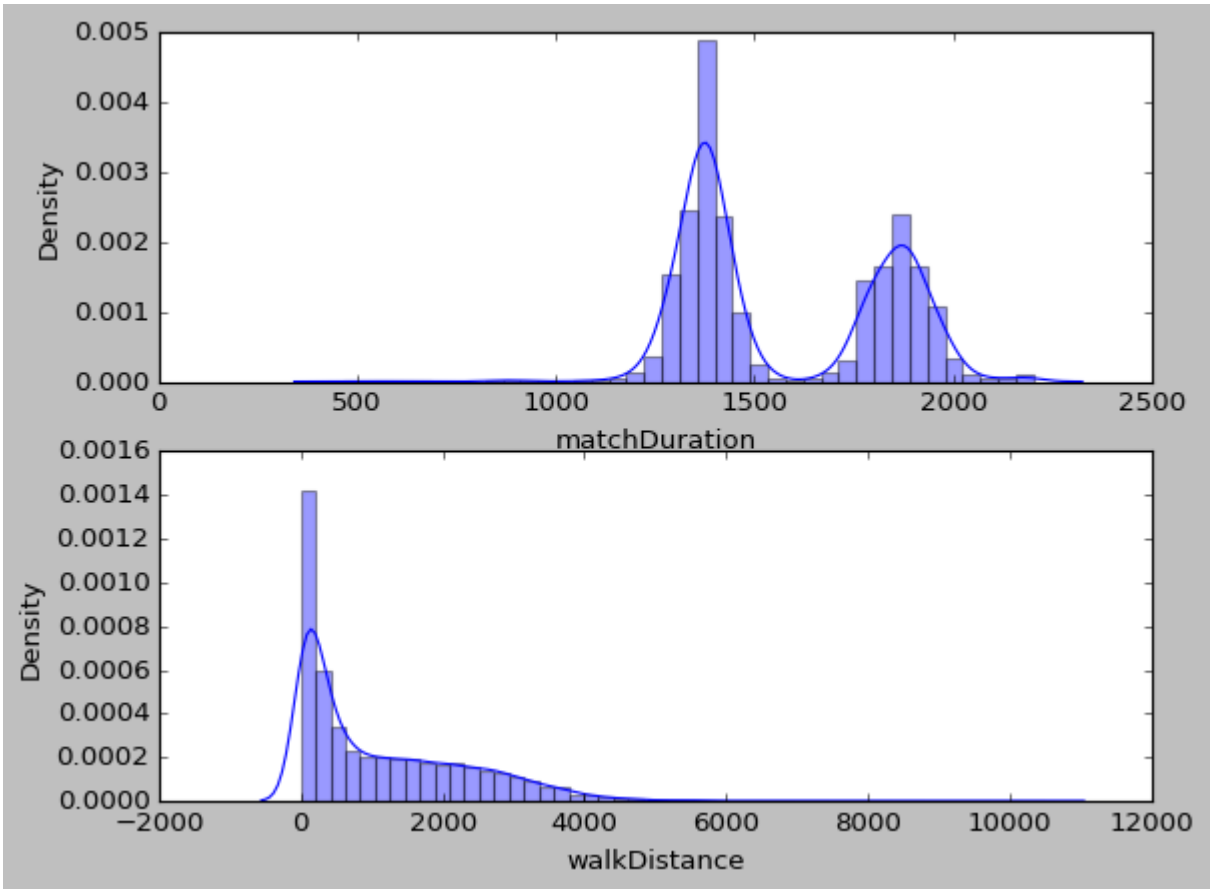


or using distplot

```
fig,ax=plt.subplots(2,1)
sns.distplot(pro_df['matchDuration'], ax=ax[0])
print()
```

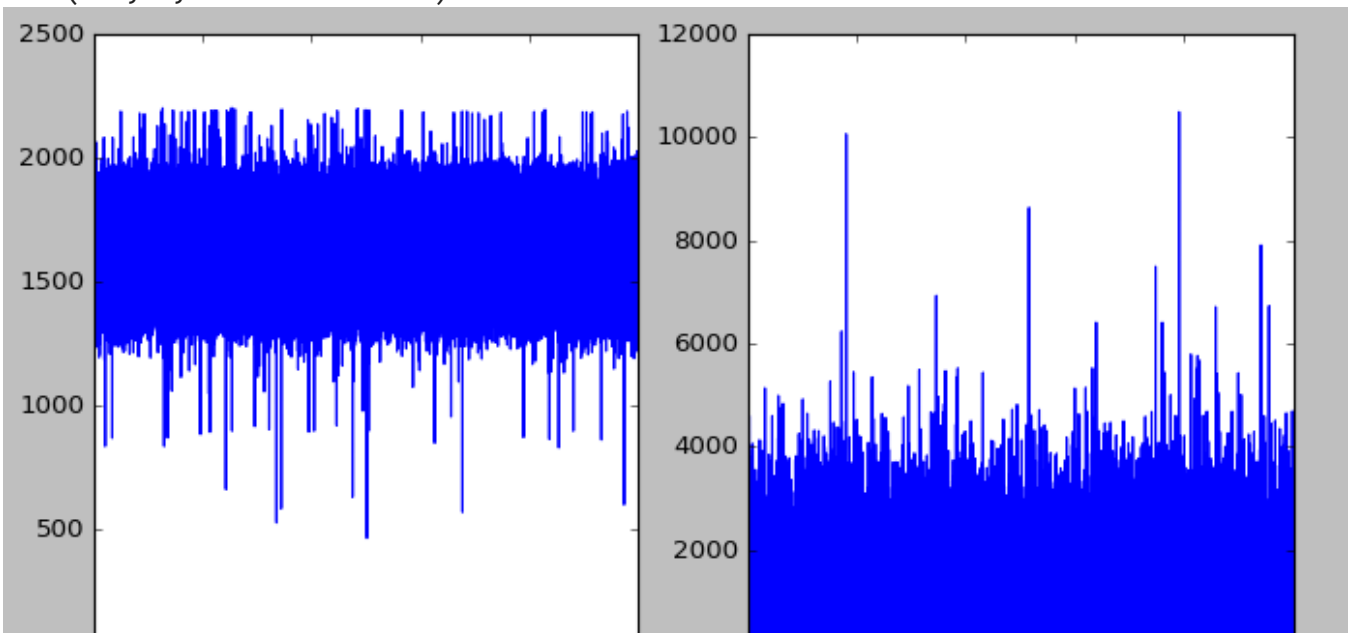
```
print()
sns.distplot(pro_df['walkDistance'],ax=ax[1])
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
```




```
%matplotlib inline
plt.style.use('classic')
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.plot(pro_df["matchDuration"])
plt.xlabel("Match Duration")
plt.subplot(1,2,2)
plt.plot(pro_df["walkDistance"])
plt.xlabel("Walk Distance")
```

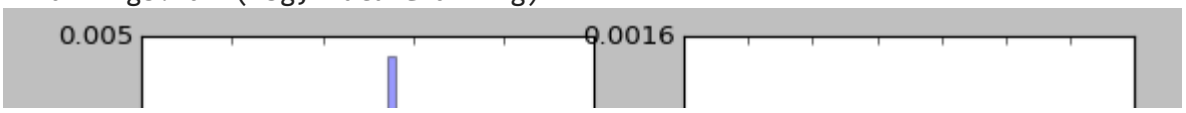
```
Text(0.5, 0, 'Walk Distance')
```



or using distplot

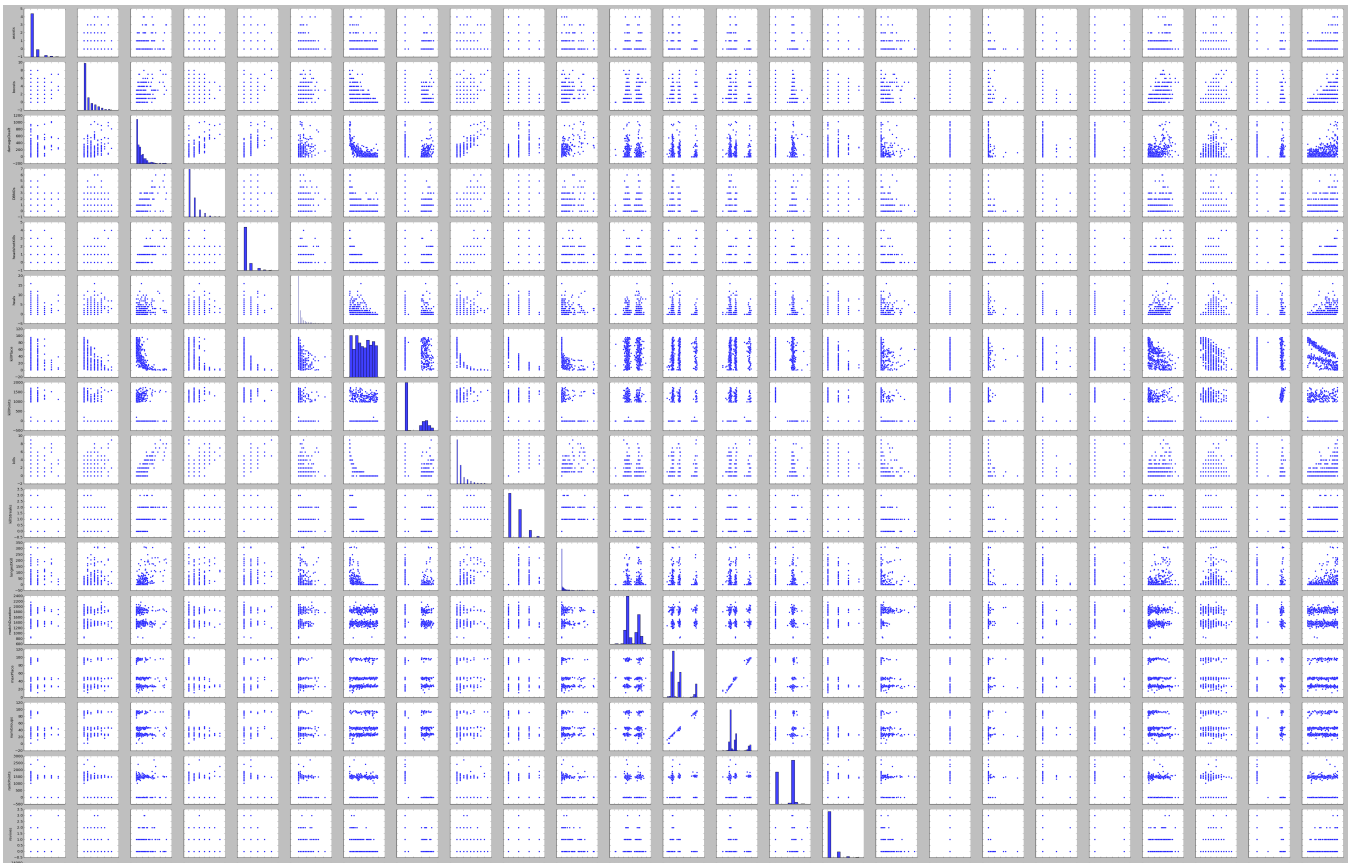
```
fig, axs=plt.subplots(1,2)
sns.distplot(pro_df.matchDuration,ax=axs[0]);
sns.distplot(pro_df.walkDistance,ax=axs[1]);
plt.xticks(rotation=60);
plt.show()
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.6/dist-packages/seaborn/distributions.py:2551: FutureWarning: `di
  warnings.warn(msg, FutureWarning)
```



12) Pairplot the dataframe. Comment on kills vs damage dealt, Comment on maxPlace vs numGroups.

```
sns.pairplot(pro_df.head(500));
```



13) How many unique values are there in 'matchType' and what are their counts?

```
uni=pd.unique(pro_df['matchType'])
print(uni)
pro_df['matchType'].value_counts()
```

```
['squad-fpp' 'squad' 'duo-fpp' 'solo-fpp' 'duo' 'solo' 'crashfpp'
 'flaretpp' 'normal-squad-fpp' 'normal-duo-fpp' 'flarefpp' 'normal-squad'
 'normal-solo-fpp' 'crashtpp']
```

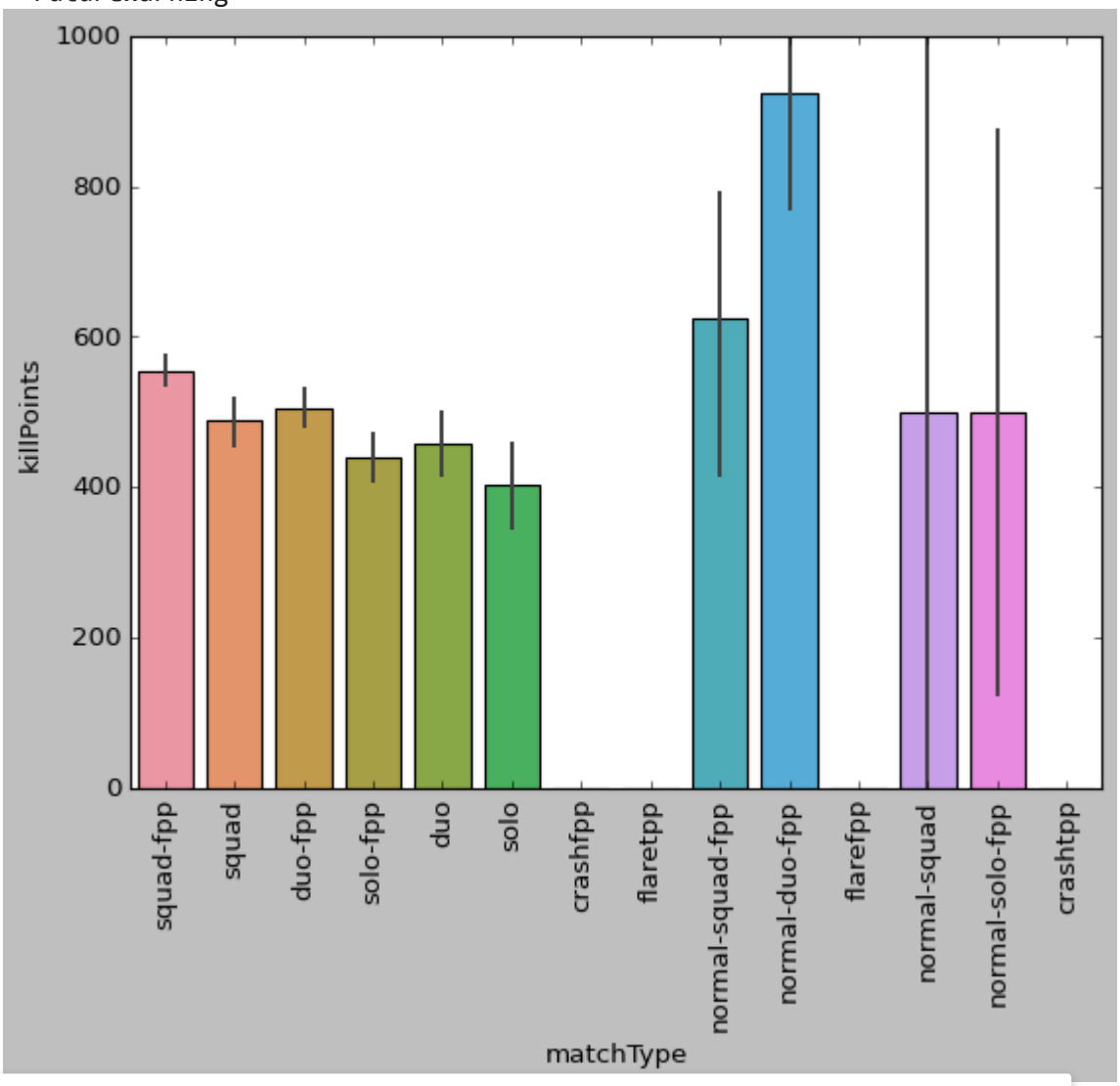


```
solo-fpp             1234
duo                   702
solo                  386
normal-squad-fpp       24
crashfpp               13
normal-duo-fpp         13
normal-solo-fpp         8
normal-squad            4
flaretpp                3
crashtpp                2
flarefpp                1
Name: matchType, dtype: int64
```

14) Plot a barplot of 'matchType' vs 'KillsPoints. Write your inferences.

```
sns.barplot(pro_df['matchType'],pro_df['killPoints']);
plt.xticks(rotation=90);
```

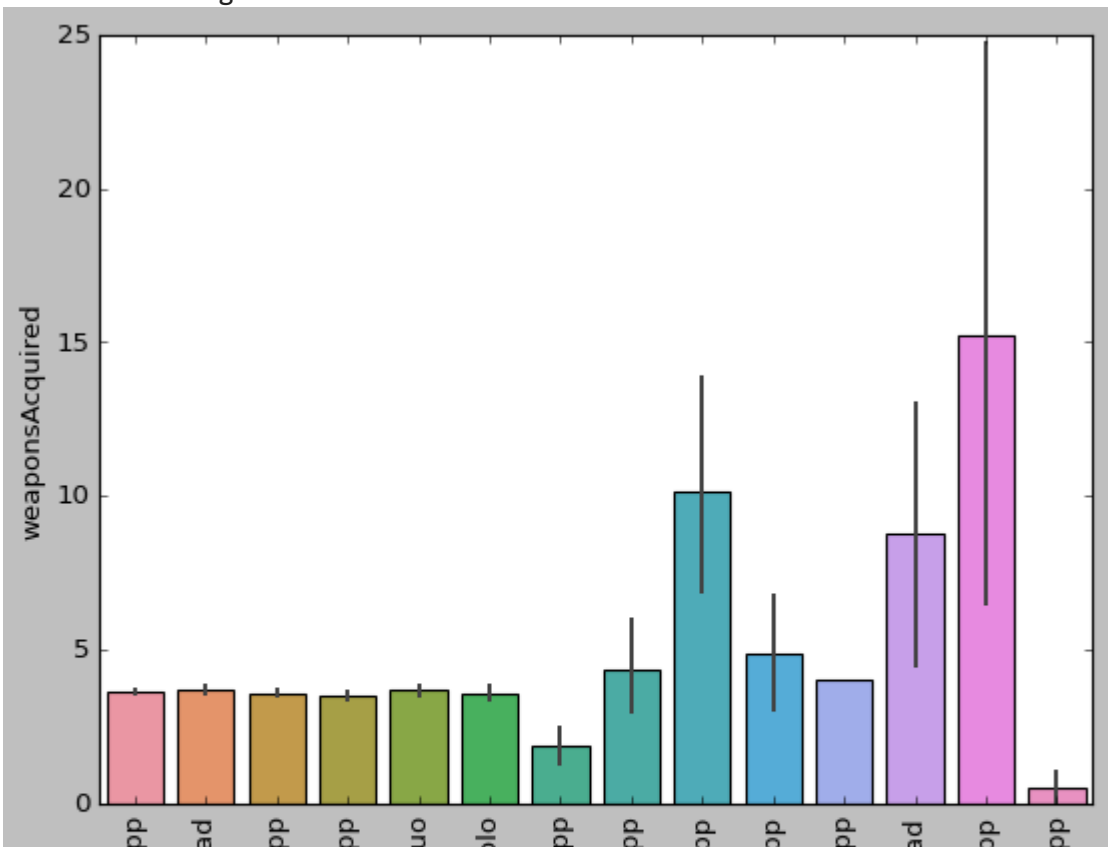
```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
```



15) Plot a barplot of'matchType' vs 'weaponsAcquired'. Write your inferences.

```
sns.barplot(pro_df['matchType'],pro_df['weaponsAcquired']);
plt.xticks(rotation=90);
```

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
```



16) Find the Categorical columns.

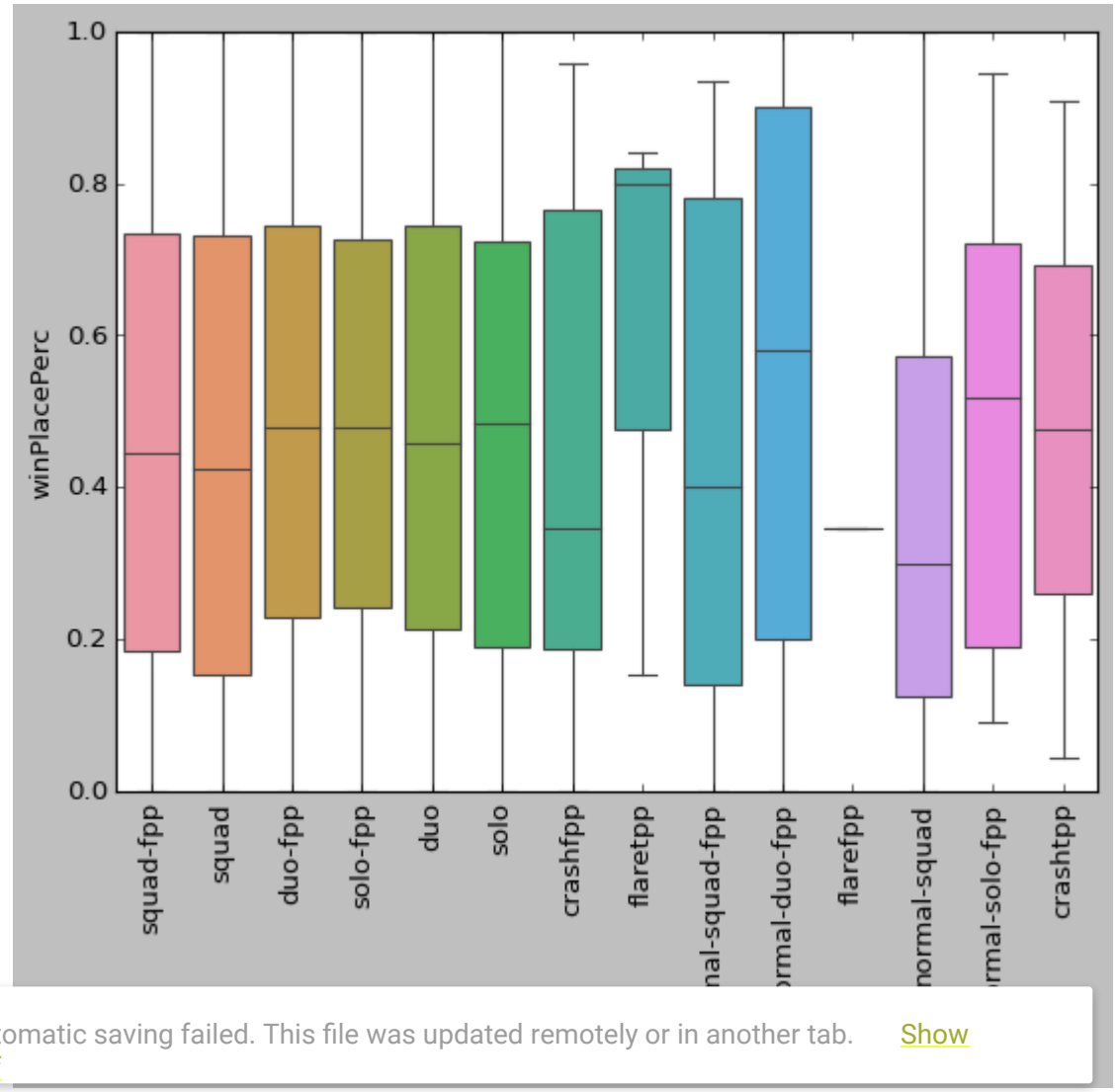
```
pro_df.select_dtypes('object')
```

	Id	groupId	matchId	matchType
0	2f262dd9795e60	78437bcd91d40e	d5db3a49eb2955	squad-fpp
3	f589dd03b60bf2	804ab5e5585558	c4a5676dc91604	squad
4	c23c4cc5b78b35	b3e2cd169ed920	cd595700a01bfa	squad-fpp
...	...	...	...	...
9995	ef4f474acd8e85	2eca2a8391f75d	492ecdfae90b46	squad-fpp
9996	cf0bf82fb4d80e	2eaf2765f93adb	14bffd71e96320	duo-fpp
9997	a0a31a0b1dcbe1	8d50c64ccc5071	147e4bbb62e3bb	duo-fpp
9998	f6874657399d69	d31843d7e62ccb	662567dcf280f5	duo-fpp
9999	90359b0b8f8b0d	61d5b1bb8da43f	258bfa48d88014	solo

10000 rows × 4 columns

17) plot a boxplot of 'matchType' vs 'winPlacePerc'. Write your inferences.

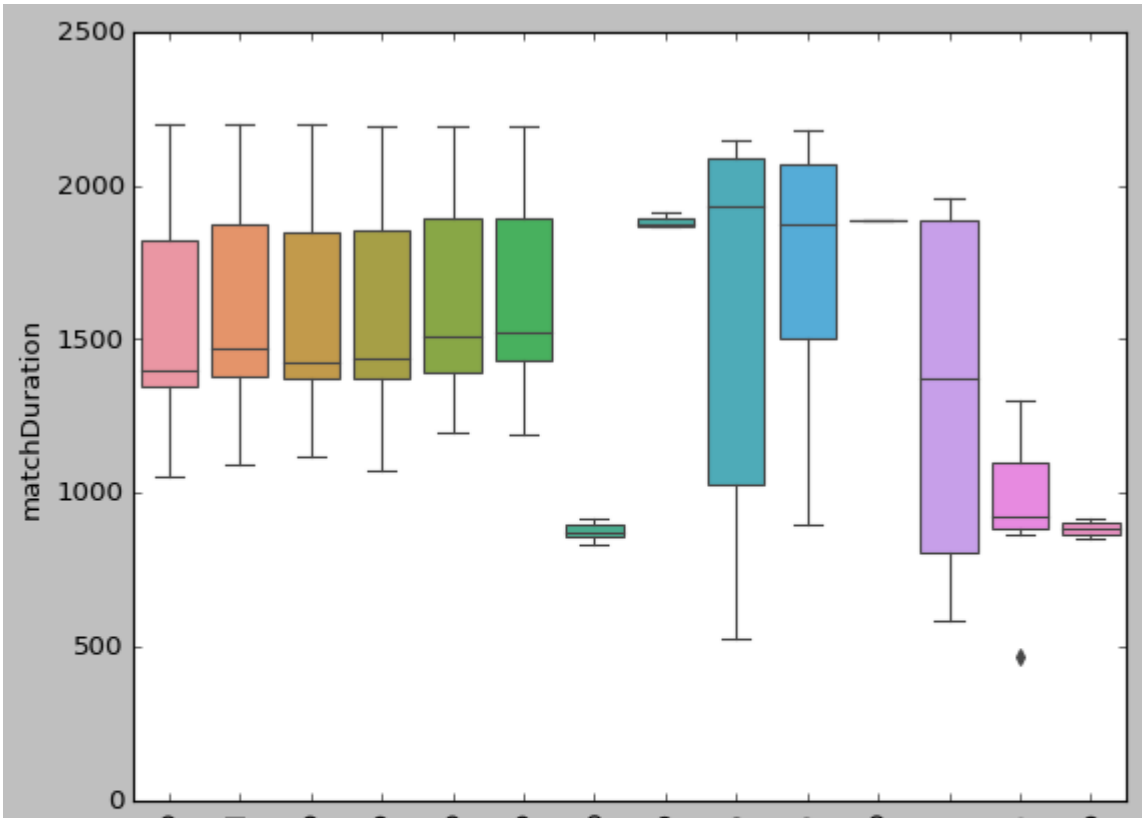
```
sns.boxplot(x='matchType',y='winPlacePerc',data=pro_df)
plt.xticks(rotation=90);
```



Automatic saving failed. This file was updated remotely or in another tab. Show diff

18) Plot a boxplot of 'matchType' vs 'matchDuration'. Write your inferences.

```
sns.boxplot(x='matchType',y='matchDuration',data=pro_df)
plt.xticks(rotation=90);
```



19) Change the orientation of the above plot to horizontal.

```
sns.boxplot(pro_df.matchDuration,pro_df.matchType)
plt.xticks(rotation=90);
```

Automatic saving failed. This file was updated remotely or in another tab. Show diff

```
/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass th
  FutureWarning
```

20) Add a new column called 'KILL' which contains the sum of following columns viz. headshotKills, teamKills, roadkills.

```
pro_df['KILL']=pro_df['headshotKills']+pro_df['teamKills']+pro_df['roadKills']
pro_df.head()
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs
0	2f262dd9795e60	78437bcd91d40e	d5db3a49eb2955	0	0	0.0	0
1	a32847cf5bf34b	85b7ce5a12e10b	65223f05c7fdb4	0	0	163.2	1
2	1b1900a9990396	edf80d6523380a	1cadec4534f30a	0	3	278.7	2
3	f589dd03b60bf2	804ab5e5585558	c4a5676dc91604	0	0	191.9	1
4	c23c4cc5b78b35	b3e2cd169ed920	cd595700a01bfa	0	0	100.0	1

21) Round off column 'winPlacePerc' to 2 decimals.

```
pro_df['winPlacePerc']=pro_df['winPlacePerc'].round(decimals=2)
pro_df.head()
```

	Id	groupId	matchId	assists	boosts	damageDealt	DBNOs
						0.0	0
						163.2	1
2	1b1900a9990396	edf80d6523380a	1cadec4534f30a	0	3	278.7	2
3	f589dd03b60bf2	804ab5e5585558	c4a5676dc91604	0	0	191.9	1
4	c23c4cc5b78b35	b3e2cd169ed920	cd595700a01bfa	0	0	100.0	1

22) Take a sample of size 50 from the column damageDealt for 100 times and calculate its mean. Plot it on a histogram and comment on its distribution.

```
sam_100= []
for i in range(100):
    sample = pro_df['damageDealt'].sample(n=50)
    sam_100.append(sample.mean())
```

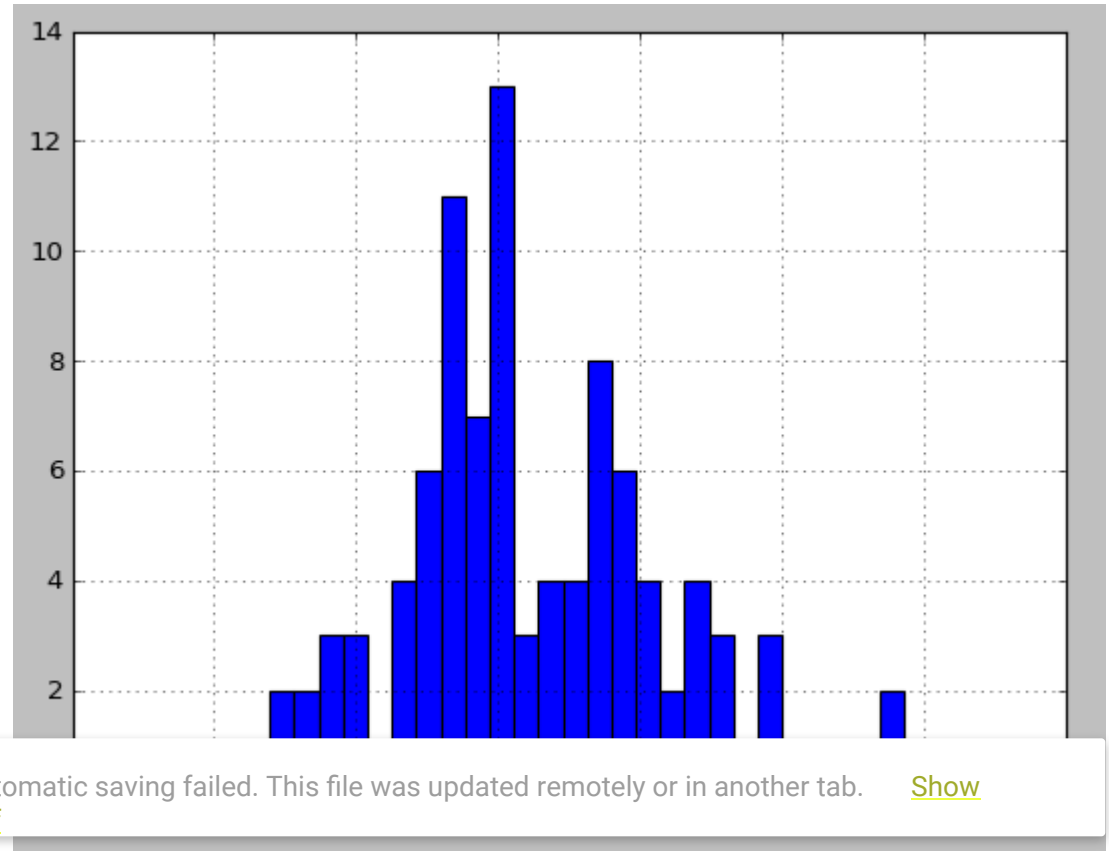
```
    sam_100.append(sample.mean())
bootstarp_avg = sum(sam_100)/100
bootstarp_avg
```

```
125.67906697999999
```

```
pro_df['damageDealt'].mean()
```

```
129.2112641000002
```

```
plt.hist(sam_100,bins=30);
plt.grid()
```



Automatic saving failed. This file was updated remotely or in another tab. Show diff