1. Read the dataset.

In [31]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as ps
import seaborn as sb
import numpy as np

pubg=ps.read_csv("pubg.csv")
```

2. List the datatype of all the columns.

In [32]:
```
pubg.dtypes
```

```
killPoints           int64
kills                int64
killStreaks          int64
longestKill        float64
matchDuration        int64
matchType           object
maxPlace             int64
numGroups            int64
rankPoints           int64
revives              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.

In [33]:
```
pubg.describe()
```

Out[33]:

	assists	boosts	damageDealt	DBNOs	headshotKills	heals	kil
count	10000.000000	10000.000000	10000.000000	10000.00000	10000.000000	10000.000000	10000.0
mean	0.234600	1.088500	129.211264	0.64400	0.221700	1.354000	47.6
std	0.575149	1.703279	167.193945	1.09562	0.577046	2.629102	27.4
min	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	1.0
25%	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	24.0
50%	0.000000	0.000000	83.805000	0.00000	0.000000	0.000000	48.0
75%	0.000000	2.000000	185.325000	1.00000	0.000000	2.000000	71.0
max	7.000000	18.000000	3469.000000	11.00000	14.000000	31.000000	100.0

8 rows × 25 columns

With only one function(i.e., describe()) we can find the Mean, Standard Deviation, Min Value, Max Value and more Statistical analysis for a dataset. This function only processses the Numerical Valued Columns

1. The dataset consists of total 20 columns of numerical datatype
2. mean gives the average value of that column
3. std gives the standard deviation of that column
4. min gives the minimum value present in that column
5. max gives the maximum value present in that column
6. 25% gives the 25% of that Particular Column
7. 50% gives the 50% of that Particular Column
8. 75% gives the 75% of that Particular Column

4. The average person kills how many players

In [34]:
```
pubg['kills'].mean()
```

Out[34]:
```
0.9134
```

5. 99% of people have how many kills

In [35]:
```
np.percentile(pubg['kills'],99)
```

Out[35]:
```
7.0
```

6. The most kills ever recorded are how much

In [36]:
```
pubg['kills'].max()
```

Out[36]:
```
35
```

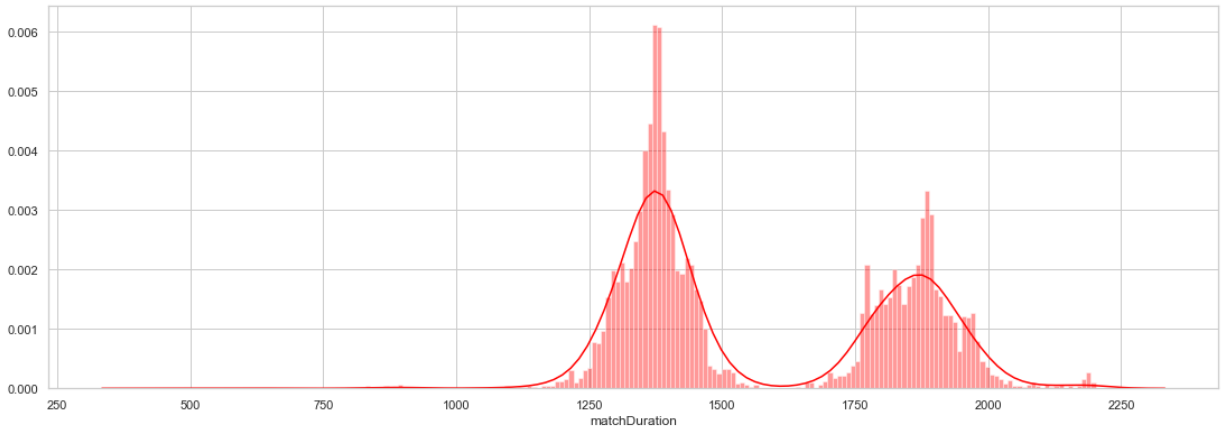
7. Print all the columns of the dataframe.

In [37]:
```
list(pubg.columns)
```

```
 'killPlace',
 'killPoints',
 'kills',
 'killStreaks',
 'longestKill',
 'matchDuration',
 'matchType',
 'maxPlace',
 'numGroups',
 'rankPoints',
 'revives',
 'rideDistance',
 'roadKills',
 'swimDistance',
 'teamKills',
 'vehicleDestroys',
 'walkDistance',
 'weaponsAcquired',
 'winPoints',
 'winPlacePerc']
```

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

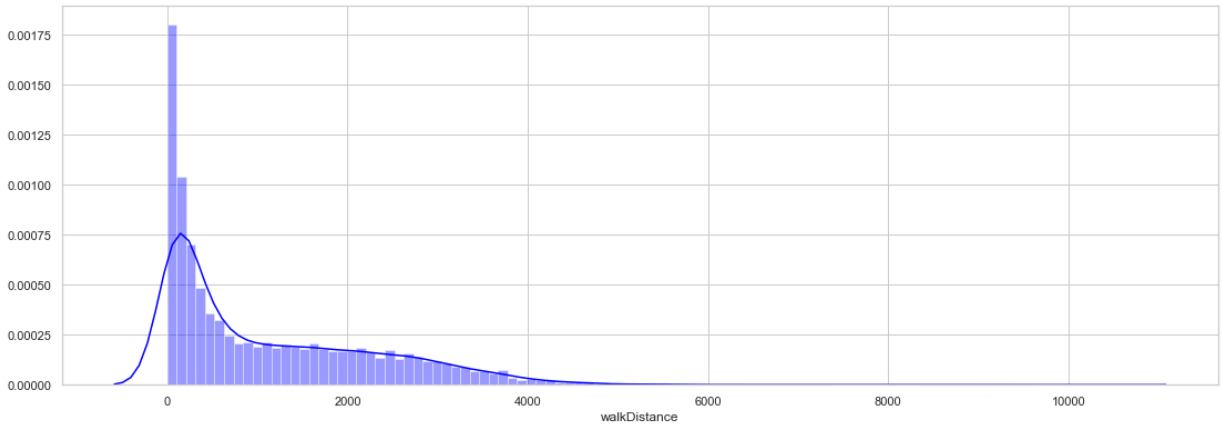
In [69]:
```
sb.set_style('whitegrid')
sb.distplot(pubg["matchDuration"], color ='red', bins = 200)
plt.show()
```




```
# By looking at this we can say that:
    1. min value is 464
    2. max values is 2202
    3. std is 258.96
    4. mean is 1575.39
```

9. Comment on distribution of the walk distance. Use seaborn

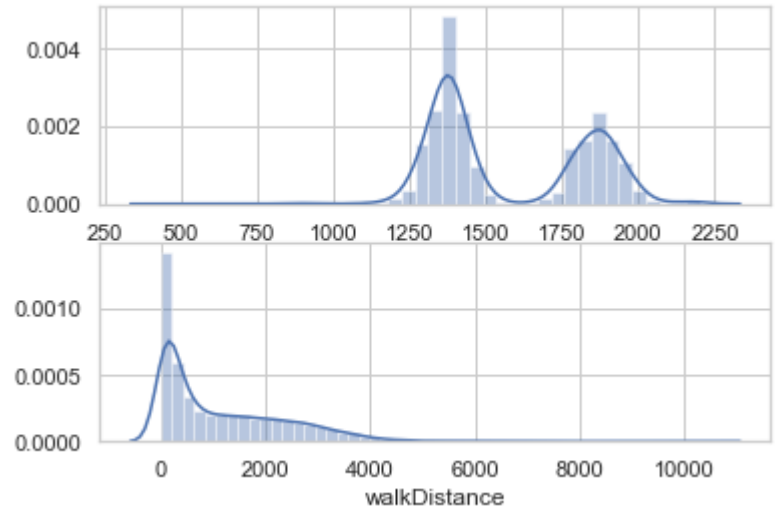
In [29]:
```
sb.set_style('whitegrid')
sb.distplot(pubg["walkDistance"], color ='blue', bins = 100)
plt.show()
```




```
# By looking at this we can conclude that :
    1. min value is 0
    2. max values is 10490
    3. std is 1168.59
    4. mean is 1130.00
```

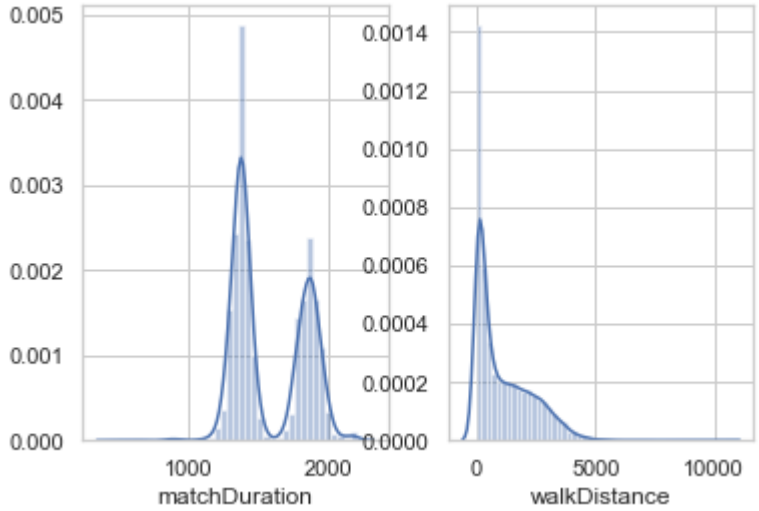
10. Plot distribution of the match's duration vs walk distance one below the other.

In [40]:
```
fig,axs=plt.subplots(2,1)
sb.distplot(pubg.matchDuration,ax=axs[0]);
sb.distplot(pubg.walkDistance,ax=axs[1]);
```



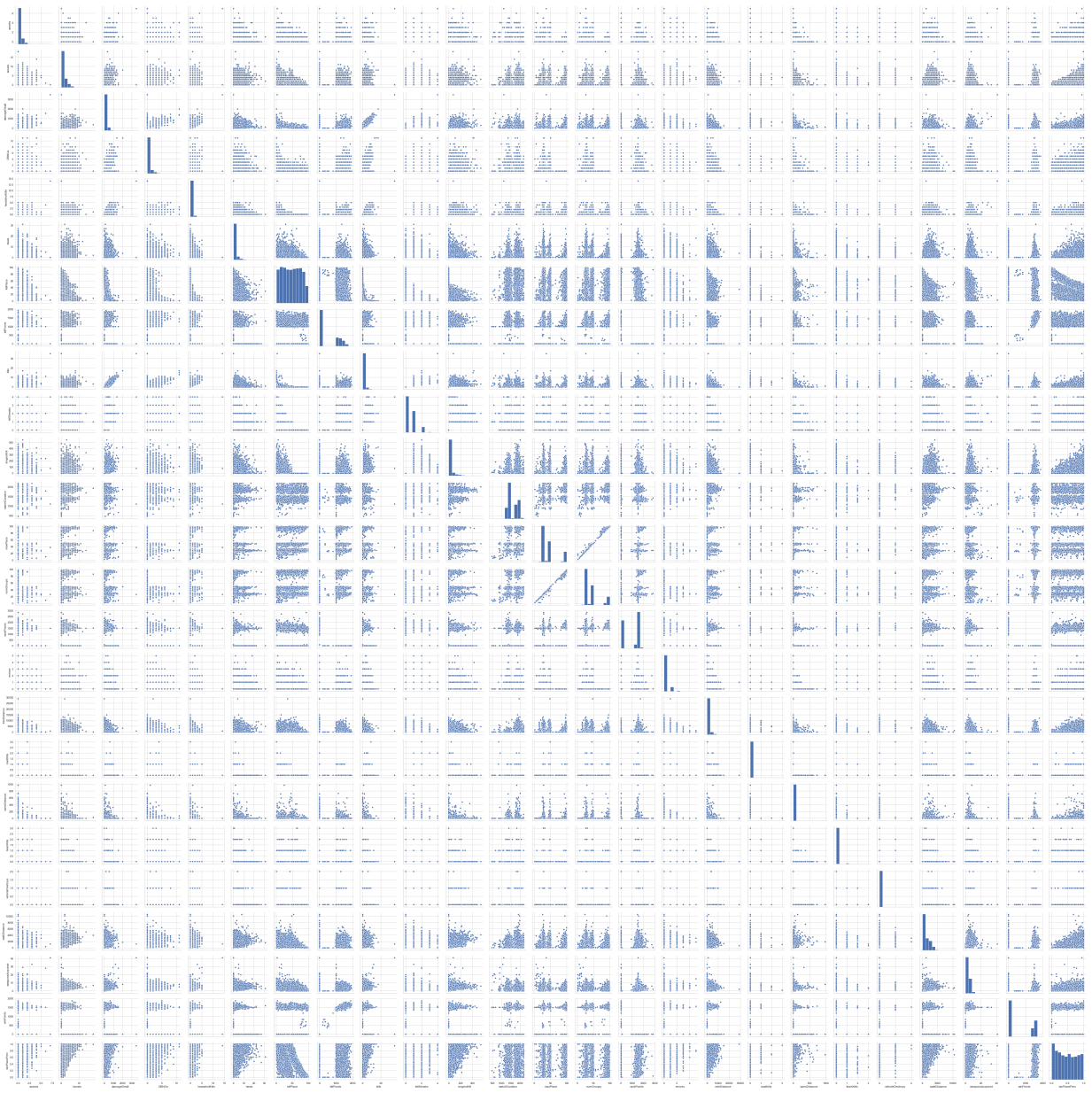
11. Plot distribution of the match's duration vs walk distance side by side.

In [42]:
```
import matplotlib.pyplot as plt
fig,axs=plt.subplots(1,2)
sb.distplot(pubg.matchDuration,ax=axs[0]);
sb.distplot(pubg.walkDistance,ax=axs[1]);
```



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

In [44]:
```
sb.pairplot(pubg);
```



```
Comment on kills vs damage dealt
Kills and Damage dealt have linear relationship
```

```
Comment on maxPlace vs numGroups
Kills and Damagae dealt numGroups have linear relationship
```

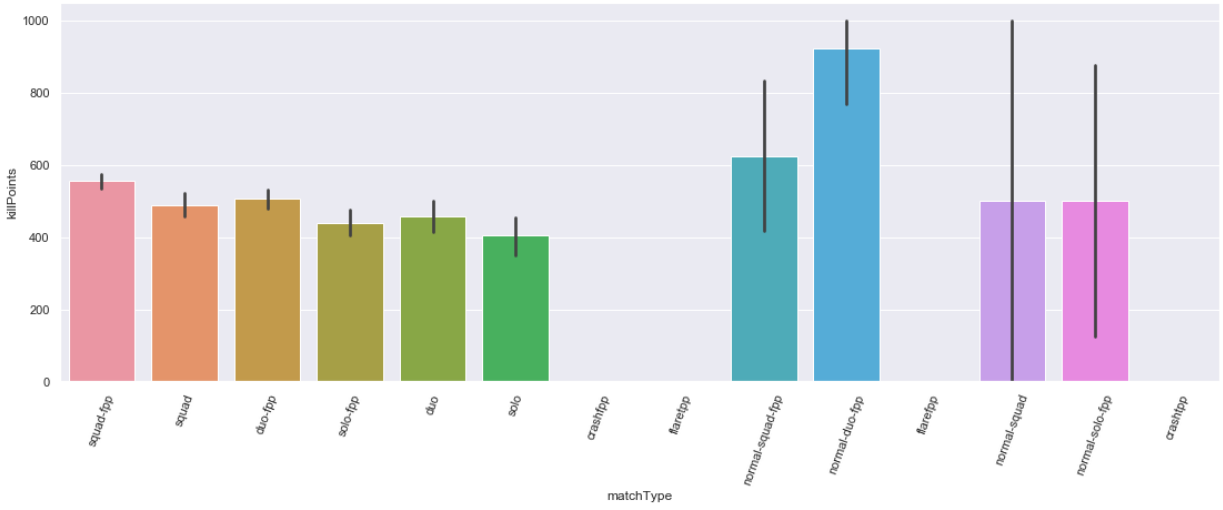
13. How many unique values are there in 'matchType' and what are their counts?

In [46]:
```
pubg.matchType.value_counts()
```

Out[46]:
```
squad-fpp           3969
duo-fpp             2282
squad               1359
solo-fpp            1234
duo                  702
solo                 386
normal-squad-fpp      24
normal-duo-fpp        13
crashfpp              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 'killPoints'. Write your inferences.

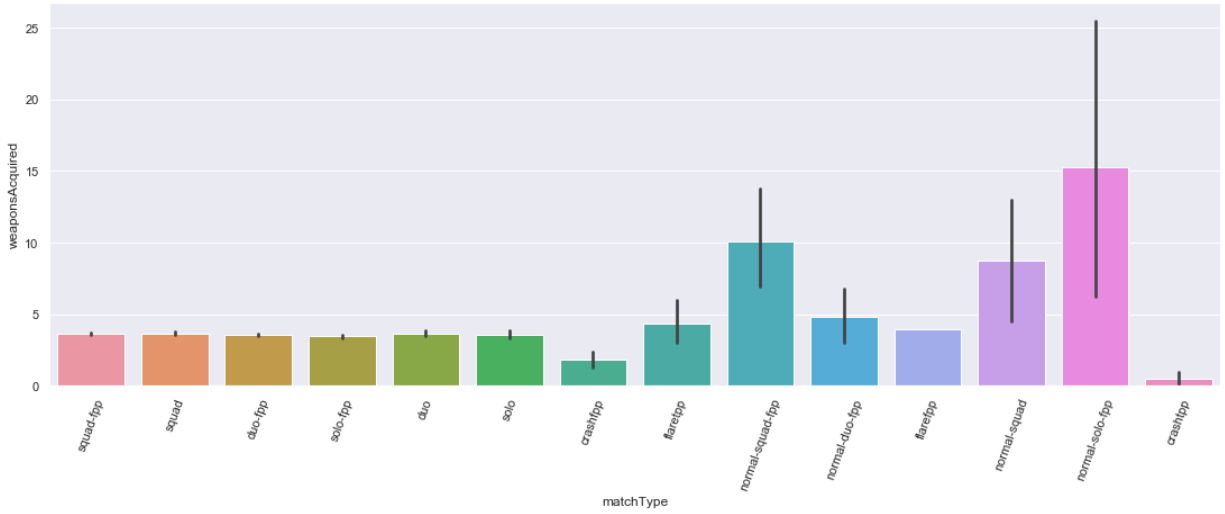
In [53]:
```
sb.set(rc={'figure.figsize':(18.7,6.27)})
sb.barplot(x="matchType", y="killPoints", data=pubg)
plt.xticks(rotation=70)
plt.show()
```




```
# For crashtpp,crashfpp,flaretpp,flareftp there won't be any killPoints for
these type of Matches
# Where as normal-duo-fpp has the most killPoints
```

15. Plot a barplot of ‘matchType’ vs ‘weaponsAcquired’. Write your inferences.

In [52]:
```
sb.set(rc={'figure.figsize':(18.7,6.27)})
sb.barplot(x="matchType", y="weaponsAcquired", data=pubg)
plt.xticks(rotation=70)
plt.show()
```




```
# Weapons acquired in crashtpp matchType are very less
# Weapons acquired in normal-solo-fpp matchType are very high
# Weapons acquired in squad-fpp, squad, duo-fpp, solo-fpp, duo, solo
matchTypes are almost similar
```

16. Find the Categorical columns.

In [55]:
```
pubg.select_dtypes(exclude=['number'])
```

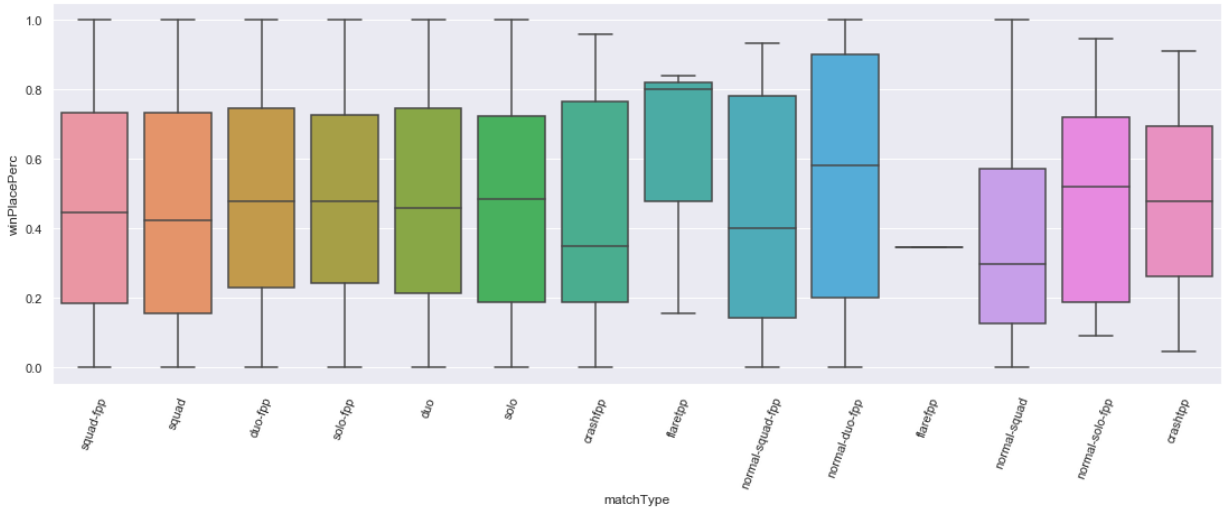
Out[55]:

	Id	groupId	matchId	matchType
0	2f262dd9795e60	78437bcd91d40e	d5db3a49eb2955	squad-fpp
1	a32847cf5bf34b	85b7ce5a12e10b	65223f05c7fdb4	squad-fpp
2	1b1900a9990396	edf80d6523380a	1cadec4534f30a	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.

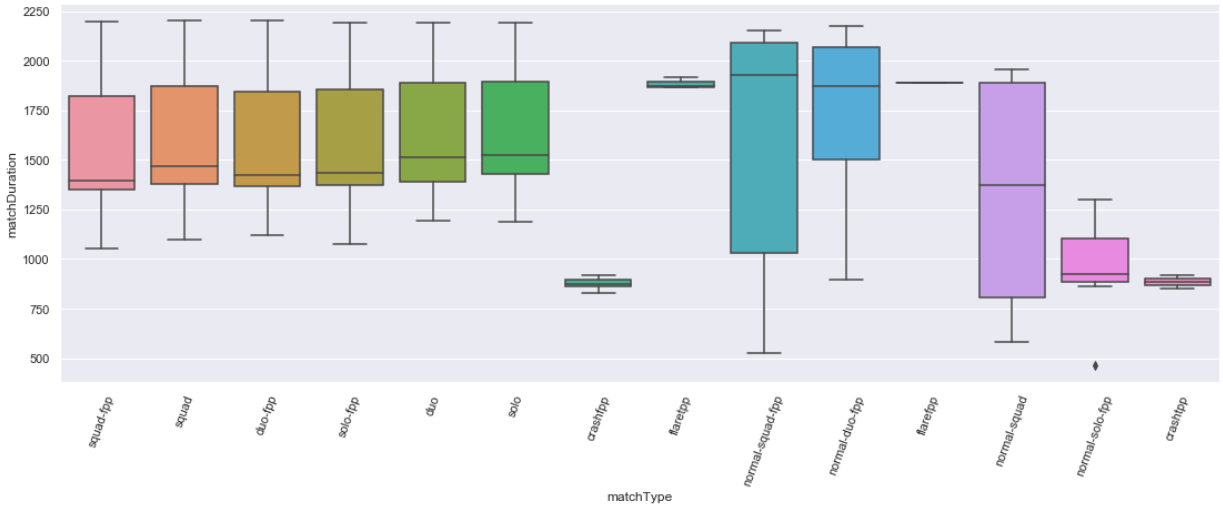
In [61]:
```
sb.boxplot(pubg.matchType,pubg.winPlacePerc)
plt.xticks(rotation=70)
plt.show()
```




```
# Almost every match type has the first and last place in Win percentile
# Most of the match types have the win place percentile between 0.4 to 0.6
# flarefpp match type has only median percentile
```

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

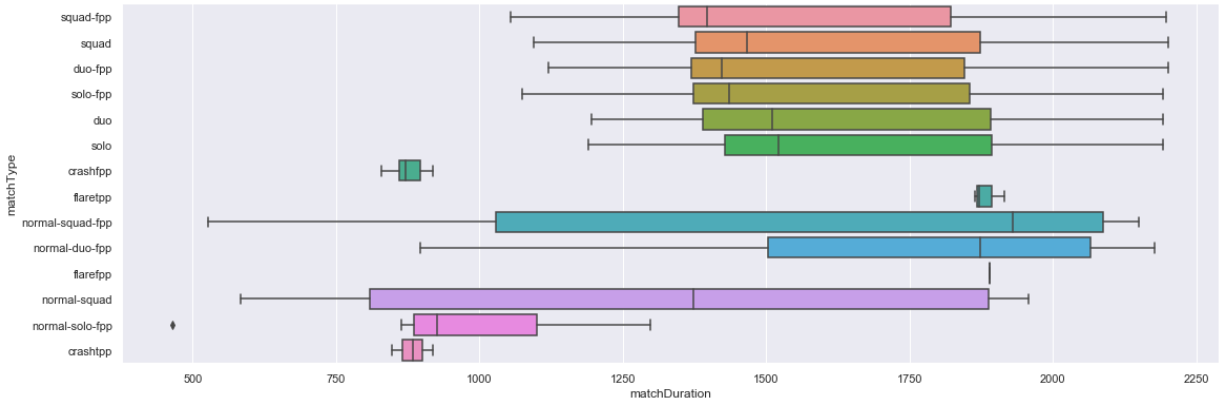
In [60]:
```
sb.boxplot(pubg.matchType,pubg.matchDuration);
plt.xticks(rotation=70)
plt.show()
```




```
# matchDuration for the matchtypes like crashfpp, crashtpp is between 750 to
1000
# matchDuration for the matchtypes like flarefpp, flaretpp is between 1750 to
2000
# Most of the matchtypes has matchDuration with min duration between 1000 to
1250 and max duration between 2000 to 2250
```

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

In [63]:
```
sb.boxplot(pubg.matchDuration,pubg.matchType)
plt.show()
```



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

In [66]:
```
pubg["KILL"]=pubg["headshotKills"]+pubg["roadKills"]+pubg["teamKills"]
kd=ps.DataFrame(pubg[['headshotKills','roadKills','teamKills','KILL']])
print(kd.head())
print(kd.tail())
list(pubg.columns)
```

```
   headshotKills  roadKills  teamKills  KILL
0              0          0          0     0
1              1          0          0     1
2              1          0          0     1
3              0          0          0     0
4              0          0          0     0
      headshotKills  roadKills  teamKills  KILL
9995              0          0          0     0
9996              0          0          0     0
9997              0          0          0     0
9998              0          0          0     0
9999              0          0          0     0
```

Out[66]:
```
['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',
 'KILL']
```

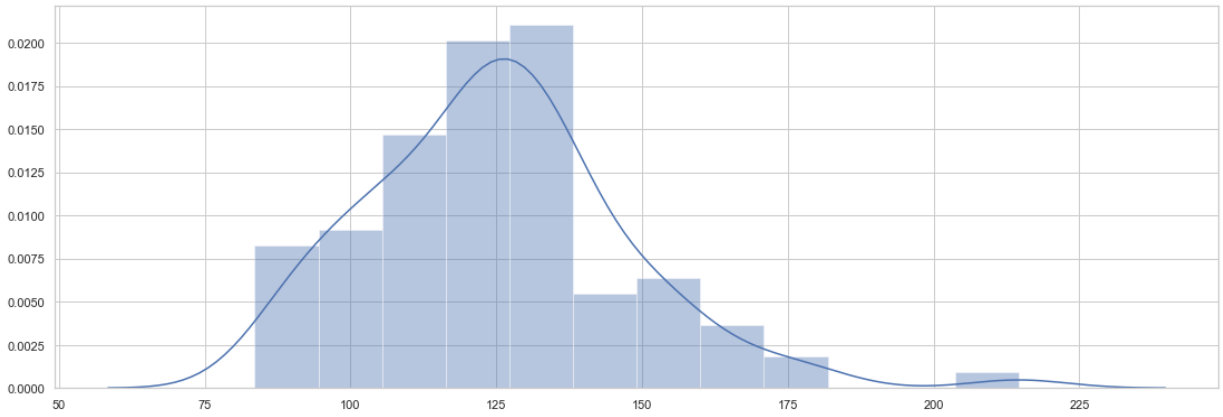
21. Round off column 'winPlacePerc' to 2 decimals.

In [67]:
```
pubg["winPlacePerc"].round(2)
```

Out[67]:
```
0       0.00
1       0.22
2       0.86
3       0.35
4       0.07
        ...
9995    0.83
9996    0.72
9997    0.21
9998    0.24
9999    0.19
Name: winPlacePerc, Length: 10000, dtype: float64
```

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.

In [70]:
```
x = []
for i in range(100):
    x.append(pubg['damageDealt'].sample(50).mean())
means = np.array(x)
sb.distplot(means);
```




```
# The mean increases first (upto in between 100 to 125 ) then it decreases and
finally its constant in between 175 to 225
```

Thanks Let's Upgarde