Exploratory Data Analysis and visualization for Player Rank prediction in a PUBG Tournament

¶

Name : Keerthi Prasad Ganganna

Student No :18200322

Name : Manoj Kumar Sugumaran

Student No :18200062

Table of Contents

In [34]:

```
import requests
import pandas as pd
import time
import seaborn as sns
from matplotlib import pyplot as plt
from IPython.display import display


```

executed in 4ms, finished 15:19:41 2019-04-30

Task 1 : Identify one or more suitable web APIs

API Chosen:

PUBG official API

Task 2 : Collecting Data from APIs

Getting Data using Authorization Header

Getting Leaderboard data from [https://api.pubg.com/shards/steam/leaderboards/{gameMode}](https://api.pubg.com/shards/steam/leaderboards/%7BgameMode%7D) API

In [17]:

```
def write_to_df(r):
    df = pd.DataFrame(columns=['accountid', 'name', 'rank'])
    for temp in r['data'] :
        name = temp['attributes']['name']
        rank = temp['attributes']['rank']
        accountid = temp['id'].replace('account.','')
        df.loc[len(df)] = [accountid, name, rank]

    df = df.sort_values(by=['rank'])
    return df
```

executed in 8ms, finished 15:11:13 2019-04-30

In [ ]:

```
leaderboard_df = pd.DataFrame(columns=['accountid', 'name', 'rank'])

for i in range(0,2):

    url = "https://api.pubg.com/shards/steam/leaderboards/squad?page[number]="+

    header = {
      "Authorization": "Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJqdGkiOiI
      "Accept": "application/vnd.api+json"
    }

    r = requests.get(url, headers=header)

    r = r.json()
    leaderboard_df = leaderboard_df.append(write_to_df(r))

```

executed in 923ms, finished 15:11:14 2019-04-30

In [22]:

```
r['data']['attributes']
```

executed in 7ms, finished 15:12:09 2019-04-30

Out[22]:

```
{'shardId': 'steam', 'gameMode': 'squad'}
```

Task 3: Parse the collected data, and store it in an appropriate file format

Storing data in a CSV file

In [ ]:

```

#write to csv
leaderboard_df.to_csv("PUBG_leaderboard.csv", sep='\t',index=False)

```

executed in 11.4s, finished 15:06:50 2019-04-30

In [ ]:

```
leaderboard = pd.read_csv('PUBG_leaderboard.csv', sep='\t')
```

executed in 11.1s, finished 15:06:50 2019-04-30

In [ ]:

```
leaderboard.info()
```

executed in 10.8s, finished 15:06:50 2019-04-30

In [ ]:

```
leaderboard.head()
```

executed in 10.4s, finished 15:06:50 2019-04-30

Getting Player data from https://api.pubg.com/shards/steam/players/{accountId} (https://api.pubg.com/shards/steam/players/%7BaccountId%7D) API

In [23]:

```
player_df = pd.DataFrame(columns=['assists','bestRankPoint','boosts','dBNOs','d

#player_df = get_player_data_api(player_df)
#write to csv
#player_df.to_csv("PUBG_players_data.csv", sep='\t',index=False)
```

executed in 32ms, finished 15:13:36 2019-04-30

In [24]:

```
def get_player_data_api(player_df):
    count = 10
    for i in range(0,500):
        if(count == 10):
            time.sleep(60)
            count = 0
        url = "https://api.pubg.com/shards/steam/players/"+str(leaderboard['acc
        header = {
          "Authorization": "Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJqdGk
          "Accept": "application/vnd.api+json"
        }
        r = requests.get(url, headers=header)
        r = r.json()
        df = pd.DataFrame()
        df = df.append(r['data']['attributes']['gameModeStats']['squad'], ignor
        df['accountid'] = leaderboard['accountid'][i]
        df['rank'] = leaderboard['rank'][i]
        player_df = player_df.append(df)
        count+=1
        return player_df


```

executed in 17ms, finished 15:13:37 2019-04-30

Task 4: Load and represent the data

Using an appropriate data structure. Apply any pre-processing steps to clean/filter/combine the data

Load the CSV into a Pandas data structure

In [25]:

```
player_data = pd.read_csv('PUBG_players_data.csv', sep='\t')
```

executed in 319ms, finished 15:13:39 2019-04-30

In [26]:

```
1 player_data.info()
```

executed in 28ms, finished 15:13:39 2019-04-30

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 900 entries, 0 to 899
Data columns (total 38 columns):
assists                 900 non-null int64
bestRankPoint           900 non-null float64
boosts                  900 non-null int64
dBNOs                   900 non-null int64
dailyKills              900 non-null int64
dailyWins               900 non-null int64
damageDealt             900 non-null float64
days                    900 non-null int64
headshotKills           900 non-null int64
heals                   900 non-null int64
killPoints              900 non-null int64
kills                   900 non-null int64
longestKill             900 non-null float64
longestTimeSurvived     900 non-null float64
losses                  900 non-null int64
maxKillStreaks          900 non-null int64
mostSurvivalTime        900 non-null float64
rankPoints              900 non-null int64
rankPointsTitle         0 non-null float64
revives                 900 non-null int64
rideDistance            900 non-null float64
roadKills               900 non-null int64
roundMostKills          900 non-null int64
roundsPlayed            900 non-null int64
suicides                900 non-null int64
swimDistance            900 non-null float64
teamKills               900 non-null int64
timeSurvived            900 non-null float64
top10s                  900 non-null int64
vehicleDestroys         900 non-null int64
walkDistance            900 non-null float64
weaponsAcquired         900 non-null int64
weeklyKills             900 non-null int64
weeklyWins              900 non-null int64
winPoints               900 non-null int64
wins                    900 non-null int64
accountid               900 non-null object
rank                    900 non-null int64
dtypes: float64(10), int64(27), object(1)
memory usage: 267.3+ KB
```

Information on the player API attributes

assists - Number of enemy players this player damaged that were killed by teammates

bestRankPoint - Highest number of rank points the player was awarded

boosts - Number of boost items used

dBNOs - Number of enemy players knocked

dailyKills - Number of kills during the most recent day played.

damageDealt - Total damage dealt. Note- Self inflicted damage is subtracted

days - Number of days continously played

dailyWins - Number of wins during the most recent day played.

headshotKills - Number of enemy players killed with headshots

heals - Number of healing items used

killPoints - Kill points obtained by number of enemy players killed

kills - Number of enemy players killed

longestKill - Enemy killed from a long distance

longestTimeSurvived - Longest time survived in a match

losses - Number of matches lost

maxKillStreaks - Enemy killed from a long distance

mostSurvivalTime - Longest time survived in a match

rankPoints - Number of rank points the player was awarded. This value will be 0 when roundsPlayed < 10

rankPointsTitle - Rank title in the form title-level

revives Number of times this player revived teammates

rideDistance - Total distance traveled in vehicles measured in meters

roadKills - Number of kills while in a vehicle

roundMostKills - Highest number of kills in a single match

roundsPlayed - Number of matches played

suicides - Number of self-inflicted deaths

swimDistance - Total distance traveled while swimming measured in meters

teamKills - Number of times this player killed a teammate

timeSurvived - Total time survived

top10s - Number of times this player made it to the top 10 in a match

vehicleDestroys - Number of vehicles destroyed

walkDistance - Total distance traveled on foot measured in meters

weaponsAcquired - Number of weapons picked up

weeklyKills - Number of kills during the most recent week played

weeklyWins - Number of wins during the most recent week played.

winPoints - number

wins - Number of matches won

In [27]:

```
1 player_data.head()
```

executed in 117ms, finished 15:13:41 2019-04-30

Out[27]:

	assists	bestRankPoint	boosts	dBNOs	dailyKills	dailyWins	damageDealt	days	headshotl
0	4053	7887.4570	9089	9123	17	0	1334558.8	152	2
1	4080	7479.5060	11902	8339	15	0	1307706.0	154	2
2	3078	7420.9287	8659	10238	103	0	1493163.5	112	3
3	4900	7396.7200	10411	12510	28	3	1811403.9	165	3
4	3581	7291.7780	11080	8037	40	0	1180883.9	156	2

5 rows × 38 columns

Task 5: Analyse and summarise the cleaned dataset

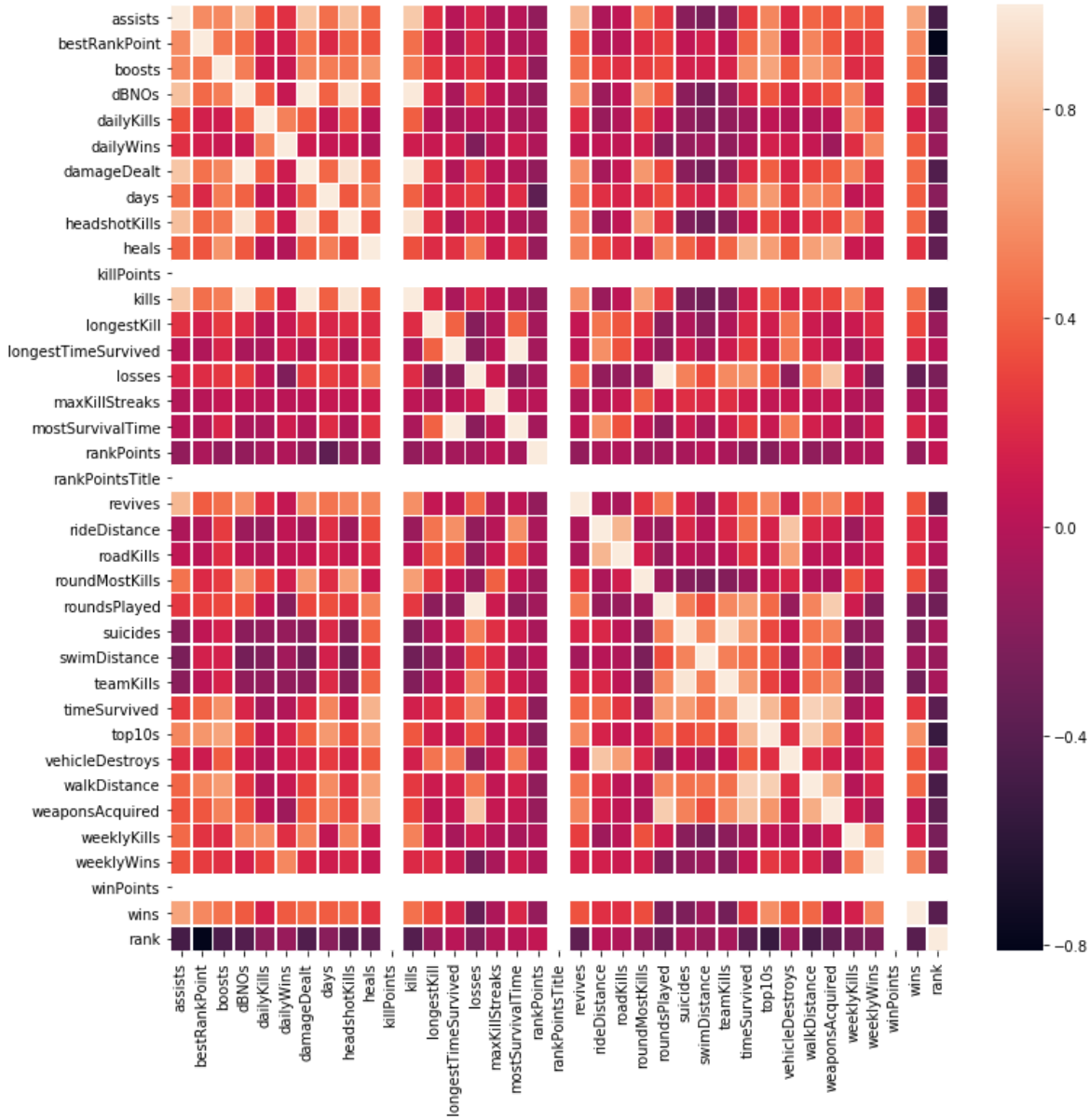
Exploratory Data Analysis

Correlation plot for the attrbutes obtained through the API

In [28]:

```
fig,ax = plt.subplots(figsize=(12, 12))
sns.heatmap(player_data.corr(), annot=False, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```

executed in 1.77s, finished 15:13:44 2019-04-30



Removing unwanted features

Removing the features that are deprecated by the API provider - 'killPoints', 'rankPointsTitle', 'winPoints'

In [29]:

```
player_data = player_data.drop(['killPoints', 'rankPointsTitle', 'winPoints'],
#player_data columns after removing -'killPoints', 'rankPointsTitle', 'winPoint
player_data.columns
```

executed in 10ms, finished 15:13:46 2019-04-30

Out[29]:

```
Index(['assists', 'bestRankPoint', 'boosts', 'dBNOs', 'dailyKills',
       'dailyWins', 'damageDealt', 'days', 'headshotKills', 'heals',
'kills',
       'longestKill', 'longestTimeSurvived', 'losses', 'maxKillStreak
s',
       'mostSurvivalTime', 'rankPoints', 'revives', 'rideDistance',
       'roadKills', 'roundMostKills', 'roundsPlayed', 'suicides',
       'swimDistance', 'teamKills', 'timeSurvived', 'top10s',
       'vehicleDestroys', 'walkDistance', 'weaponsAcquired', 'weeklyKi
lls',
       'weeklyWins', 'wins', 'accountid', 'rank'],
      dtype='object')
```

Now to explore the data for finding different type of players

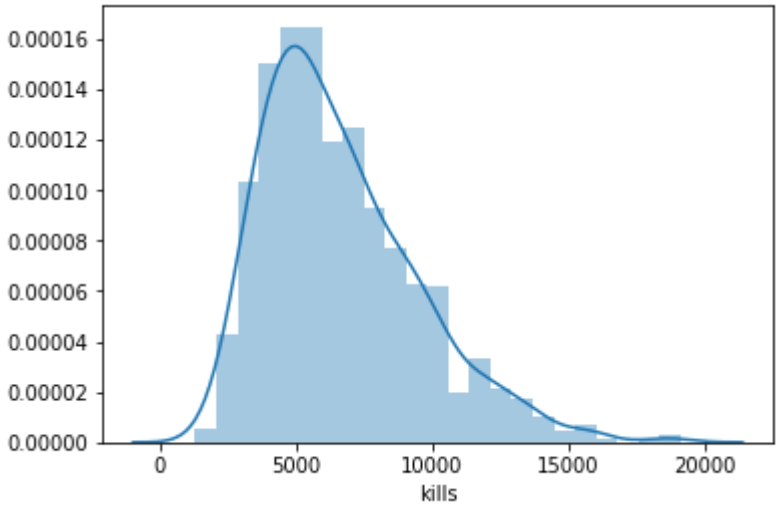
Terminators

In [30]:

```
1 #distribution plot for kills
2 sns.distplot(player_data['kills'])
3 display("The average person on the leaderboard has killed {:.0f} players, 99% o
4
```

executed in 617ms, finished 15:13:49 2019-04-30

```
'The average person on the leaderboard has killed 6656 players, 99% of
people have killed 15234 or less, while the most kills ever recorded i
s 19131.'
```

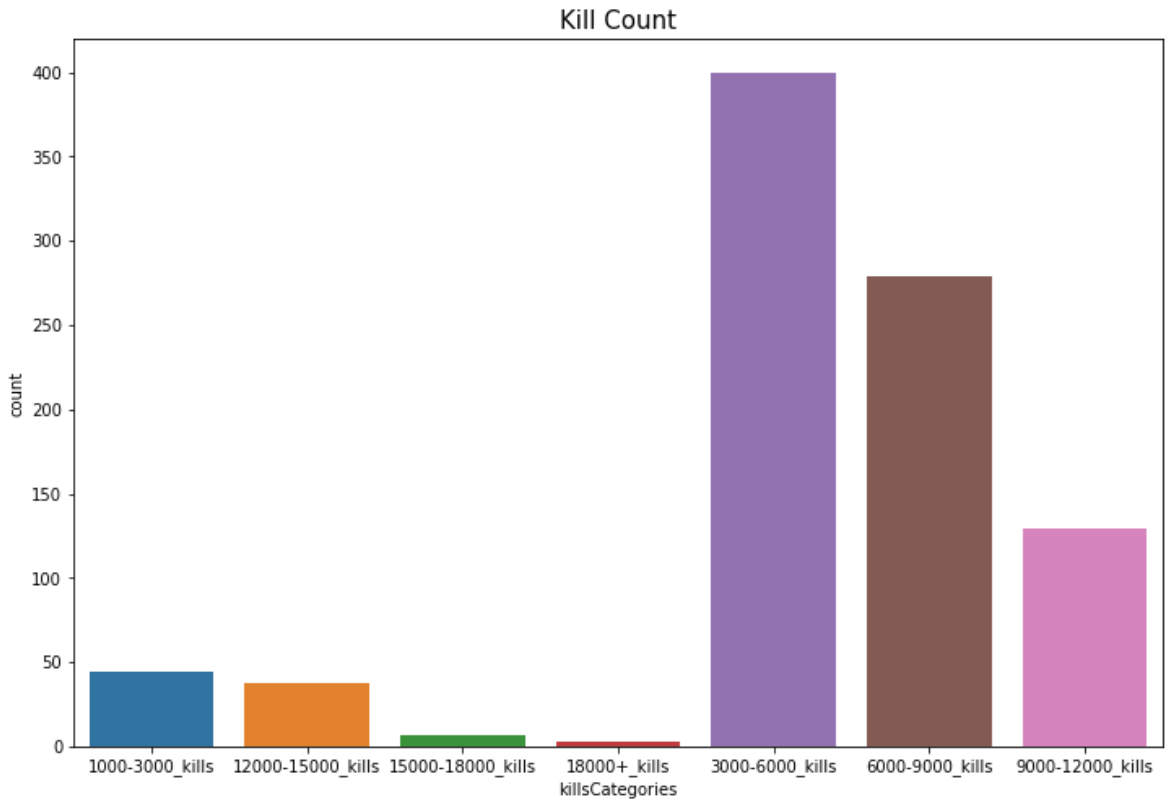


In [31]:

```
# bar plot for number of kills
kills = player_data.copy()
kills['killsCategories'] = pd.cut(kills['kills'], [0, 3000, 6000, 9000, 12000,

plt.figure(figsize=(12,8))
sns.countplot(kills['killsCategories'].astype('str').sort_values())
plt.title("Kill Count",fontsize=15)
plt.show()
```

executed in 445ms, finished 15:13:49 2019-04-30



In [32]:

```
#Getting top 1% players with most kills
terminators_df = player_data[player_data['kills'] > player_data['kills'].quanti
display(terminators_df)

```

executed in 84ms, finished 15:13:49 2019-04-30

	kills	rank	accountid
316	19131	317	48a9c9fe9ed7440b96845e91998f7c9c
23	18649	24	c7ff66ef44a242a7a81eef1096b8eda1
131	18216	132	f6a478cea6fa4249a846943286948662
12	16198	13	c77e50f09fbd4923bbae557b7e342589
121	15997	122	e8a93676626e4280be988aeb344cdabb
155	15993	156	bc4730a155bd48ac893e8b2f5290acba
33	15816	34	807bbb2a661d40a8929c5280d9729581
377	15704	378	39e7fa76538a430b9a1dc4a762b9ca74
3	15522	4	ab46de8032f24d799c7dd4e3c3305883

Though number of kills is important, it doesn't guarantee the top rank in the leaderboard

The Terminators in the leaderboard are

In [33]:

```
names = []
for accountid in terminators_df['accountid']:
    names.append(leaderboard[leaderboard['accountid'] == accountid]['name'].val

terminators_df['name'] = names
display(terminators_df)
```

executed in 457ms, finished 15:13:51 2019-04-30

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
<ipython-input-33-60e05ba7d472> in <module>
      1 names = []
      2 for accountid in terminators_df['accountid']:
----> 3     names.append(leaderboard[leaderboard['accountid'] == accou
ntid]['name'].values[0])
      4
      5 terminators_df['name'] = names

NameError: name 'leaderboard' is not defined
```

Deadly Snipers

In [ ]:

```
#distribution plot for longest kills
sns.distplot(player_data['longestKill'])
display("The average person on the leaderboard has killed from a distance of {:
```

executed in 232ms, finished 15:06:50 2019-04-30

In [ ]:

```
# bar plot for kill distance
kill_distance = player_data.copy()
kill_distance['killsDistanceCategories'] = pd.cut(kill_distance['longestKill'],

plt.figure(figsize=(8,8))
sns.countplot(kill_distance['killsDistanceCategories'].astype('str').sort_value
plt.title("Kill Distance",fontsize=15)
plt.show()
```

executed in 70ms, finished 15:06:50 2019-04-30

In [4]:

```
#Getting top 1% snipers on the leaderboard
snipers_df = player_data[player_data['longestKill'] > player_data['longestKill'
display(snipers_df)

```

executed in 25ms, finished 15:06:50 2019-04-30

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
 last)
<ipython-input-4-42e0d920c34d> in <module>
      1 #Getting top 1% snipers on the leaderboard
----> 2 snipers_df = player_data[player_data['longestKill'] > player_d
ata['longestKill'].quantile(0.99)].sort_values(by='longestKill', ascen
ding=False)[['longestKill','rank','accountid']]
      3 display(snipers_df)

NameError: name 'player_data' is not defined
```

LongestKill feature has very less correlation with the Rank in leaderboard. None of the top 1% of snipers are in the Top 200 of the leaderboard

The Snipers in the leaderboard are

In [5]:

```
names = []
for accountid in snipers_df['accountid']:
    names.append(leaderboard[leaderboard['accountid'] == accountid]['name'].val

snipers_df['name'] = names
display(snipers_df)
```

executed in 16ms, finished 15:06:50 2019-04-30

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
 last)
<ipython-input-5-eba7c6f35b96> in <module>
      1 names = []
----> 2 for accountid in snipers_df['accountid']:
      3     names.append(leaderboard[leaderboard['accountid'] == accou
ntid]['name'].values[0])
      4
      5 snipers_df['name'] = names

NameError: name 'snipers_df' is not defined
```

Marathon Man

In [6]:

```
1 #distribution plot for distance walked
2 sns.distplot(player_data['walkDistance'])
3 display("The average person on the leaderboard has walked a distance of {:.2f}
4
```

executed in 14ms, finished 15:06:51 2019-04-30

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
<ipython-input-6-7939b3711ca8> in <module>
      1 #distribution plot for distance walked
----> 2 sns.distplot(player_data['walkDistance'])
      3 display("The average person on the leaderboard has walked a di
stance of {:.2f} kilometre, 99% of people have walked  {:.2f} kilometr
e or less, while the longest walk ever recorded is {}. kilometres".for
mat(player_data['walkDistance'].mean()/1000,player_data['walkDistance'
].quantile(0.99)/1000, player_data['walkDistance'].max()/1000))

NameError: name 'player_data' is not defined
```

In [7]:

```
# bar plot for walk distance
kill_distance = player_data.copy()
kill_distance['walkDistance']/=1000
kill_distance['walkDistance'] = pd.cut(kill_distance['walkDistance'], [0, 3000,

plt.figure(figsize=(8,8))
sns.countplot(kill_distance['walkDistance'].astype('str').sort_values())
plt.title("Walk Distance",fontsize=15)
plt.show()

```

executed in 16ms, finished 15:06:52 2019-04-30

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
<ipython-input-7-452940db8b0e> in <module>
      1 # bar plot for walk distance
----> 2 kill_distance = player_data.copy()
      3 kill_distance['walkDistance']/=1000
      4 kill_distance['walkDistance'] = pd.cut(kill_distance['walkDist
ance'], [0, 3000, 5000, 7000,9000,11000], labels=['0-3000_km', '3000-5
000_km', '5000-7000_km',
      5
'7000-9000km','9000km+'])

NameError: name 'player_data' is not defined
```

In [8]:

```
#Getting top 1% runners on the leaderboard
runners_df = player_data[player_data['walkDistance'] > player_data['walkDistanc
display(runners_df)

```

executed in 22ms, finished 15:06:52 2019-04-30

```
---------------------------------------------------------------------------
-----
NameError                                 Traceback (most recent call
 last)
<ipython-input-8-5116c298d7b9> in <module>
      1 #Getting top 1% runners on the leaderboard
----> 2 runners_df = player_data[player_data['walkDistance'] > player_
data['walkDistance'].quantile(0.99)].sort_values(by='longestKill', asc
ending=False)[['walkDistance','rank','accountid']]
      3 display(runners_df)

NameError: name 'player_data' is not defined
```

WalkDistance feature has very close correlation with the Rank in leaderboard. The number one in the leaderboard belongs to the runner category

The Runners in the leaderboard are

In [28]:

```
names = []
for accountid in runners_df['accountid']:
    names.append(leaderboard[leaderboard['accountid'] == accountid]['name'].val

runners_df['name'] = names
display(runners_df)
```

executed in 105ms, finished 16:48:31 2019-03-24

	walkDistance	rank	accountid	name
148	8592122.0	149	187396e882e54109be4d50b05bcb6340	ShuFuM
10	8588254.0	11	4b4f953ad152498b839fd2350776e9e8	CoolF1sh
33	8961436.0	34	807bbb2a661d40a8929c5280d9729581	DouYuTv-3722664
0	9171129.0	1	f072a307f36f401e93b1ddc9db0155c3	DouYu-33783
450	8745266.0	451	dec14d24b9e749f9abdea4b8988f38f4	Huya___13238962
30	8581076.0	31	3e20d003959144e189b40ccc043b8d31	BX_LiuBo
106	8959651.0	107	93be53e87264433f8b5b2fce852ca8c8	HonorFaith007
236	8587462.0	237	594ff7ba1a0f46dfb9d2ca23f976ad59	virginlover1
219	8592798.0	220	2cb129875cb94dd995826d626a4c81f5	1377348545

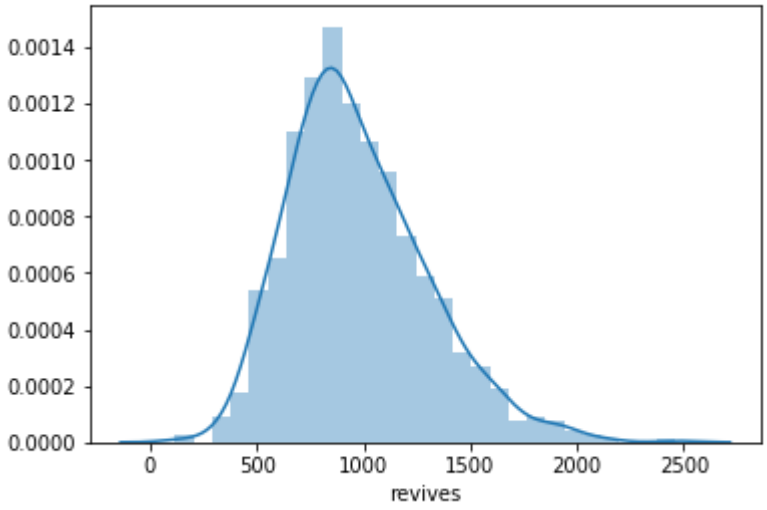
Field Medic

In [29]:

```
#distribution plot for revives
sns.distplot(player_data['revives'])
display("The average person on the leaderboard has revived  {:.2f} times, 99% o
```

executed in 417ms, finished 16:48:34 2019-03-24

```
'The average person on the leaderboard has revived  975.11 times, 99%
of people have revived  1913.07 times or less, while the highest numbe
r of revives is 2460. times'
```



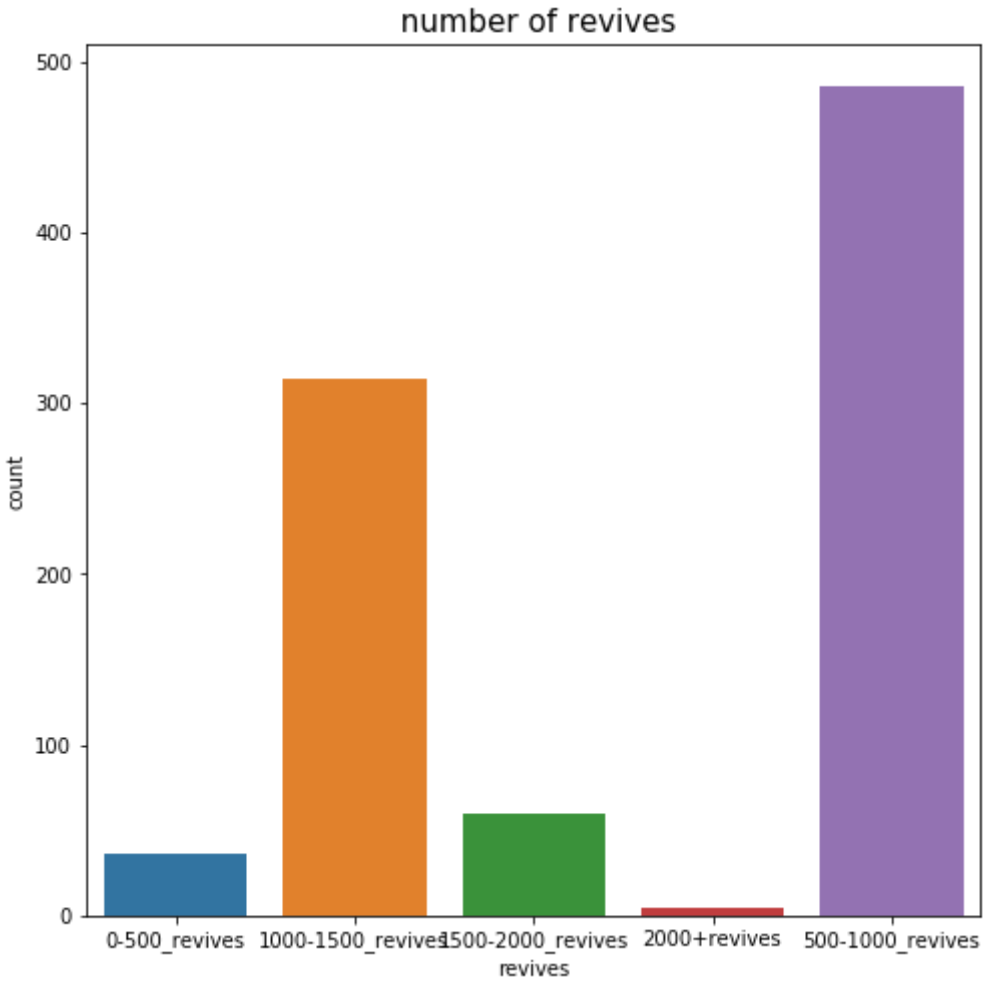
In [30]:

```
# bar plot for  revive numbers
medic_df = player_data.copy()
medic_df['revives'] = pd.cut(medic_df['revives'], [0, 500, 1000, 1500,2000,2500

plt.figure(figsize=(8,8))
sns.countplot(medic_df['revives'].astype('str').sort_values())
plt.title(" number of revives",fontsize=15)
plt.show()

```

executed in 286ms, finished 16:48:34 2019-03-24



In [31]:

```
#Getting top 1% medics on the leaderboard
medics_df = player_data[player_data['revives'] > player_data['revives'].quantil
display(medics_df)

```

executed in 26ms, finished 16:48:35 2019-03-24

	revives	rank	accountid
379	2460	380	a40b1c88f46e41fd961c8e6c04509b4c
155	2142	156	bc4730a155bd48ac893e8b2f5290acba
886	2106	887	07abebf3b5c641b5bfe492dcd336ec72
316	2027	317	48a9c9fe9ed7440b96845e91998f7c9c
183	1981	184	2e0140f6380b4180af7131deeb6440d5
11	1951	12	fc16e54c8e8e466ca9ab3d3890723fd4
17	1946	18	a1572c40a0e14a95b7923c3abedd0184
23	1924	24	c7ff66ef44a242a7a81eef1096b8eda1
377	1920	378	39e7fa76538a430b9a1dc4a762b9ca74

The revives feature has good correlation with the Rank in leaderboard. A good number of top players have good revive record

The Medics in the leaderboard are

In [32]:

```
names = []
for accountid in medics_df['accountid']:
    names.append(leaderboard[leaderboard['accountid'] == accountid]['name'].val

medics_df['name'] = names
display(medics_df)
```

executed in 100ms, finished 16:48:42 2019-03-24

	revives	rank	accountid	name
379	2460	380	a40b1c88f46e41fd961c8e6c04509b4c	kk0oooo
155	2142	156	bc4730a155bd48ac893e8b2f5290acba	0_0yangxuan
886	2106	887	07abebf3b5c641b5bfe492dcd336ec72	Queen__Coco
316	2027	317	48a9c9fe9ed7440b96845e91998f7c9c	DouYuTv_5631012
183	1981	184	2e0140f6380b4180af7131deeb6440d5	JustGoToSlee9
11	1951	12	fc16e54c8e8e466ca9ab3d3890723fd4	ACE-cGeggggg
17	1946	18	a1572c40a0e14a95b7923c3abedd0184	Faith-KingH
23	1924	24	c7ff66ef44a242a7a81eef1096b8eda1	YishengOwO
377	1920	378	39e7fa76538a430b9a1dc4a762b9ca74	Jviv

In [33]:

```
1 player_data.columns
```

executed in 6ms, finished 16:48:42 2019-03-24

Out[33]:

```
Index(['assists', 'bestRankPoint', 'boosts', 'dBNOs', 'dailyKills',
       'dailyWins', 'damageDealt', 'days', 'headshotKills', 'heals',
'kills',
       'longestKill', 'longestTimeSurvived', 'losses', 'maxKillStreak
s',
       'mostSurvivalTime', 'rankPoints', 'revives', 'rideDistance',
       'roadKills', 'roundMostKills', 'roundsPlayed', 'suicides',
       'swimDistance', 'teamKills', 'timeSurvived', 'top10s',
       'vehicleDestroys', 'walkDistance', 'weaponsAcquired', 'weeklyKi
lls',
       'weeklyWins', 'wins', 'accountid', 'rank'],
      dtype='object')
```

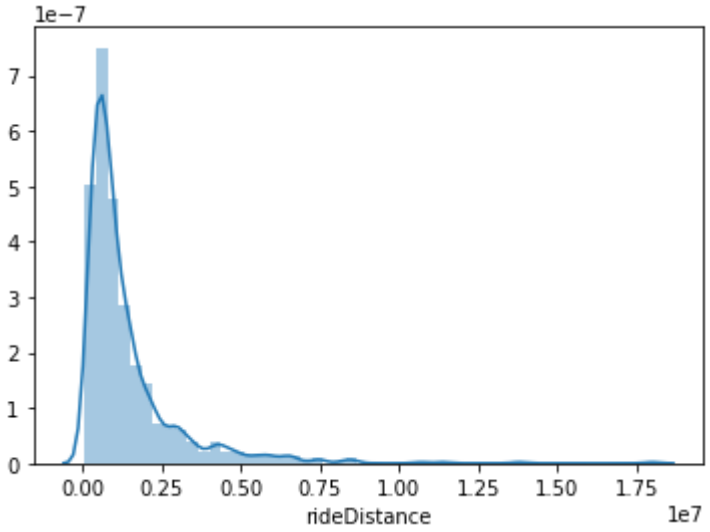
Drivers

In [34]:

```
1 #distribution plot for rideDistance
2 sns.distplot(player_data['rideDistance'])
3
4 display("The average person on the leaderboard has driven {:.2f} kms, 99% of pe
5
```

executed in 483ms, finished 16:48:45 2019-03-24

```
'The average person on the leaderboard has driven 1413.64 kms, 99% of
people have driven  7479.62 kms or less, while the highest number of d
riven distance is 17989.726. kms'
```



In [35]:

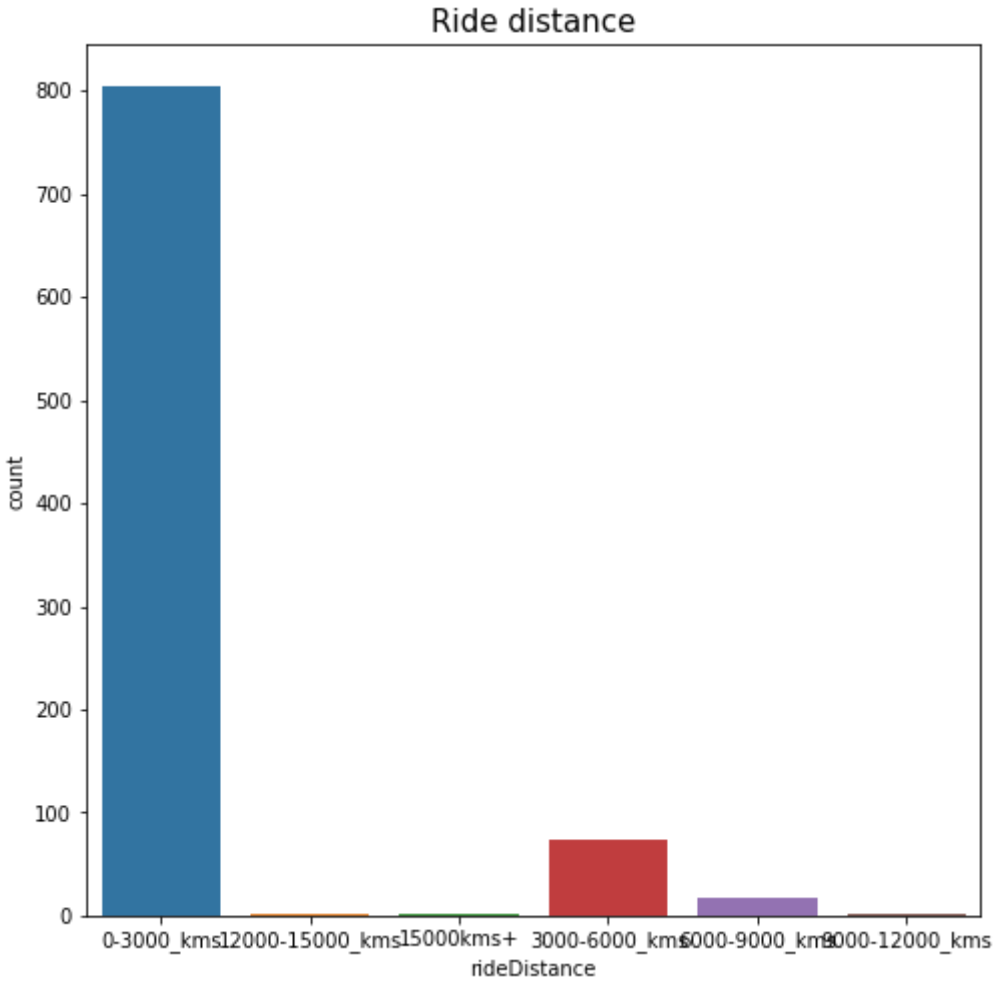
```
# bar plot for  revive numbers
ride_distance = player_data.copy()
ride_distance['rideDistance']/=1000

ride_distance['rideDistance'] = pd.cut(ride_distance['rideDistance'], [0, 3000,

plt.figure(figsize=(8,8))
sns.countplot(ride_distance['rideDistance'].astype('str').sort_values())
plt.title("Ride distance",fontsize=15)
plt.show()

```

executed in 323ms, finished 16:48:45 2019-03-24



In [36]:

```
#Getting top 1% drivers on the leaderboard
drivers_df = player_data[player_data['rideDistance'] > player_data['rideDistanc
display(drivers_df)

```

executed in 30ms, finished 16:48:46 2019-03-24

	rideDistance	rank	accountid
32	17989726.0	33	953038827af449809706b66f32a91551
173	13785254.0	174	408a9cdad4a24e03b4a1280ede0df4cc
189	11366042.0	190	44d795bac87d4d17b76497642fe2ece2
451	10687554.0	452	8373768ee09d40d089e3d72bef60884f
629	8621123.0	630	041fa64194df4219a1e4505f6c21f368
583	8475949.0	584	6beeb77095834216812d9b2f184fcd8c
568	8391898.0	569	d01821ab1d434cafaa176691e58c04b8
114	8207590.5	115	48545bf563ad427092e6561852b556a8
813	7545186.5	814	0a6a9deb59c141fa89da41e4864329b5

The feature rideDistance has no correlation with the Rank in leaderboard. Top riders are distributed all over the spectrum

The Drivers in the leaderboard are

In [37]:

```
names = []
for accountid in drivers_df['accountid']:
    names.append(leaderboard[leaderboard['accountid'] == accountid]['name'].val

drivers_df['name'] = names
display(drivers_df)
```

executed in 108ms, finished 16:48:47 2019-03-24

	rideDistance	rank	accountid	name
32	17989726.0	33	953038827af449809706b66f32a91551	whdals256
173	13785254.0	174	408a9cdad4a24e03b4a1280ede0df4cc	Twitch_TheJoonTV
189	11366042.0	190	44d795bac87d4d17b76497642fe2ece2	TTS_996
451	10687554.0	452	8373768ee09d40d089e3d72bef60884f	Dangersn
629	8621123.0	630	041fa64194df4219a1e4505f6c21f368	untitle1
583	8475949.0	584	6beeb77095834216812d9b2f184fcd8c	JustBe_Patient
568	8391898.0	569	d01821ab1d434cafaa176691e58c04b8	lp__link
114	8207590.5	115	48545bf563ad427092e6561852b556a8	4SMR_panpan
813	7545186.5	814	0a6a9deb59c141fa89da41e4864329b5	OMG-Messi

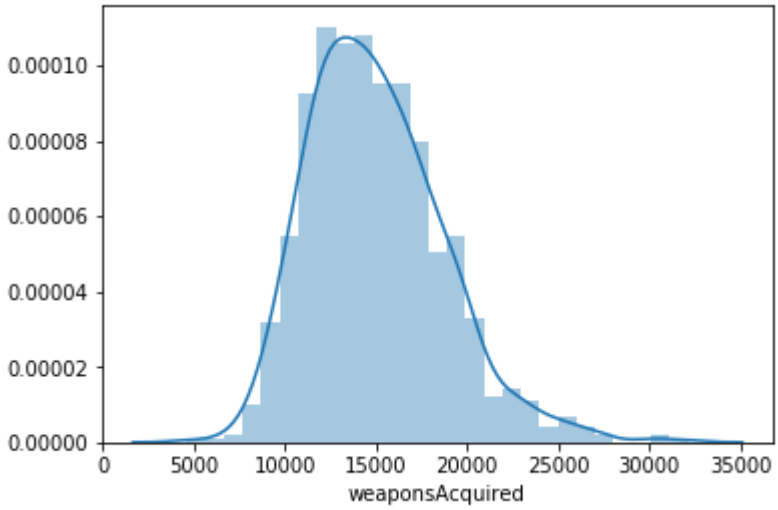
Weapon Masters

In [38]:

```
1 #distribution plot for weaponsAcquired
2 sns.distplot(player_data['weaponsAcquired'])
3
4 display("The average person on the leaderboard has acquired {:.0f} weapons, 99%
5
```

executed in 449ms, finished 16:48:49 2019-03-24

```
'The average person on the leaderboard has acquired 15007 weapons, 99%
of people have acquired 25773 weapons or less, while the highest numbe
r of weapons acquired by a person is 32121.'
```

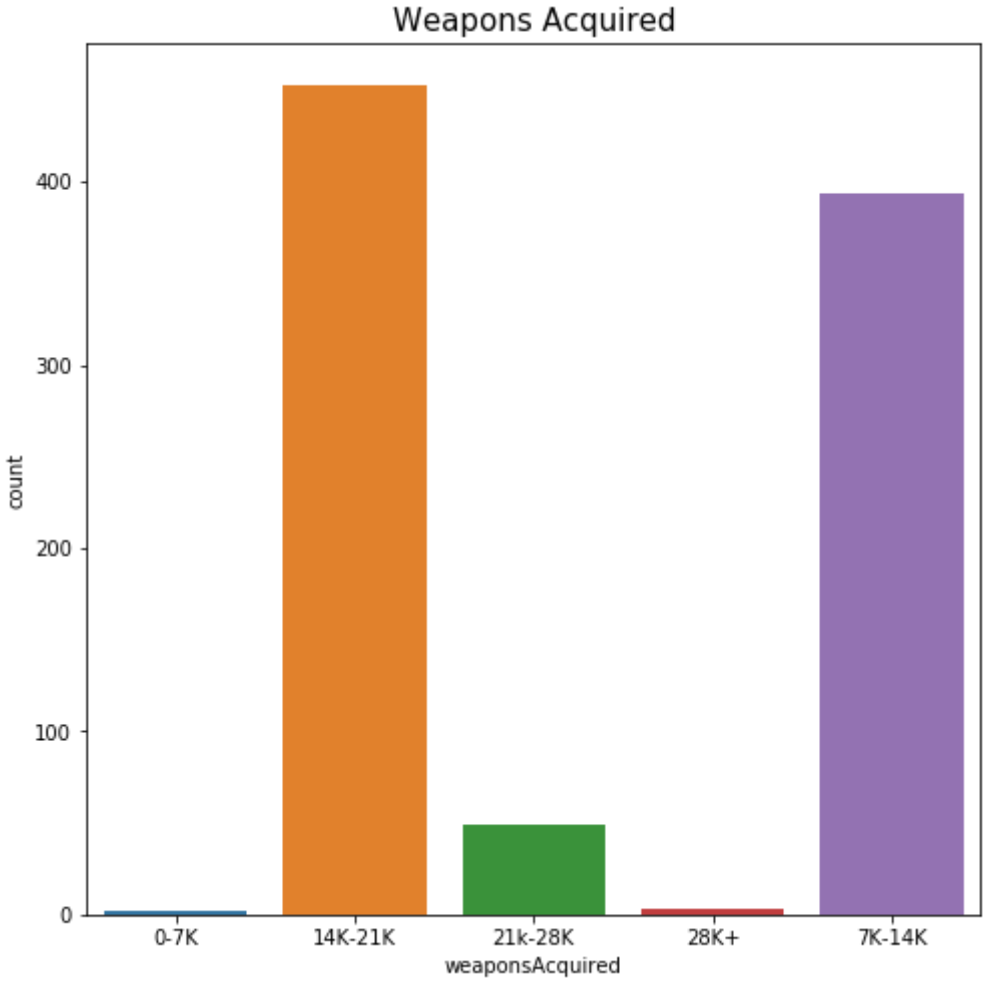


In [39]:

```
# bar plot for  revive numbers
weapons_acquired = player_data.copy()
weapons_acquired['weaponsAcquired'] = pd.cut(weapons_acquired['weaponsAcquired'

plt.figure(figsize=(8,8))
sns.countplot(weapons_acquired['weaponsAcquired'].astype('str').sort_values())
plt.title("Weapons Acquired",fontsize=15)
plt.show()
```

executed in 318ms, finished 16:48:49 2019-03-24



In [40]:

```
#Getting top 1% weapon masters on the leaderboard
weapon_masters = player_data[player_data['weaponsAcquired'] > player_data['weap
display(weapon_masters)
```

executed in 35ms, finished 16:48:49 2019-03-24

	weaponsAcquired	rank	accountid
158	32121	159	8395775ccdb940c498320dfa10ff2e4e
45	30468	46	a9ceccdc9377491ba85bd3bbb57a7a2e
148	30097	149	187396e882e54109be4d50b05bcb6340
28	27657	29	4c17a53188004ef596c0e788c53e7164
147	27164	148	0745e0b43bc04a8da572962ba8c5eb30
764	26940	765	2cabb5aabf65462c8e7f09c4dfbf4ef4
219	26839	220	2cb129875cb94dd995826d626a4c81f5
30	26364	31	3e20d003959144e189b40ccc043b8d31
90	26049	91	6846d7f76a344414850dcf93c29b8afc

The feature weaponsAcquired has good correlation with the Rank in leaderboard. Top Weapon Masters lie in the first quantile in the Leaderboard Rank

The Weapon Masters in the leaderboard are

In [41]:

```
names = []
for accountid in weapon_masters['accountid']:
    names.append(leaderboard[leaderboard['accountid'] == accountid]['name'].val

weapon_masters['name'] = names
display(weapon_masters)
```

executed in 102ms, finished 16:48:51 2019-03-24

	weaponsAcquired	rank	accountid	name
158	32121	159	8395775ccdb940c498320dfa10ff2e4e	Huya_17105607
45	30468	46	a9ceccdc9377491ba85bd3bbb57a7a2e	M24_killer_A
148	30097	149	187396e882e54109be4d50b05bcb6340	ShuFuM
28	27657	29	4c17a53188004ef596c0e788c53e7164	LeShanAwenOvO
147	27164	148	0745e0b43bc04a8da572962ba8c5eb30	QY-ZGY
764	26940	765	2cabb5aabf65462c8e7f09c4dfbf4ef4	FuxkBitch_
219	26839	220	2cb129875cb94dd995826d626a4c81f5	1377348545
30	26364	31	3e20d003959144e189b40ccc043b8d31	BX_LiuBo
90	26049	91	6846d7f76a344414850dcf93c29b8afc	TAOZHIYAO131

In [42]:

```
1 player_data.columns
```

executed in 10ms, finished 16:48:51 2019-03-24

Out[42]:

```
Index(['assists', 'bestRankPoint', 'boosts', 'dBNOs', 'dailyKills',
       'dailyWins', 'damageDealt', 'days', 'headshotKills', 'heals',
'kills',
       'longestKill', 'longestTimeSurvived', 'losses', 'maxKillStreak
s',
       'mostSurvivalTime', 'rankPoints', 'revives', 'rideDistance',
       'roadKills', 'roundMostKills', 'roundsPlayed', 'suicides',
       'swimDistance', 'teamKills', 'timeSurvived', 'top10s',
       'vehicleDestroys', 'walkDistance', 'weaponsAcquired', 'weeklyKi
lls',
       'weeklyWins', 'wins', 'accountid', 'rank'],
      dtype='object')
```

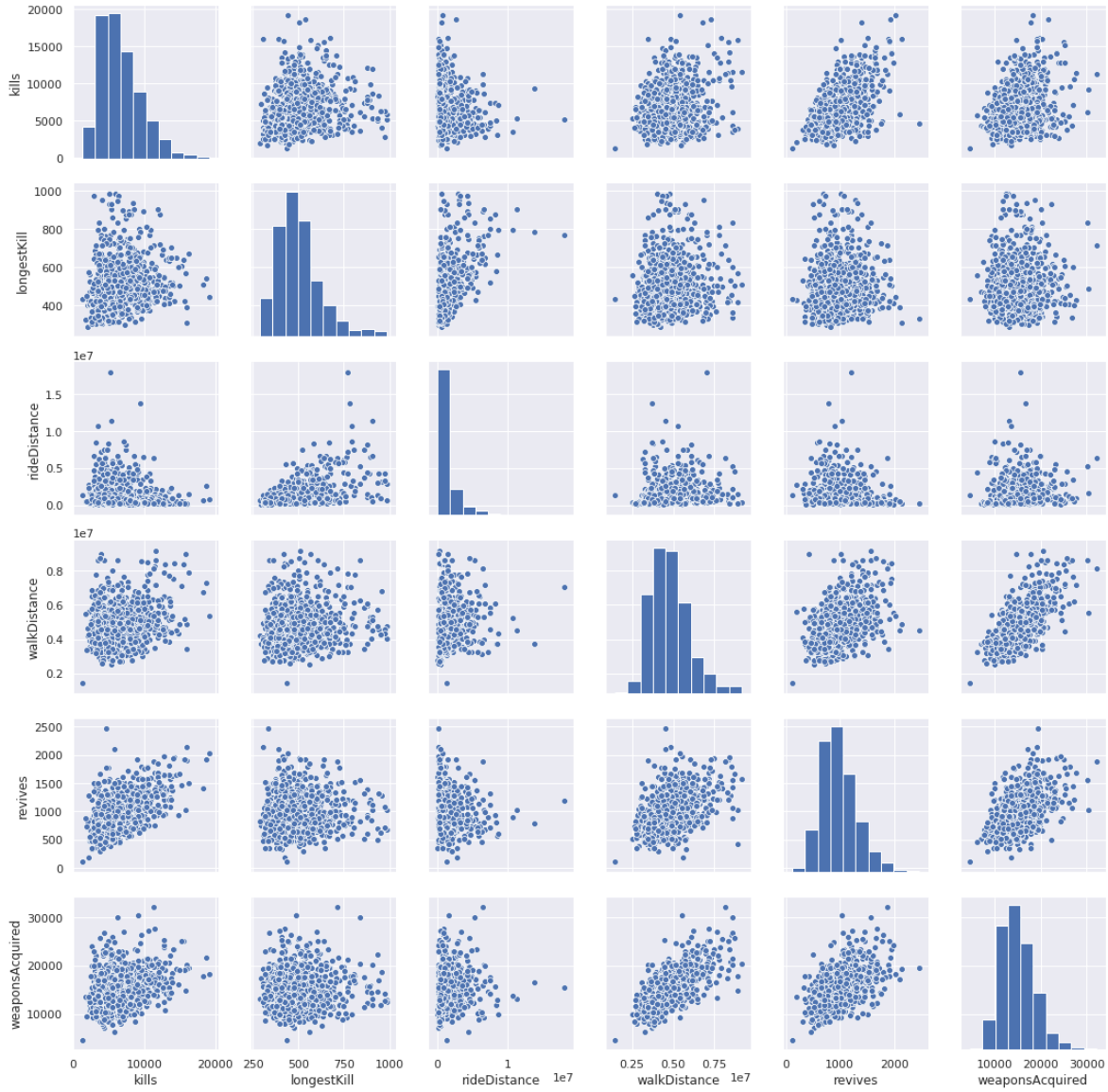
Pair plot of features selected for Exploraory Data Analysis

In [64]:

```
sns.set()
cols = ['kills','longestKill', 'rideDistance','walkDistance', 'revives', 'weapo
sns.pairplot(player_data[cols], size = 2.5)
plt.show()
```

executed in 8.27s, finished 17:13:46 2019-03-24

```
/home/keerthi/anaconda3/lib/python3.6/site-packages/seaborn/axisgrid.p
y:2065: UserWarning: The `size` parameter has been renamed to `height
`; pleaes update your code.
  warnings.warn(msg, UserWarning)
```



Conclusion

By the above Exploratary Data Analysis we can see that the rank is indicative of players' skills, which is factored by multiple features. We are trying to find what feature mostly correlates to rank. The analysis shows that Terminators, Medics, Runners and Weapon Masters are most likely to be in the top league of the leaderboard. This analysis is backed by the correlation plot and the top 1% players in each category and comparision with the leaderboard top players. We can conclude that, to be in top of the leadeboard, a player should be anyone of good Terminator, Runner, medic, Driver and Weapon Master.

Go to Top

In [ ]:

```
1
```