Starcraft 2 Online Ranked Matchmaking Rating Analysis and Visualization

What? Why?

One day a few months ago, someone posed a question regarding Matchmaking Rating (MMR) for the game Starcraft 2 (SC2). I can't remember the exact question anymore but I thought I'd give answering it a go and, well, I kind of ended up going down the rabbit hole a bit.

If you are not familiar with what MMR is, it's a system for rating the skills of players such that players of similar skill levels get matched together. It's a system that's most commonly known about from in chess in the form of the Elo rating system, but has been used elsewhere for similar purposes, most commonly in video games but also in other services that offer matchmaking services like Tinder or Bumble. Matchmaking data tends to follow a normal distribution, which we'll see play out in the data as we examine it.

Anywho, let's get started. We're mostly just going to be looking at descriptive statistics along with visualizing data so we don't need too many fancy libraries; mostly just the basics.

First things first, we need to import pandas, seaborn, and matplotlib. If you've done pretty much anything with data in python you'll likely have tinkered with at least the first and last of these modules. If you are not familiar with seaborn, it's basically matplotlib but fancier and easier to use.

In [ ]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Cool. Now that the modules we need are ready to go, we can read in the Starcraft 2 ladder data that has what we want. The data was scraped from the website `rankedftw.com`, which itself gets the data from Blizzard themselves. I could have went straight to Blizzard's API, but I wanted practice with web scraping so I went this route. If you want to test out my web scraper itself, it's included in this repository and there are instructions included on how to use it. A bit of data cleaning is required though to get things working perfectly so I've also included instructions for that.

In [ ]:
```
MMR_data = pd.read_csv("data/SC2_ladder_data.csv")
MMR_data.shape[0]
```

Out[ ]:
```
421331
```

Yay. That's done. That number directly above is the total number of rows stored in our DataFrame of SC2 accounts. So that means there are nearly half a million active accounts still in SC2. Not bad for a decade-plus old game. Do keep in mind some people have multiple accounts, but that's not likely to be too many folks these days.

A Couple Preliminary Bits of Information

Okay, before we go any further, there needs to be a bit of an explanation of what we're going to be looking at. In StarCraft and StarCraft 2 multiplayer, the player can choose from three different races. The races are called Protoss, Terran, and Zerg. The player also can choose to play "Random", which means the game will randomly choose a race for the player. The game is designed such that each race is different mechanically but is created such that they all have roughly an equal chance to win against the other races.

Secondly, we'll be looking at MMR breakdown based on region. In StarCraft 2, there are currently four regions where players are stratified. The two largest are the American region, which is made up of North America and South America, and the European region, which encompasses The UK, all of the EU countries, the eastern European countries, and Russia. The third server is the Chinese server, which contains players from China and various parts of Asia. Finally, the last and smallest region is the Korean region, which despite its diminutive size is regarded as the highest skill-level region. It is mostly populated with people from Korea, Asia, and a small amount of players from the western half of North America.

Now That That's Over With, Data Time!

When I first started doing this analysis, I initially only looked at the Americas region before expanding out to the other regions. To make what we're looking at easier to understand, I'm going to stick with one region so you can get a feel for what the data looks like. Then later on we can compare the other regions.

First, we will be loooking at each individual race. We'll take a look at the basic information for each before finally comparing them.

First up is Protoss. We'll use pandas' handy dandy query function to filter our data as we would like. Here's what the data currently looks like.

In [ ]:
```
protossAM_data = MMR_data.query('region == "AM" and race == "protoss"')
protossAM_data.head()
```

Out[ ]:

	rank	tier	name	mmr	points	wins	losses	played	winrate	age	ra
0	1	1	Memestermind	6685	934	20	3	23	87.0%	20m	proto
1	2	1	[N0SCAM]NoWCSForU	6673	668	15	0	15	100.0%	38m	proto
2	3	1	PartinG	6643	3381	107	19	126	84.9%	2m	proto
3	4	1	lllllllllll	6635	1056	37	7	44	84.1%	9m	proto
4	5	1	Hana	6604	1839	229	65	294	77.9%	10m	proto

In [ ]:
```
protossAM_data.shape[0]
```

Out[ ]:
```
46089
```

So currently there are about 46000 people on the Americas region's servers playing Protoss. Neat!

Now we'll look at some of the usual basic descriptive statistics to get a feel for how the MMR data looks.

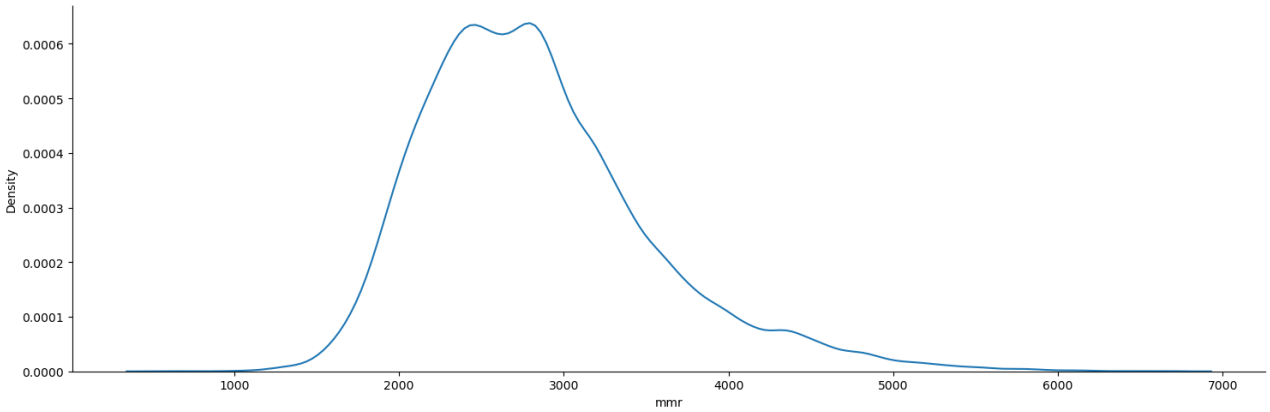
In [ ]:
```
print("Protoss MMR on the Americas server:\nmean: {}\nmedian: {}\nstandard d
```

```
Protoss MMR on the Americas server:
mean: 2833.2559612922823
median: 2739.0
standard deviation:700.249565906824
```

The average MMR for a Protoss player on the AM server is 2833 and half of all players are at or below an MMR of 2739. Looking at the density plot below we see that this all seems reasonable. A big chunk of players have an MMR around 2800 plus or minus about 500-700, which lines up with our standard deviation found.

In [ ]:
```
sns.displot(data=protossAM_data, x='mmr', kind='kde', height=5, aspect=3)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1786469e0>
```



Repeating the same steps for AM Zerg players gets us the following information:

In [ ]:
```
zergAM_data = MMR_data.query('region == "AM" and race == "zerg"')
zergAM_data.head()
```

Out[ ]:

	rank	tier	name	mmr	points	wins	losses	played	winrate	age	rac
46089	1	1	[BJWD]MaBaoGuo	6601	1104	50	26	76	65.8%	34m	zer
46090	2	1	Scarlett	6584	48	2	0	2	100.0%	42m	zer
46091	3	1	Denver	6348	0	0	1	1	0.0%	31m	zer
46092	4	1	[PSISTM]Namshar	6317	1848	113	41	154	73.4%	29m	zer
46093	5	1	[ArGNA]SortOf	6298	48	1	5	6	16.7%	24m	zer

In [ ]:
```
zergAM_data.shape[0]
```

Out[ ]:
```
40376
```

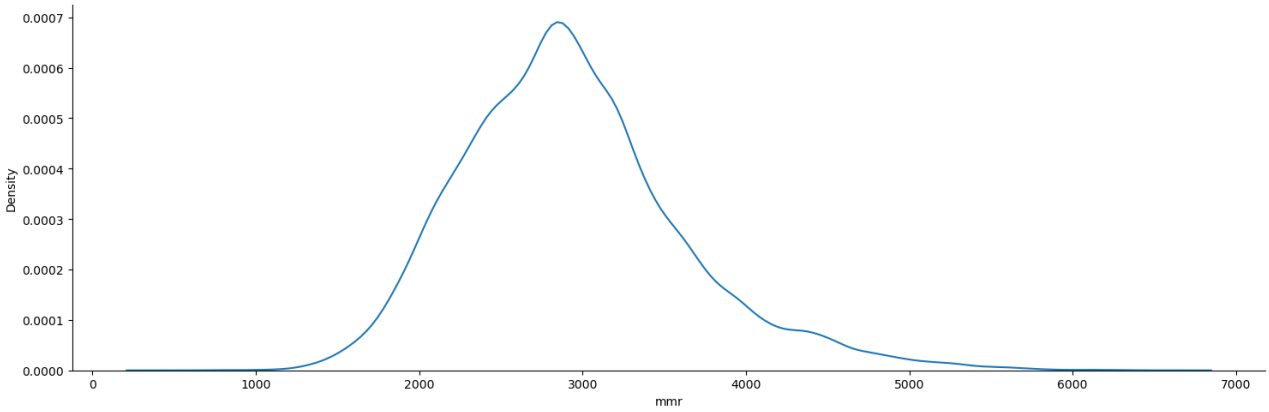
Overall, there are around 6000 fewer Zerg players on the AM server. The average MMR of these players is 2931 and the median is slightly higher at 2872. This suggests that overall, most Zerg players have slightly higher MMR than their Protoss counterparts.

In [ ]:
```
print("Zerg MMR on the Americas server:\nmean: {}\nmedian: {}\nstandard devi
```

```
Zerg MMR on the Americas server:
mean: 2931.5853477313253
median: 2872.0
standard deviation:683.7423985781613
```

In [ ]:
```
sns.displot(data=zergAM_data, x='mmr', kind='kde', height=5, aspect=3)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x28b6a7460>
```



Looking at the plot of all AM zerg players, it's noticeable how pointy the graph looks. This suggests that there's a huge concentration of Zerg players at about 3000 MMR plus or minus about 200. Both the left and especially the right side of the curve seem steeper than on the Protoss graph, suggesting there could be a bigger concentration of players within a smaller MMR range. This conclusion agrees with the standard deviation being smaller than for Protoss.

Finally, we'll repeat the test for Terran.

In [ ]:
```
terranAM_data = MMR_data.query('region == "AM" and race == "terran"')
terranAM_data.head()
```

Out[ ]:

	rank	tier	name	mmr	points	wins	losses	played	winrate	age	race
86465	1	1	[Ex0n]SpeCial	6345	1883	72	29	101	71.3%	7m	terran
86466	2	1	Sonagi	6258	0	0	2	2	0.0%	31m	terran
86467	3	1	lllllllllll	6232	1280	29	6	35	82.9%	26m	terran
86468	4	1	[αX]MaSa	6196	240	6	2	8	75.0%	7m	terran
86469	5	1	Spear	6186	0	0	1	1	0.0%	7m	terran

In [ ]:
```
terranAM_data.shape[0]
```

Out[ ]:
```
50579
```

As has almost always been the case with SC2, Terran is the most popular race to play online. The Terran portion of the singleplayer campaign is available for free so a lot of players go from the free singleplayer into multiplayer using Terran. Because of this, there also is often a bigger proportion of lower MMR players playing Terran since they are just beginning playing the game. The data demonstrates this:

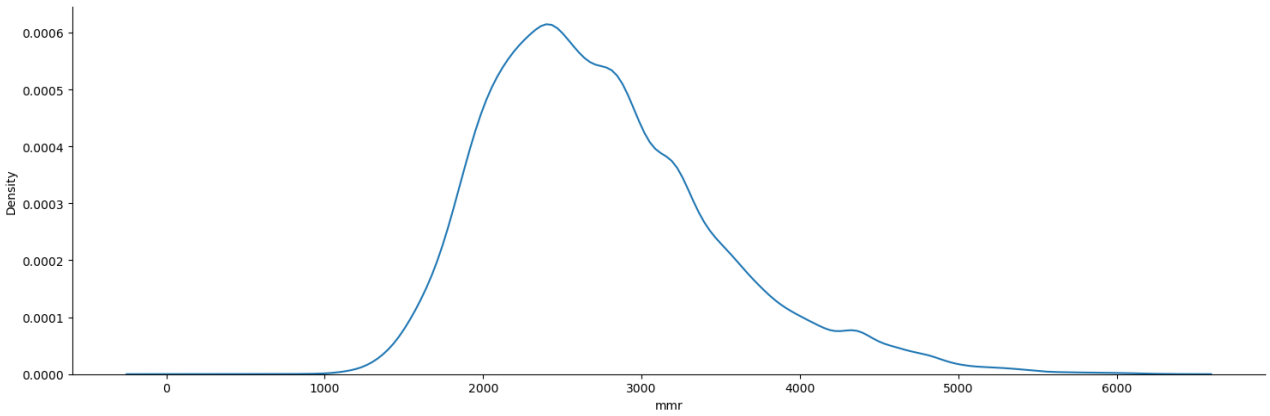
In [ ]:
```
print("Terran MMR on the Americas server:\nmean: {}\nmedian: {}\nstandard de
```

```
Terran MMR on the Americas server:
mean: 2744.95745269776
median: 2635.0
standard deviation:729.4974736679407
```

Terran MMR is on average almost 100 points lower than Protoss and 200 points lower than Zerg. The plot of the Terran data below does a great job demonstrating the neat impact there being a lot of beginner Terran player. The graph skews leftward and is much less symmetrical compared to the other races. A disproportionately large chunk of Terran players sit on the lower end of the MMR spectrum but we can also see that past that huge group of players, there are other spikes in the plot of players at certain MMR levels.

In [ ]:
```
sns.displot(data=terranAM_data, x='mmr', kind='kde', height=5, aspect=3)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x28b72fc70>
```



In [ ]:
```
randomAM_data = MMR_data.query('region == "AM" and race == "random"')
randomAM_data.head()
```

Out[ ]:

	rank	tier	name	mmr	points	wins	losses	played	winrate	age	
137044	1	1	[xGenGx]PiLiPiLi	6054	0	1	1	2	50.0%	28m	r
137045	2	1	[MFSC2]pezzaperry	5911	48	2	1	3	66.7%	7m	r
137046	3	1	[Ex0n]Burninator	5670	0	1	0	1	100.0%	38m	r
137047	4	1	[LRKT]Guitarcheese	5636	0	1	0	1	100.0%	46m	r
137048	5	1	[R1seUP]TheBlazer	5554	48	1	2	3	33.3%	38m	r

In [ ]:
```
randomAM_data.shape[0]
```

Out[ ]:
```
13736
```

Finally, we have Random. Not a lot of people play Random (which is where the game randomly picks a race for you), and those that do tend to be good at one race and weak at the other two, so we end up with MMR averaging out to be around the same range as the other races. Many Random players will overperform compared to MMR with their most comfortable race then underperform with the others, which means their MMR will often stabilize at about average. This also has the effect of making it so there isn't as big of a range of MMRs, which is how we end up with such a low standard deviation.

In [ ]:
```
print("Random MMR on the Americas server:\nmean: {}\nmedian: {}\nstandard de
```

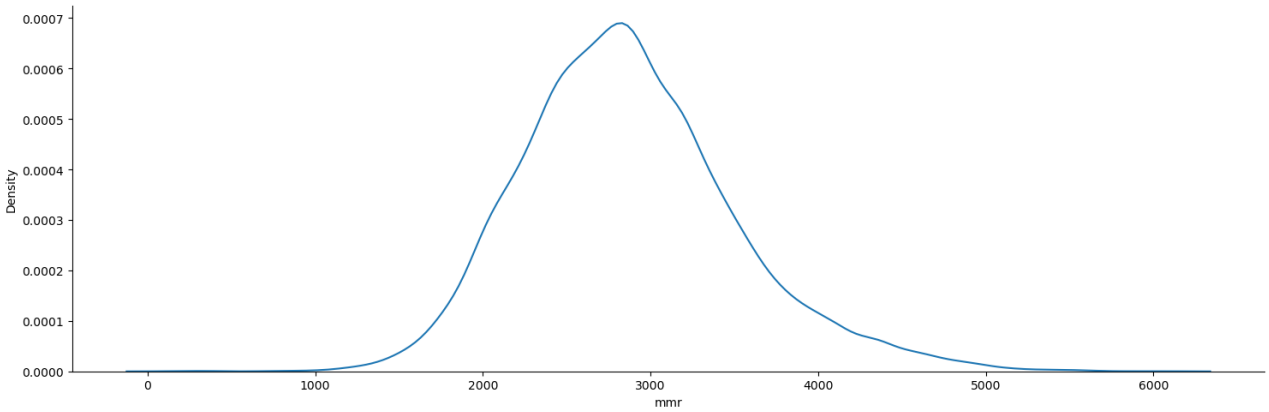
```
Random MMR on the Americas server:
mean: 2870.2438846825858
median: 2822.0
standard deviation:639.3204071216842
```

In [ ]:
```
randomAM_data['mmr'].mean()
```

Out[ ]:
```
2870.2438846825858
```

In [ ]:
```
sns.displot(data=randomAM_data, x='mmr', kind='kde', height=5, aspect=3)
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x298dc9450>
```



We can see that the Random density plot is very pointy and symmetrical, showing that MMR is fairly evenly distributed around the mean.

Combining The Data

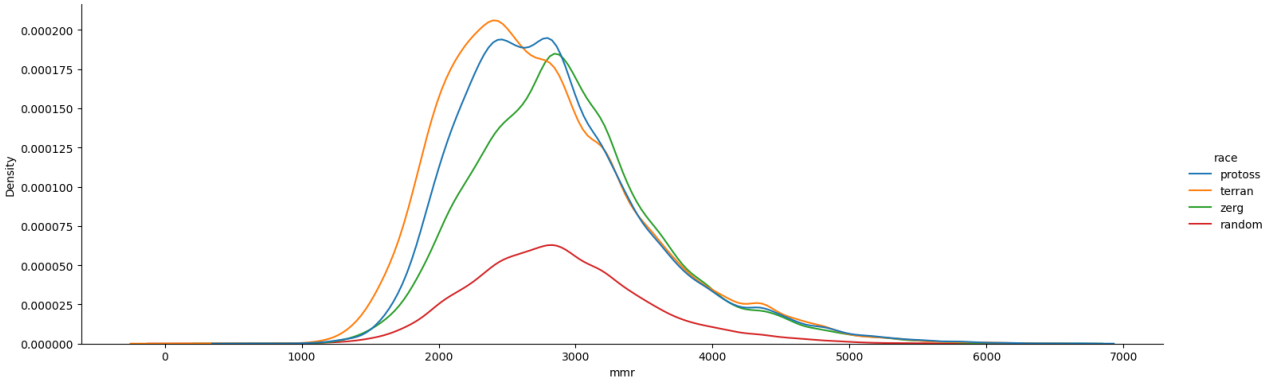
Now we will combine all of these results so we can look at them side by side in an easier manner.

In [ ]:
```
combined_AM = pd.concat([protossAM_data, terranAM_data, zergAM_data, randomA
combined_AM.shape[0]
```

Out[ ]:
```
150780
```

In [ ]:
```
sns.displot(data=combined_AM, x='mmr',  hue='race', kind='kde', height=5, as
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x298cff9d0>
```



This combined density graph gives us a handy means of looking at our previous analysis. For instance, green line agrees with our conclusion that Zerg MMR is overall higher than the other races on average while the Terran curve skews left. We can also see that past around 3700 MMR, the three races more or less are equally distributed in terms of representation. Their density lines start to overlap, suggesting the number of players in those MMR ranges are roughly the same for each race.

We can also take a look at the overall ladder situation now for AM. The average AM player has an MMR of 2833 and standard deviation is roughly the same as with all of the races individually.

In [ ]:
```
print("Overall MMR on the Americas server:\nmean: {}\nmedian: {}\nstandard d
```

```
Overall MMR on the Americas server:
mean: 2833.3366295264623
median: 2760.0
standard deviation:704.374637273895
```

Looking at how player activity influences MMR

Anyone who has played SC2 for a long time will tell you that when they look at their ladder leaderboard, often a huge chunk of accounts only play one or two ranked games in a season, just enough to keep their ladder ranking somewhat active from season to season. The result directly below shows us that on average, players play 55 games per season.

In [ ]:
```
combined_AM['played'].mean()
```

Out[ ]:
```
55.30782597161427
```

55 sounds like a reasonable number of games per player, but this is where we have to remember that means can be skewed, and in this case the mean is massively skewed.

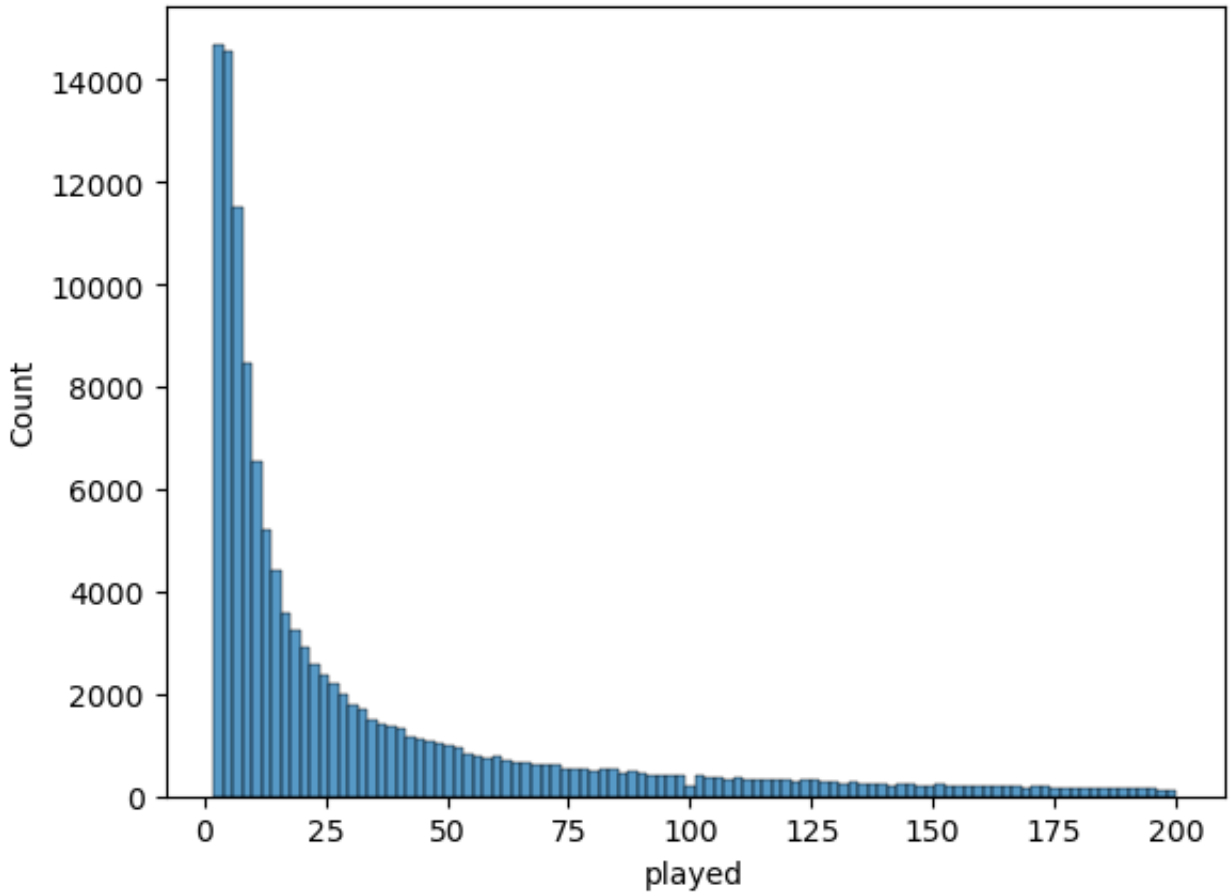
In [ ]:
```
combined_AM['played'].median()
```

Out[ ]:
```
13.0
```

The median here demonstrates just how heavily the mean is skewed. 50% of SC2 players play 13 or fewer games per season. A histrogram of the number of games played shows us a neat result:

In [ ]:
```
sns.histplot(data=combined_AM.query('played <= 200 and played > 1'), x='play
```

Out[ ]:
```
<AxesSubplot:xlabel='played', ylabel='Count'>
```



Our histogram shows us that a colossal number of players play fewer than 25 games. Looking further, we can see that over 20% of players, or ~35500 only play 5 games at the most per season.

In [ ]:
```
combined_AM.query('played < 5').shape[0]
```

Out[ ]:
```
35567
```

So how does the average number of games played get to 55? As it turns out, there are some people who play, let's say, a lot of StarCraft. Over 10000 people play a couple hundred games per season. However, they are not the 1% of SC2. The top 1% of players play roughly 580 games per season. There is a dedicated group that plays thousands of games per season. In this season, there were 310 people who played over 1000 games, which is a lot of StarCraft. One person though, played 7773 games, which is several times what most people play in total across all seasons. Now we can see how the mean got boosted so much compared to the median.

In [ ]:
```
combined_AM.query('played >= 200').shape[0]
```

Out[ ]:
```
10666
```

In [ ]:
```
# finding the SC2 1%
combined_AM.query('played >= 580').shape[0]
```

Out[ ]:
```
1545
```

In [ ]:
```
# The most hardcore people
combined_AM.query('played >= 1000').shape[0]
```

Out[ ]:
```
310
```

In [ ]:
```
# finding the most games played
combined_AM.query('played >= 7000')['played'].max()
```

Out[ ]:
```
7773
```

The question now is, what happens if we take away the barely active players. How does it influence MMR on the SC2 ladder?

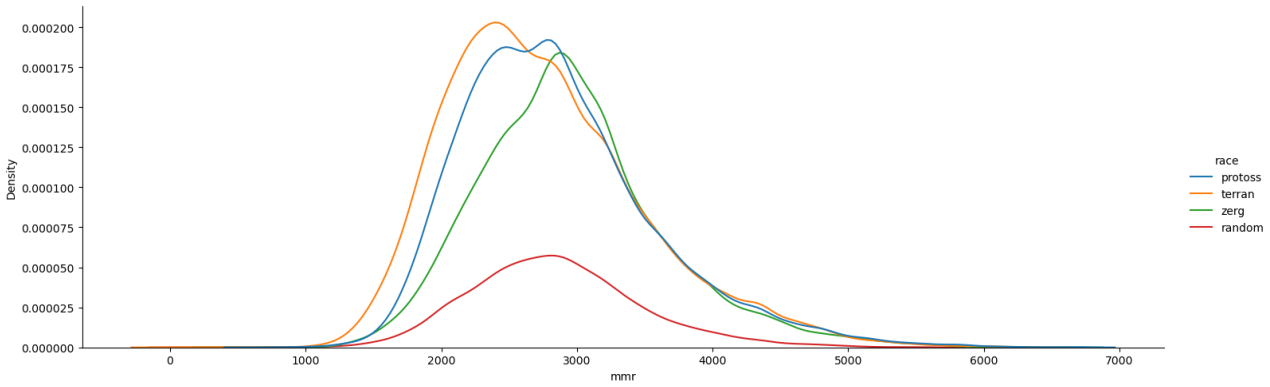
In [ ]:
```
combined_AM.query('played >= 10').shape[0]
```

Out[ ]:
```
85902
```

If we get rid of people who play under 10 games, we end up losing a bit less than half of our dataset.

In [ ]:
```
sns.displot(data=combined_AM.query('played >= 10'), x='mmr',  hue='race', ki
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x28b744fa0>
```



The combined density plot filtering out people who played fewer than 10 games looks fairly similar to the overall plot. Including more dedicated players only does increase the overall mean a bit, but not nearly as much as one would expect. The mean only increased by 17 points, and the median 18. For each race, the mean and median also are essentially the same. The race distributions also stay roughly the same, with there being more Terran players and fewer Zergs, and each race loses a similar proportion of players. The number of games played on average increases dramatically to 93 from 55, which makes sense since the less active accounts made up a big portion of that 55 average.

In [ ]:
```
combined_AM.query('played >= 10')['mmr'].mean()
```

Out[ ]:
```
2850.84547507625
```

In [ ]:
```
combined_AM.query('played >= 10')['mmr'].median()
```

Out[ ]:
```
2778.0
```

In [ ]:
```
print('mean:\nprotoss:{}\nterran: {}\nzerg: {}\nrandom:{}'.format(
    protossAM_data.query('played >= 10')['mmr'].mean(),
    terranAM_data.query('played >= 10')['mmr'].mean(),
    zergAM_data.query('played >= 10')['mmr'].mean(),
    randomAM_data.query('played >= 10')['mmr'].mean()))
```

```
mean:
protoss:2862.6679554070793
terran: 2759.2762558629597
zerg: 2958.1387587304393
random:2844.194263544409
```

In [ ]:
```
print('median:\nprotoss:{}\nterran: {}\nzerg: {}\nrandom:{}'.format(
    protossAM_data.query('played >= 10')['mmr'].median(),
    terranAM_data.query('played >= 10')['mmr'].median(),
    zergAM_data.query('played >= 10')['mmr'].median(),
    randomAM_data.query('played >= 10')['mmr'].median()))
```

```
median:
protoss:2769.0
terran: 2650.5
zerg: 2899.0
random:2803.0
```

In [ ]:
```
print('protoss:{}\nterran: {}\nzerg: {}\nrandom:{}'.format(
    protossAM_data.query('played >= 10')['mmr'].shape[0],
    terranAM_data.query('played >= 10')['mmr'].shape[0],
    zergAM_data.query('played >= 10')['mmr'].shape[0],
    randomAM_data.query('played >= 10')['mmr'].shape[0]))
```

```
protoss:26641
terran: 29422
zerg: 22622
random:7217
```

In [ ]:
```
combined_AM.query('played >= 10')['played'].mean()
```

Out[ ]:
```
93.99606528369537
```

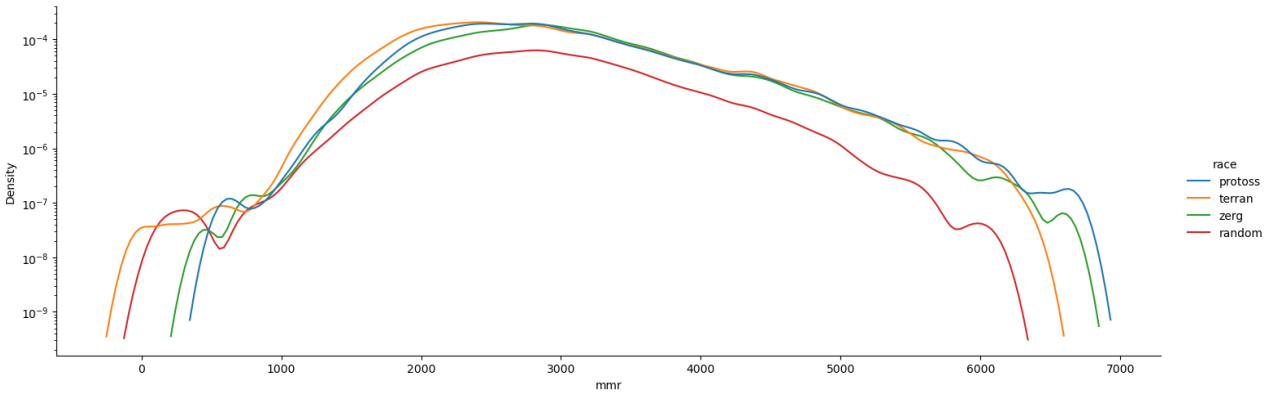
In [ ]:
```
print('protoss:{}\nterran: {}\nzerg: {}\nrandom:{}'.format(
    protossAM_data.query('played >= 10')['mmr'].shape[0]/protossAM_data['mmr
    terranAM_data.query('played >= 10')['mmr'].shape[0]/terranAM_data['mmr']
    zergAM_data.query('played >= 10')['mmr'].shape[0]/zergAM_data['mmr'].sha
    randomAM_data.query('played >= 10')['mmr'].shape[0]/randomAM_data['mmr']
```

```
protoss:0.5780338041615136
terran: 0.581703869194725
zerg: 0.5602833366356251
random:0.5254076878276063
```

Now that that's out of the way, let's just do some neat visuals. Doing a log y-scale allows us to see the density differences at the extremities of the plot. In the below plot, we can see that Terran drops off completely at 6500 MMR while Protoss and Zerg have a few people represented up to the 7000 MMR range. We can also see that from about 3000 MMR to 5500, the density for each race is similar, which lines up with a previous result we noticed.

In [ ]:
```
sns.displot(data=combined_AM, x='mmr',  hue='race', kind='kde', height=5, as
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x17fdcebc0>
```



The Fun Visual

I initially did this visual as a joke, not expecting it to give us anything meaningful, but it actually ended up showing something really cool. This is a plot of every player on the NA ladder who has played more than 10 games. There are over 85000 data points on it. These data points show something really cool.

If you look at the graph, you can see these weird vertical lines show up at various points. These lines are groups of players coalescing around something. Looking into it, I found out that the lines roughly align with certain MMR values that relate to the various leagues in the game (There are seven ladder leagues in the game, along subdivisions of each league). This suggests that people will get their MMR up to or down to a certain level to get into a league and then stay there.

In [ ]:
```
plt.figure(figsize=(300,15))
sns.scatterplot(data=combined_AM.query('played >= 10'), x='mmr', y='played',
```

Out[ ]:
```
[None, (10, 3500)]
```

In [ ]:
```
plt.figure(figsize=(25,10))
sns.scatterplot(data=combined_AM.query('played >= 10'), x='mmr', y='played',
```

Out[ ]:
```
[None, (100.0, 3500)]
```

In [ ]:
```
len(set(combined_AM.query('played >= 10')['name']))
```

Out[ ]:
```
58921
```

The Other Servers

Since there are three more servers to examine, we won't be going through them with the same level of granularity as we did with the North American server but they are available if you want to look through them yourself.

In [ ]:
```
protossEU_data = MMR_data.query('region == "EU" and race == "protoss"')
zergEU_data = MMR_data.query('region == "EU" and race == "zerg"')
terranEU_data = MMR_data.query('region == "EU" and race == "terran"')
randomEU_data = MMR_data.query('region == "EU" and race == "random"')

combined_EU = pd.concat([protossEU_data, terranEU_data, zergEU_data, randomE
print('Total number of players: {}\nTotal Protoss: {}\nTotal Terran: {}\nTot
```

```
Total number of players: 154243
Total Protoss: 47751
Total Terran: 41326
Total Zerg: 51938
Total Random: 13228
```

In [ ]:
```
print('mean MMR:\nprotoss:{}\nterran: {}\nzerg: {}\nrandom:{}'.format(
    protossEU_data['mmr'].mean(),
    terranEU_data['mmr'].mean(),
    zergEU_data['mmr'].mean(),
    randomEU_data['mmr'].mean()))
```

```
mean MMR:
protoss:2873.401771690645
terran: 2780.4738149331897
zerg: 2976.576828146929
random:2923.5136074992442
```

In [ ]:
```
print('median MMR:\nprotoss:{}\nterran: {}\nzerg: {}\nrandom:{}'.format(
    protossEU_data['mmr'].median(),
    terranEU_data['mmr'].median(),
    zergEU_data['mmr'].median(),
    randomEU_data['mmr'].median()))
```

```
median MMR:
protoss:2767.0
terran: 2665.0
zerg: 2912.0
random:2868.0
```

In [ ]:
```
protossKR_data = MMR_data.query('region == "KR" and race == "protoss"')
zergKR_data = MMR_data.query('region == "KR" and race == "zerg"')
terranKR_data = MMR_data.query('region == "KR" and race == "terran"')
randomKR_data = MMR_data.query('region == "KR" and race == "random"')

combined_KR = pd.concat([protossKR_data, terranKR_data, zergKR_data, randomK
print('Total number of players: {}\nTotal Protoss: {}\nTotal Terran: {}\nTot
```

```
Total number of players: 33284
Total Protoss: 9916
Total Terran: 8341
Total Zerg: 12253
Total Random: 2774
```

In [ ]:
```
print('mean MMR:\nprotoss:{}\nterran: {}\nzerg: {}\nrandom:{}'.format(
    protossKR_data['mmr'].mean(),
    terranKR_data['mmr'].mean(),
    zergKR_data['mmr'].mean(),
    randomKR_data['mmr'].mean()))
```

```
mean MMR:
protoss:3135.0100847115773
terran: 3132.793846404962
zerg: 3197.605562882148
random:3133.057678442682
```

In [ ]:
```
print('median MMR:\nprotoss:{}\nterran: {}\nzerg: {}\nrandom:{}'.format(
    protossKR_data['mmr'].median(),
    terranKR_data['mmr'].median(),
    zergKR_data['mmr'].median(),
    randomKR_data['mmr'].median()))
```

```
median MMR:
protoss:2983.0
terran: 3024.0
zerg: 3083.0
random:3057.0
```

In [ ]:
```
protossCN_data = MMR_data.query('region == "CN" and race == "protoss"')
zergCN_data = MMR_data.query('region == "CN" and race == "zerg"')
terranCN_data = MMR_data.query('region == "CN" and race == "terran"')
randomCN_data = MMR_data.query('region == "CN" and race == "random"')
combined_CN = pd.concat([protossCN_data, terranCN_data, zergCN_data, randomC
print('Total number of players: {}\nTotal Protoss: {}\nTotal Terran: {}\nTot
```

```
Total number of players: 83024
Total Protoss: 26502
Total Terran: 21025
Total Zerg: 29369
Total Random: 6128
```

In [ ]:
```
print('mean MMR:\nprotoss:{}\nterran: {}\nzerg: {}\nrandom:{}'.format(
    protossCN_data['mmr'].mean(),
    terranCN_data['mmr'].mean(),
    zergCN_data['mmr'].mean(),
    randomCN_data['mmr'].mean()))
```

```
mean MMR:
protoss:2849.4832842804317
terran: 2810.524396472471
zerg: 2922.0080856123664
random:2925.101174934726
```

In [ ]:
```
print('median MMR:\nprotoss:{}\nterran: {}\nzerg: {}\nrandom:{}'.format(
    protossCN_data['mmr'].median(),
    terranCN_data['mmr'].median(),
    zergCN_data['mmr'].median(),
    randomCN_data['mmr'].median()))
```

```
median MMR:
protoss:2741.0
terran: 2717.0
zerg: 2855.0
random:2876.0
```

In [ ]:
```
plt.figure(figsize=(300,15))
sns.scatterplot(data=combined_EU, x='mmr', y='played', hue='race', alpha=0.6
```

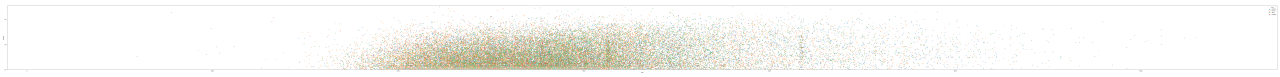
Out[ ]:
```
[None, (10, 3500)]
```

In [ ]:
```
plt.figure(figsize=(100,5))
sns.scatterplot(data=combined_KR, x='mmr', y='played', hue='race', alpha=0.6
```

Out[ ]:
```
[None, (10, 3500)]
```

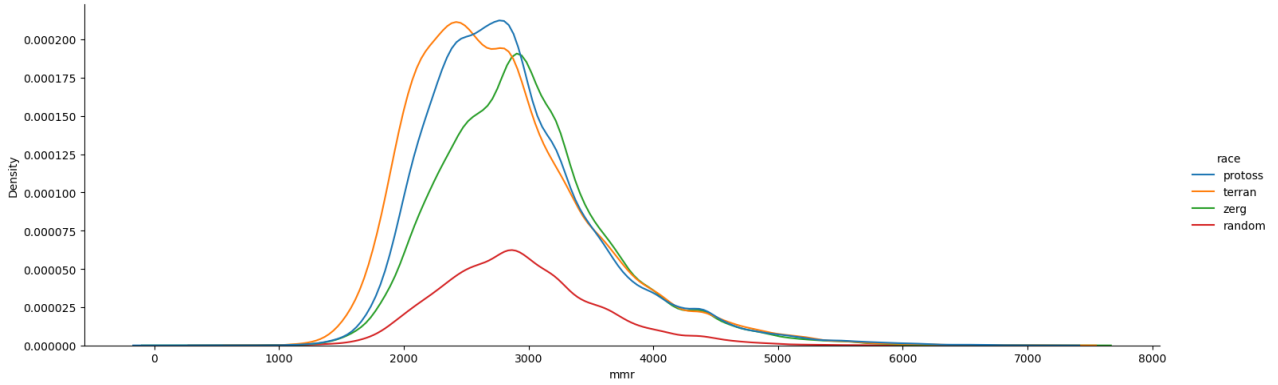
In [ ]:
```
plt.figure(figsize=(200,10))
sns.scatterplot(data=combined_CN, x='mmr', y='played', hue='race', alpha=0.6

# I ran into a weird issue that required me to manually fix the data for the
# Essentially, 8 or so rows of data ended up with no name, and thus were mis
# percentage to show up as the number of games played. My suspicion is it's
# the scraper was reading in one encoding standard, and the characters used
# nothing. Anyhow, I created broken.py, which can find the broken rows, then
# a placeholder name and delete the extra comma near the end of each row and
```

Out[ ]:
```
[None, (10, 3500)]
```



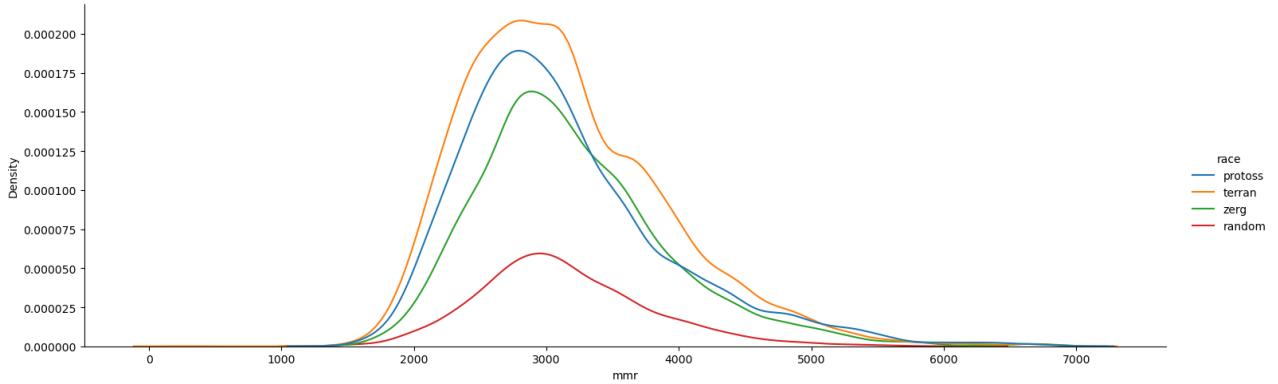
In [ ]:
```
sns.displot(data=combined_EU, x='mmr',  hue='race', kind='kde', height=5, as
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x298cfe9e0>
```



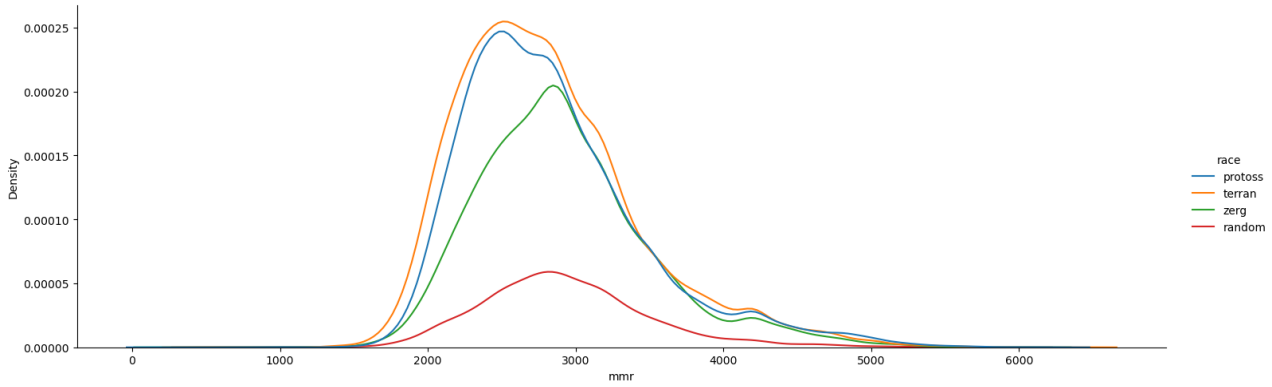
In [ ]:
```
sns.displot(data=combined_KR, x='mmr',  hue='race', kind='kde', height=5, as
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x178426920>
```



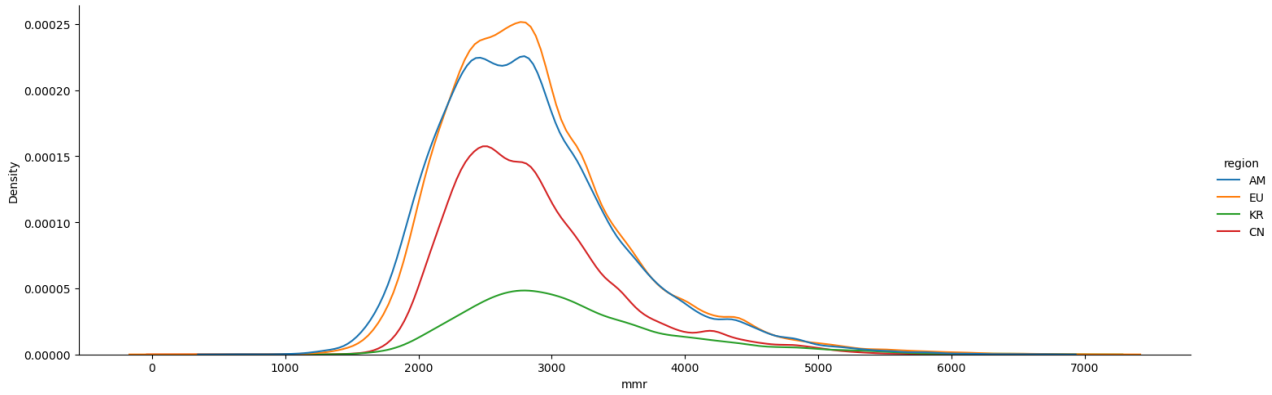
In [ ]:
```
sns.displot(data=combined_CN, x='mmr',  hue='race', kind='kde', height=5, as
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1783d7dc0>
```



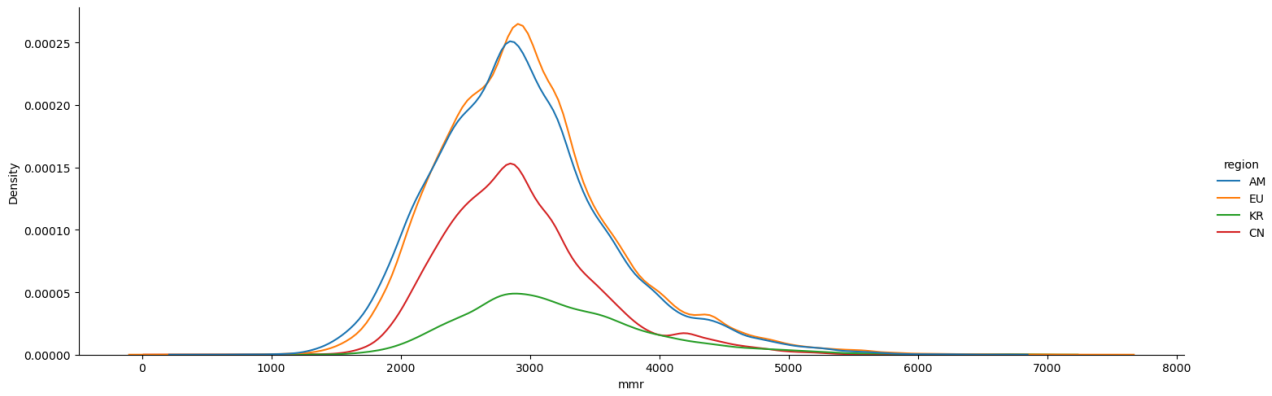
In [ ]:
```
all_protoss = MMR_data.query('race == "protoss"')
sns.displot(data=all_protoss, x='mmr',  hue='region', kind='kde', height=5,
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1782b8370>
```



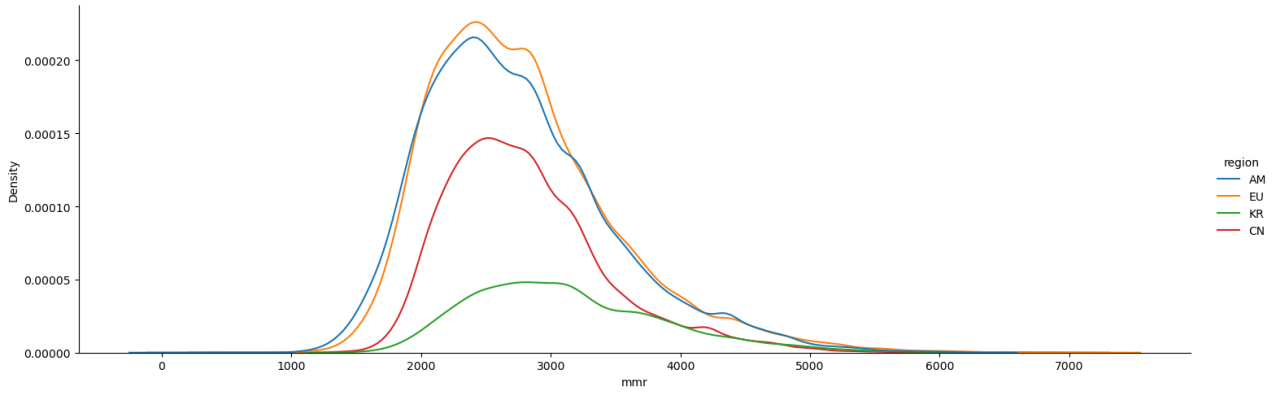
In [ ]:
```
all_zerg = MMR_data.query('race == "zerg"')
sns.displot(data=all_zerg, x='mmr',  hue='region', kind='kde', height=5, asp
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1782de7a0>
```



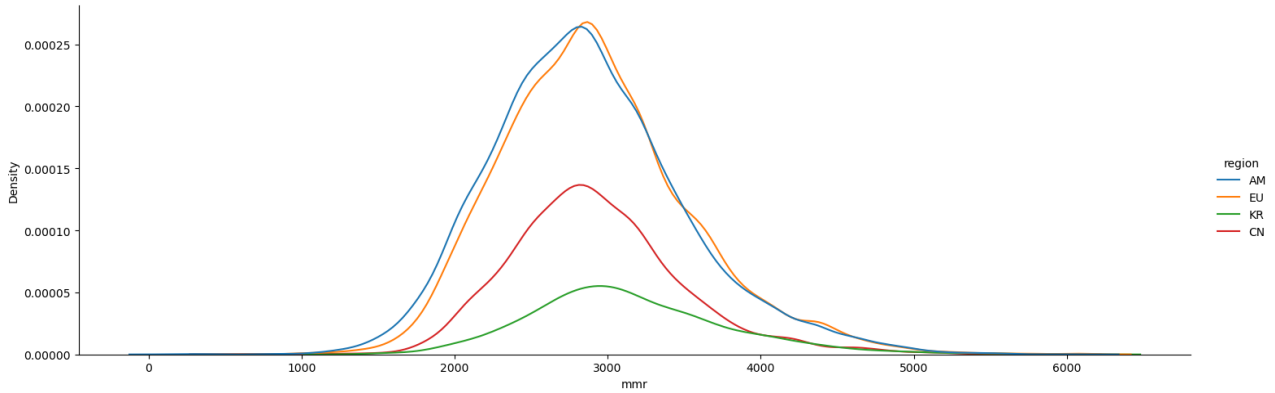
In [ ]:
```
all_terran = MMR_data.query('race == "terran"')
sns.displot(data=all_terran, x='mmr',  hue='region', kind='kde', height=5, a
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x292627a30>
```



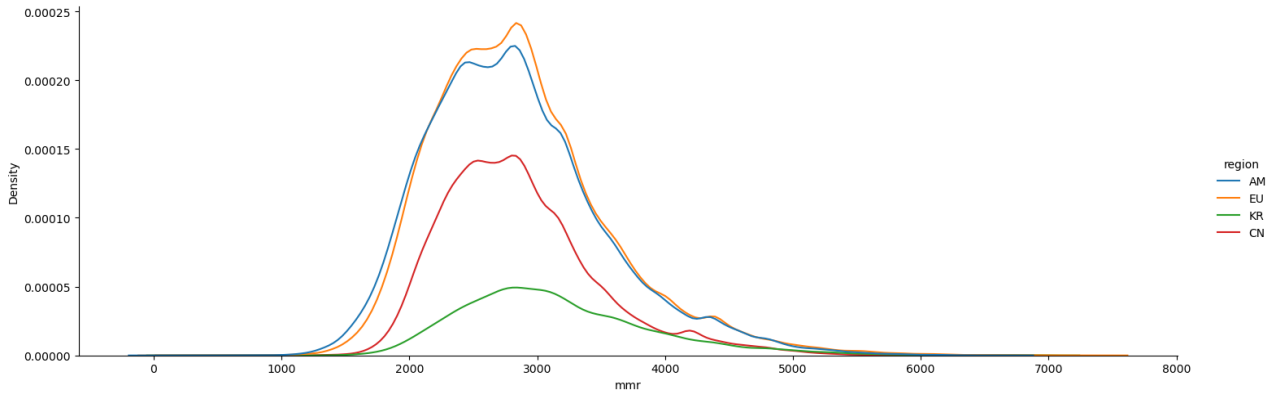
In [ ]:
```
all_random = MMR_data.query('race == "random"')
sns.displot(data=all_random, x='mmr',  hue='region', kind='kde', height=5, a
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x1795c70d0>
```



In [ ]:
```
sns.displot(data=MMR_data, x='mmr',  hue='region', kind='kde', height=5, asp
```

Out[ ]:
```
<seaborn.axisgrid.FacetGrid at 0x28af2b940>
```



In [ ]:
```
plt.figure(figsize=(300,15))
sns.scatterplot(data=MMR_data, x='mmr', y='played', hue='race', alpha=0.6).s
```

Out[ ]:
```
[None, (10, 3500)]
```

In [ ]: