Global Networking:

Standardizing International Basketball Statistics with a Network-Based Approach

Alexander Beard

Advised by:

Michael J. Tomas III

Barbara Arel

Abstract

European basketball is different than the NBA in that teams play much different schedules, due to the nature of their scheduling format. Because of this, it’s difficult to analyze player statistics across leagues. By building an incomplete competition network from each player’s actual schedule, and using supervised learning to complete it by predicting player performance in hypothetical matchups, the competition network can be completed, allowing for player statistics to be standardized across all leagues in the future.

Introduction

The table below shows Jimmer Fredette’s statistics from the 2014-2015 season (Jimmer Fredette NBA, RealGM), his last playing a significant amount of games in the NBA , and the 2016-2017 season (Jimmer Fredette International, RealGM), his first season playing in China after a brief stint on the Knicks.

Year	Team	MPG	FG%	PPG	RPG	APG
2014-2015	New Orleans	10.2	38.0	3.6	0.8	1.2
2016-2017	Shanghai Dongfang	40.5	47.4	37.6	8.2	4.2

How did Fredette’s statistics improve so much? Did he develop that much in the 18 months in between? Did his five minutes in two games in New York, where he now has the franchise record for effective FG% after hitting his only shot and going 4/5 from the line (Fox Sports), inspire him to become a different player entirely? Unlikely.

What’s really at play here is the difference in strength of schedule. Differences in strength of schedule are sometimes mentioned in the NBA, but the difference in skill level isn’t that large (no matter what some tanking teams might make you think), and regardless – each team plays roughly the same schedule (NBA Stuffer). European basketball is likely quite different, for two reasons – there are approximately three times as many teams (so teams play less similar schedules), and the European schedule format is quite different than the NBA’s (which will be discussed in greater detail in the Data Methodology section).

These differences make it difficult to take European statistics at face value. It would be extremely easy to compare statistics between players if they had all played every one of the 88 other teams in the league. To approximate this, I attempted to predict the player vs team matchup results that didn’t happen in a given year, which could then, when every player has played the “same” schedule, allows standardized statistics to be calculated.

Literature Review

There are four international leagues that I have data for: The Greek Basketball League, the Spanish Liga ACB, the Italian Lega Basket Serie A, and the French LNB Pro A. Each of these four leagues is the highest level of competition in their respective countries.

In addition to games within leagues (which is usually referred to as club play), I have data for EuroCup and EuroLeague, which are tournaments where the participants are teams from different leagues. In international competition, EuroLeague is regarded as the top-tier tournament, and EuroCup as the second-tier (Fraschilla, 2017). Both tournaments have a regular season component, which happens throughout the year, in addition to club play. The top performing teams in the regular season advance to the playoffs (EuroCup also has an intermediate round robin component), where a winner is decided with a head-to-head bracket (7Days EuroCup Format, Turkish Airlines EuroLeague Basketball Format).

Previous papers and articles have discussed the challenges of analyzing international play. For example, Motomura (2014) examined the NBA’s history of drafting international player. This article is relevant because it illustrates the ultimate goal of this work, which is to look into the performance of international players. Their input is draft position, but in future work using standardizing international statistics, they could either be placed as a comparison to draft position (which one predicts NBA performance better), or as an input, with draft position as an output, to essentially test the accuracy of the standardized statistics, assuming that the draft positions are with perfect information (which is obviously not true, but they might be close enough to provide a decent approximation).

Other authors have looked more specifically about international statistics, and how they can be adjusted to be a more accurate representation. Vashro (2015) compares leagues to other leagues, by optimizing the different relative competition scores to minimize the disagreement, then standardizing them to a z-score (mean of 0, standard deviation of 1). This is actually something I tried to do, but on a player-game level, but I found it intractable.

In 2014, Vashro’s work on projecting international players also adjusts for strength of schedule, but on a player level. His series talks a lot about the challenges of strength of schedule, and about the differences between statistics in different leagues. For example, with his league-wise adjustments, he adjusts (for example) the French league being pass-happier). An extremely relevant quote that’s really why I tried to use a separate network-based approach for each statistic:

“Strength of competition does not necessarily impact all skill-sets in the same way, and different leagues may inflate one statistic while deflating another. These factors mean that information is lost in any simple SOS adjustment.”

The next step that I take is not assuming that within leagues, the inflation/deflation for each statistic is the same. If there are significant differences between defenses within a league, that’s something my model should capture. Also, some players may have unique challenges or advantages against certain defenses, which is why looking at player performance as a individual player-game level competition network could improve the predictive power.

My network-based follows a similar approach to Park and Yook (2014). Their peer-reviewed paper is looking at a slightly different problem (inferring the ranks of nodes in a competition network, rather than the specific adjustments), but the general concept is similar. As was described in the introduction, we start out with an incomplete network, because not every player has played every team. To infer the hypothetical complete network, they infer the missing edges (as can be seen in Figure 1b of their paper). The main differences are what is trying to be predicted in the end (as mentioned), the fact that my competition network is a bipartite graph (players do not play other players specifically, they play defenses), and, of course, the specific application: While their main application is college football, mine is international basketball.

Data Methodology

Summary – before any pruning

Year	# Players	# Teams	# Matchups that happened	# Potential Matchups	Network Completion %
2014-2015	1563	104	34315	162552	21.11
2015-2016	1542	102	34746	157284	22.09
2016-2017	1378	89	30541	122642	24.90

Pruning

The tournament games mentioned in the previous section are extremely important, because they allow players in different leagues to be compared, by providing connections between teams in different leagues Without those connections, there would be no way to project player performance against teams not in their league.

Figure 1 shows the full dataset, before any pruning. The gray dots in the graphs are teams that are not in any of the four leagues that Basketball Reference has club play boxscore data for, and they appear in this dataset because their EuroLeague or EuroCup play is tracked. For this analysis, because teams with no club play data will have about half as much overall data, those teams will not be considered. Figure 2 shows the competition network with those teams removed.

Summary – Teams with no club play data removed

Year	# Players	# Teams	# Matchups that happened	# Potential Matchups	Network Completion %
2014-2015	1042	66	24509	68772	35.64
2015-2016	1050	66	25061	69300	36.16
2016-2017	1032	65	22972	67080	34.25

One special case, as can be seen in Figures 1 and 2, is that the French teams are isolated from the rest of the teams in 2016-2017. In 2016, the French Basketball Federation made the decision to no longer participate in EuroCup or EuroLeague, and instead be a part of the FIBA Champions League (Stroggylakis, 2016). Because of this, the 2016-2017 French players and teams were discarded. Figure 3 shows the effect of removing those teams on the 2016-2017 competition network.

Summary – 2016-2017 French teams removed

Year	# Players	# Teams	# Matchups that happened	# Potential Matchups	Network Completion %
2014-2015	1042	66	24509	68772	35.64
2015-2016	1050	66	25061	69300	36.16
2016-2017	765	47	16916	35955	47.05

There are some hypothetical player vs defense matchups where there's not enough information to predict the result. A minimum number of games played could be used, but there could be situations where the distribution of games played skews the amount of information available – it would be very difficult to predict hypothetical matchups for a player who has played all of his team's club games, but none of their EuroCup/EuroLeague games. To fix this, matchups were only predicted if they had at least 10 triangles (which will be explained later in this section). As you can see in the summary below, very few matchups had to be removed (around 50 each year), so there shouldn't be much of an effect on the end result.

Summary – Player-defense matchups with under 10 triangles removed

Year	# Players	# Teams	# Matchups that happened	# Potential Matchups	Network Completion %
2014-2015	983	66	24449	64878	37.68
2015-2016	1012	66	25022	66792	37.46
2016-2017	716	47	16864	33652	50.11

Scraping

All of this data is scraped from Basketball Reference. Matt Goldberg wrote a python scraping library for the NBA and NFL (Goldberg, Github). I forked it, and added international basketball support (Beard, Github). Figure 4 shows an example of a source webpage, and Figure 5 shows the resulting pandas DataFrame and the python snippet that scrapes it.

The articles in the Literature Review use season-level statistics, because for a long time, that's all that was available. However, Basketball Reference has game-level data available from the 2014-2015 season on, as can be seen in Figure 6.

By going through the the full schedule for each league, and pulling out each boxscore and concatenating it,, we obtain a list of each player's performances against each defense. Figure 7 shows the format of this dataset.

Network Representation

This allows a weighted bipartite network to be constructed, where the two disjoint groups of nodes are players and defenses, and the edge weights are the resulting statistic for that matchup. For example, if Figure 8 was the dataset of player performances against defenses, Figure 9 would show the resulting bipartite network.

Triangles

The main projection building block used is a triangle. For example, if we were trying to complete the network shown in Figures 8 and 9, the hypothetical amount of points that Player Y would score against Defense A would need to be calculated. While Player Y hasn't played Defense A, they have played Defense B, and scored 25 points. Therefore, we need some way to compare Defense A to Defense B. Luckily, Player X has played both Defense A and Defense B (see Figure 10).

Because Player X scored 22 points against Defense B, and 20 against Defense A, we assume that Defense A will allow (20/22) = 0.91 = 91% of the points Defense B does to the same player. So, we would expect Player Y to score 25 x 0.91 = 22.75 points against Defense A.

This prediction by itself would probably be very noisy. However, for each hypothetical matchup, there are multiple triangles that can be used to predict, which can be seen in the larger example in Figure 11. As you can see in the below table, to predict the result if Player Y faced Defense B (if it hadn't have actually happened), there would be four different triangles that could be used, and therefore, four different potential predictions:

Player Y vs Defense B predictors

Triangle Components	**Prediction**
Defense A, Player W	5.75
Defense C, Player W	35.11
Defense A, Player X	5.71
Defense C, Player X	29.00

Because each individual prediction is so noisy, many predictions will have to be used.

Triangle Selection/Ordering Methods

The obstacle once the triangle predictions are calculated is that to use traditional supervised learning methods, we need a properly shaped, rectangular matrix, but because each different matchups have different numbers of triangles, we actually have a jagged matrix, as can be seen in Figure 12. Also, we need some way to order the triangles, so they can then be made into a properly shaped matrix.

The ordering method used was number of games played. As can be seen in the Triangles section, a triangle is made up of three matchups – in the example shown in Figure 8 and 9, the three matchups that make up the prediction are Player X vs Defense A, Player X vs Defense B, and Player Y vs Defense B. If Player X has played Defense A three times, Player X has played Defense B two times, and Player Y has played Defense B one time, the total number of games played for this triangle is 3 + 2 + 1 = 6 games. This ordering is used because as there is more data used to make a prediction, the prediction should be less noisy.

Once the triangles were ordered, they need to be turned into a consistent set of predictions or aggregate predictions. Again operating on the assumption that more data is better, instead of only taking the top 5 or 10 predictions and leaving out the rest, the set of triangles is split into either 5 or 10 buckets. For example, if there were 100 triangles, split into 5 buckets, the first bucket would be the top 20 triangles by number of games played, the second bucket would be triangles 21-40, and so on. Because, in the

Pruning section, matchups with less than 10 triangles are removed, either method will work for any hypothetical matchup.

Once the triangles were split into buckets, either the median or mean of the predictions from the triangles in that bucket were calculated. Combining the two decisions, there were four possible treatments:

- 5 buckets, mean
- 5 buckets, median
- 10 buckets, mean
- 10 buckets, median

In addition to these 5 or 10 data points, the player's mean and median overall average was also considered (the matrix structure for the 5 buckets/mean treatment is shown in Figure 13). The null model, for comparison on if the additional data helped at all, was just the player's mean and median overall average.

Models

The two prediction models I used were Linear Regression, and Random Forest Regressor, both from the scikit-learn package (Pedregrosa et al, 2011). As you can see in Figure 14, a Linear Regression fits a line to the y-data given, using the features given in the x-matrix. It's best for data with linear relationships between the features and the output, and it also is very transparent, which is why it was chosen.

The other model chosen is a good contrast for the Linear Regression – Random Forests are not very transparent, but generally have higher performance. An example of the Random Forest Regressor can be seen in Figure 15.

Statistics

The five statistics I used were points, total rebounds (offensive + defensive), assists, steals, and blocks, all adjusted to be per minute. These five were chosen instead of a more holistic, overall performance statistic like Game Score because, as spoken to in the Literature Review section, defenses could be stingy in regards to rebounds, but allow many assists. Because all triangles were being considered, and therefore players across archetypes were being used to compare, this separation is important. Therefore, for each year, and statistic, there is a separate network.

Cross-Validation

For each of the three iterations, a random selection of 5% of the data was “hidden” for the model training. A new network was built *without* this data. The matrices used for training the model were then built, with each row being a matchup that actually happened that had not been hidden. The triangles were generated from the new network without the hidden data. After the model was trained with this training data, the model was then used to predict the 5% of matchups that had been hidden.

All of the statistics in the results are calculated from the predictions of the hidden data vs the actual values. The main statistics used were mean absolute error (MAE), and root mean squared error (RMSE). Because the different statistics can have very different averages (for example, points per minute vs blocks per minute), the MAE and RMSE were also presented scaled by the average actual value among the hidden set. To make the tables easier to read, these statistics were averaged across the three iterations – as can be seen in the two examples in Figure 16, most of the difference between iterations is the distribution of the actual values (which is variation that we want to average out)- the slope of the residuals vs actual values doesn’t change much.

Results

As can be seen in Figures 18 through 21, looking just at the linear regression variations, there’s very little difference between the null model and the other variants with additional information. So little that it’s almost definitely statistically insignificant. What’s interesting is that in almost every case, every linear regression variation performs better than almost every random forest variation.

Looking specifically at the random forest variations, as you can see in both:

null → median 5 → median 10

and: null → mean 5 → mean 10

It almost always improves with more data (since it technically contains the same data, this could be also stated as more precise data). Also, for the most part, random forest methods median variations outperform random forest mean methods, which speaks to how noisy the data is.

Conclusion

While the predictions are definitely not good, they’re somewhat encouraging – given how noisy a particular matchup is, especially with players and teams that don’t play each other often (or even have many of the same opponents), predicting points per minute within approximately 40% (which is what a 0.4 scaled MAE is equivalent to) could be a lot worse.

Even though the random forest variations are less accurate, the fact that they consistently get better with more information suggests there’s potential. Increasing the number of maximum features (considered at each split of the tree that makes up the forest), or feeding it more data somehow, looks like it would continue to improve performance. One possible way to have more data to train on would be to combine years, in conjunction with some kind of age curve to adjust for players aging into or out of their prime.

Part of the noise might be from the fact, mentioned in the Data Methodology section, that all archetypes are thrown together. One way to add more data would be to add on more features that were more of a drill down into similar players: buckets ordered by closest player heights, same player position. The same could also be done on the defensive side – similar teams, either by league, win/loss record, playoff performance, etc. could be weighted higher, or at least considered separately so that the model could decide whether they should be weighted higher.

Possible Extensions

The clear next step would be to use the best-performing model to predict the matchups that actually haven’t happened, actually doing what was simulated with cross-validation. With a completed competition network, or at least mostly complete (accounting for the pruning), a player’s perfomances across all matchups (both real and projected) could be averaged, giving them standardized per minute statistics.

With these standardized statistics in hand, there are many avenues for future analysis. Which player archetypes are over/under drafted compared to their statistics, standardized statistics vs draft position as an indicator for NBA success, which statistics translate best to the NBA… the list could keep going on. Because the predictions were so inaccurate, it seems like there’s no real point in standardizing the statistics now, but with more work on the prediction side, it’s definitely a possibility that they could be accurate enough to perform future analysis with.

References

7DAYS EuroCup Format, www.eurocupbasketball.com/eurocup/competition/format.

Beard, Alexander. “abeard1/Sportsref.” *GitHub*, www.github.com/abeard1/sportsref.

Fraschilla, Fran. “Top Basketball Leagues in the World Outside the NBA.” *ESPN, ESPN Internet Ventures*, 7 Dec. 2017, www.espn.com/nba/story/_/id/21691644/fran-fraschilla-ranks-world-top-basketball-leagues-nba.

Goldberg, Matt. “Mdgoldberg/Sportsref.” *GitHub*, www.github.com/mdgoldberg/sportsref.

“How the NBA Schedule Is Made.” *NBA Stuffer*, www.nbastuffer.com/analytics101/how-the-nba-schedule-is-made/.

“Jimmer Fredette International.” *RealGM - Basketball News, Rumors, Scores, Stats, Analysis, Depth Charts, Forums*, www.basketball.realgm.com/player/Jimmer-Fredette/Summary/9263#International.

Jimmer Fredette NBA.” *RealGM - Basketball News, Rumors, Scores, Stats, Analysis, Depth Charts, Forums*, www.basketball.realgm.com/player/Jimmer-Fredette/Summary/9263.

Fox Sports. “Knicks to Cut Jimmer Fredette despite 'Record-Setting' Stint.” *FOX Sports*, FOX Sports, 2 Mar. 2016, www.foxsports.com/nba/story/new-york-knicks-jimmer-fredette-030216.

Motomura, Akira. “MoneyRoundball? The Drafting of International Players by National Basketball Association Teams.” *Journal of Sports Economics*, vol. 17, no. 2, Mar. 2014, pp. 175–206., doi:10.1177/1527002514526411.

Park, Juyong, and Soon-Hyung Yook. “Bayesian Inference of Natural Rankings in Incomplete Competition Networks.” *Scientific Reports*, vol. 4, no. 1, 2014, doi:10.1038/srep06212.

Pedregrosa, Fabian et al. “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research*, vol 12, 2011, pp 2825-2830.

Stroggylakis, Antonis. “No French Teams in Euroleague/Eurocup.” *Eurohoops*, Eurohoops.net, 27 June 2016, www.eurohoops.net/en/featured/269207/no-french-teams-in-euroleagueeurocup/.

“Turkish Airlines EuroLeague Basketball Format.” *Turkish Airlines EuroLeague*, www.euroleague.net/competition/format.

Vashro, Layne. “Projecting International Prospects, Part 1.” *FanSided*, FanSided, 13 Aug. 2014, www.fansided.com/2014/08/13/projecting-international-prospects/.

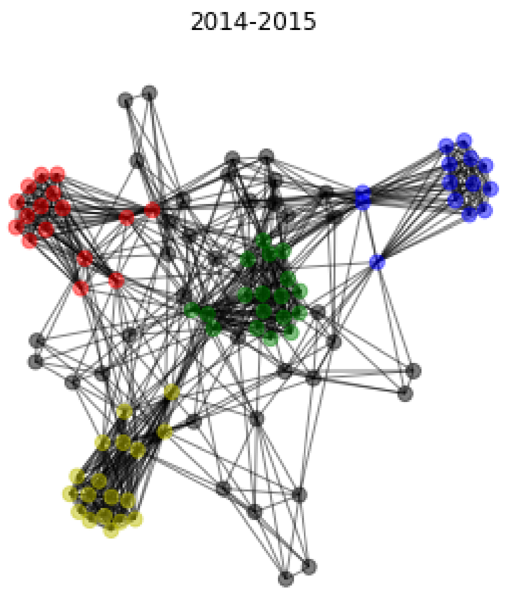
Vashro, Layne. “Projecting International Prospects, Part 2.” *FanSided*, FanSided, 15 Aug. 2014, www.fansided.com/2014/08/15/projecting-international-prospects-part-2/.

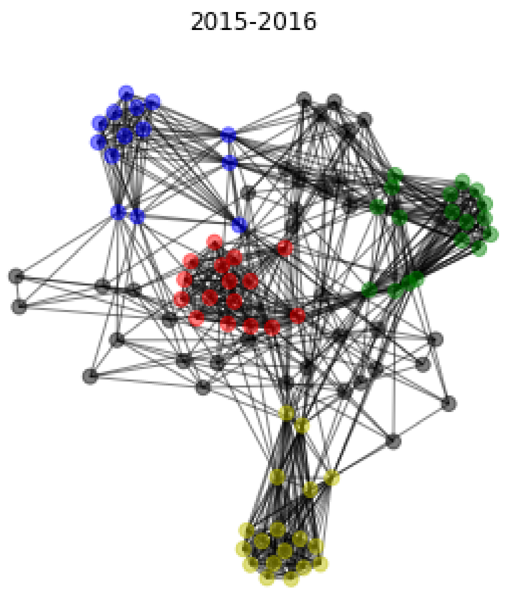
Vashro, Layne. “Projecting International Prospects, Part 3.” *FanSided*, FanSided, 21 Aug. 2014, www.fansided.com/2014/08/21/projecting-international-prospects-part-3/.

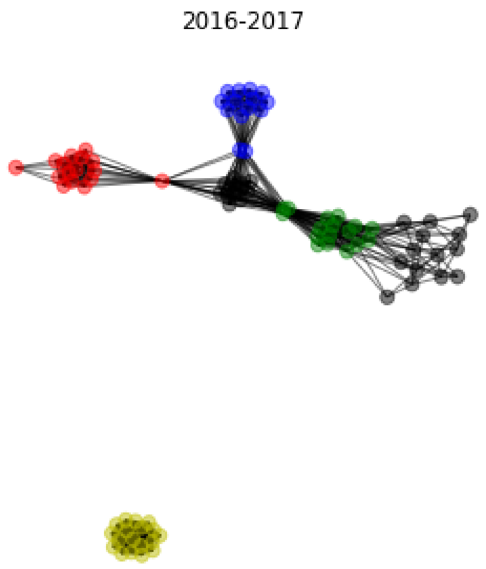
Vashro, Layne. “Deep Dives: Measuring Level of Competition Around the World.” *FanSided*, FanSided, 6 Nov. 2015, www.fansided.com/2015/11/06/deep-dives-measuring-level-of-competition-around-the-world/.

Figures

Figure 1: Competition Networks by Year – before any pruning







Legend for all competition networks:

Greek Basketball League - blue

Spanish Liga ACB – green

Italian Lega Basket Seri A – red

French LNB Pro A – yellow

Other - gray

Note: This is teams vs teams, which different than players vs teams, but that competition network is so large its hard to visualize well. Therefore, this is a simplified representation.

Figure 2: Competition Networks by Year - Teams with no club play data removed

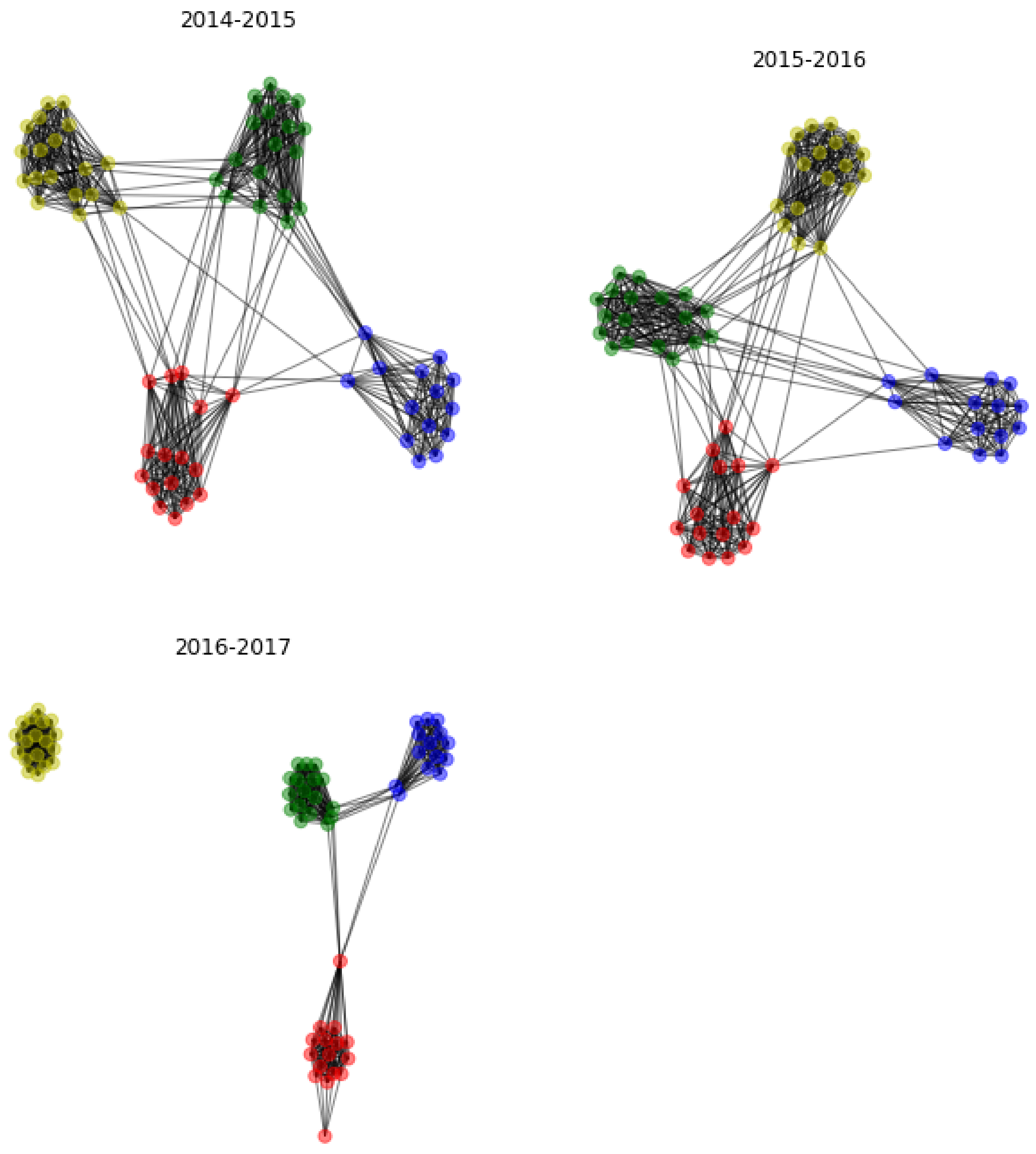


Figure 3: 2016-2017 Competition Network - French teams removed

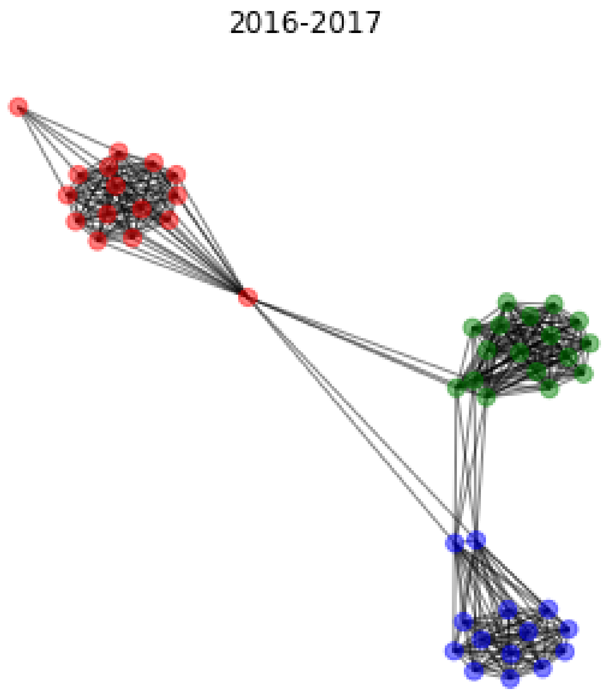
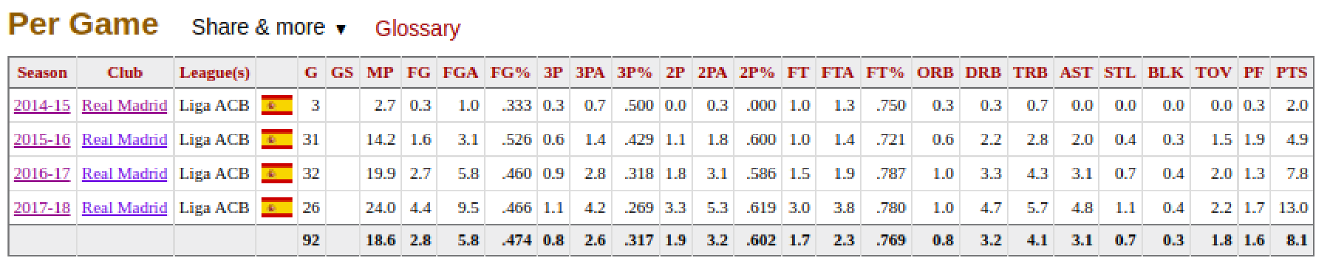


Figure 4: Sportsref Source webpage table

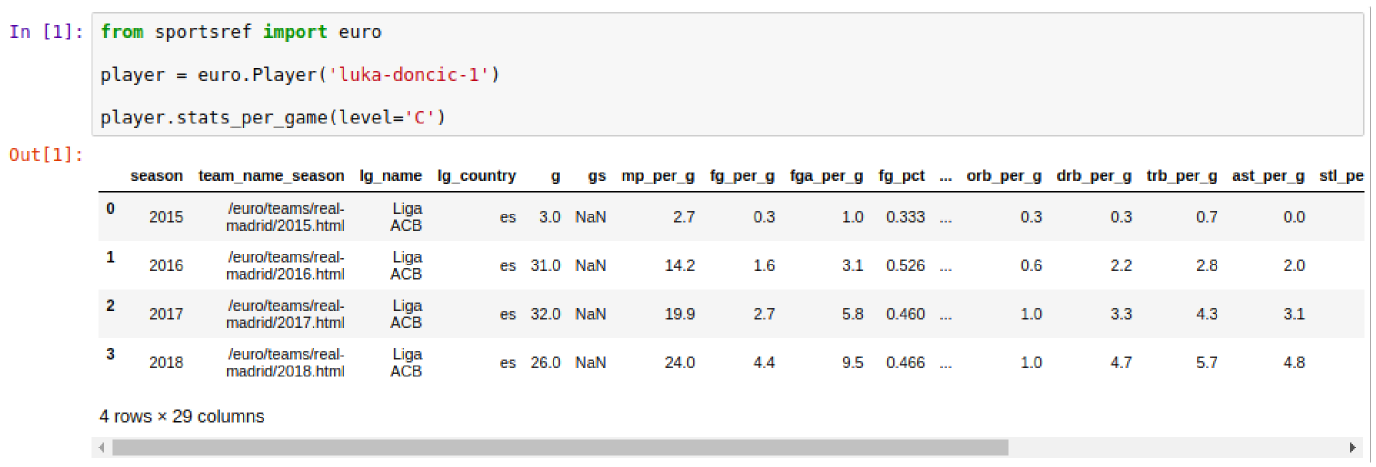

Per Game Share & more ▾ Glossary

Season	Club	League(s)		G	GS	MP	FG	FGA	FG%	3P	3PA	3P%	2P	2PA	2P%	FT	FTA	FT%	ORB	DRB	TRB	AST	STL	BLK	TOV	PF	PTS
2014-15	Real Madrid	Liga ACB		3		2.7	0.3	1.0	.333	0.3	0.7	.500	0.0	0.3	.000	1.0	1.3	.750	0.3	0.3	0.7	0.0	0.0	0.0	0.0	0.3	2.0
2015-16	Real Madrid	Liga ACB		31		14.2	1.6	3.1	.526	0.6	1.4	.429	1.1	1.8	.600	1.0	1.4	.721	0.6	2.2	2.8	2.0	0.4	0.3	1.5	1.9	4.9
2016-17	Real Madrid	Liga ACB		32		19.9	2.7	5.8	.460	0.9	2.8	.318	1.8	3.1	.586	1.5	1.9	.787	1.0	3.3	4.3	3.1	0.7	0.4	2.0	1.3	7.8
2017-18	Real Madrid	Liga ACB		26		24.0	4.4	9.5	.466	1.1	4.2	.269	3.3	5.3	.619	3.0	3.8	.780	1.0	4.7	5.7	4.8	1.1	0.4	2.2	1.7	13.0
				92		**18.6**	**2.8**	**5.8**	**.474**	**0.8**	**2.6**	**.317**	**1.9**	**3.2**	**.602**	**1.7**	**2.3**	**.769**	**0.8**	**3.2**	**4.1**	**3.1**	**0.7**	**0.3**	**1.8**	**1.6**	**8.1**

Source:

https://www.basketball-reference.com/euro/players/luka-doncic-1-club.html#per_gameCLU0::none

Figure 5: Python call and resulting DataFrame

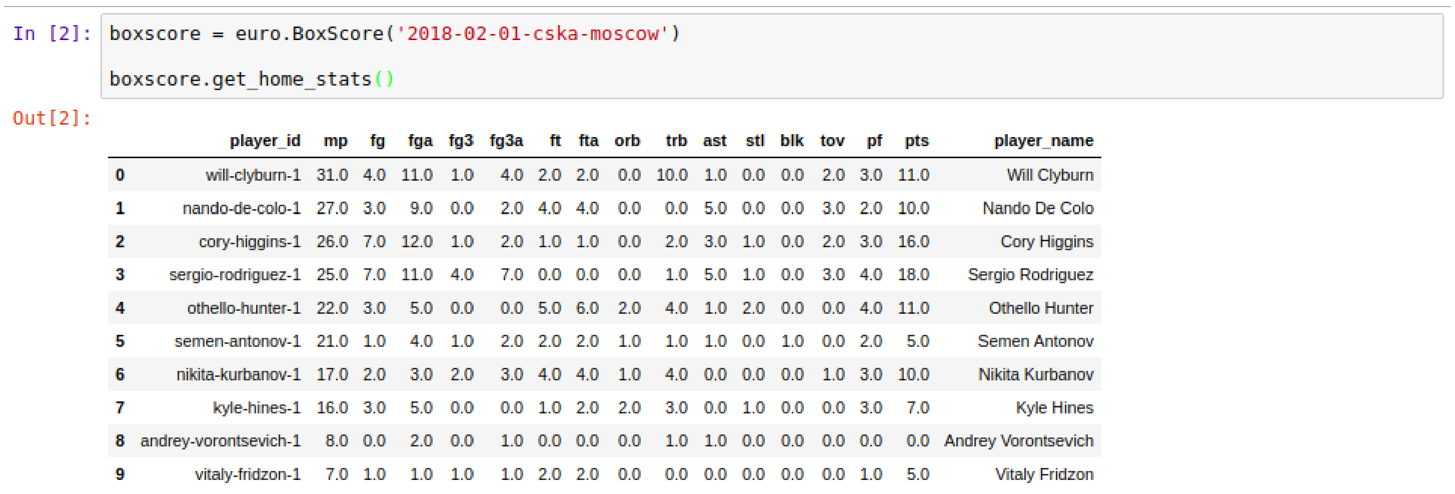


```
In [1]: from sportsref import euro

        player = euro.Player('luka-doncic-1')

        player.stats_per_game(level='C')

Out[1]:
```

	season	team_name_season	lg_name	lg_country	g	gs	mp_per_g	fg_per_g	fga_per_g	fg_pct	...	orb_per_g	drb_per_g	trb_per_g	ast_per_g	stl_pe
0	2015	/euro/teams/real-madrid/2015.html	Liga ACB	es	3.0	NaN	2.7	0.3	1.0	0.333	...	0.3	0.3	0.7	0.0	
1	2016	/euro/teams/real-madrid/2016.html	Liga ACB	es	31.0	NaN	14.2	1.6	3.1	0.526	...	0.6	2.2	2.8	2.0	
2	2017	/euro/teams/real-madrid/2017.html	Liga ACB	es	32.0	NaN	19.9	2.7	5.8	0.460	...	1.0	3.3	4.3	3.1	
3	2018	/euro/teams/real-madrid/2018.html	Liga ACB	es	26.0	NaN	24.0	4.4	9.5	0.466	...	1.0	4.7	5.7	4.8	

4 rows × 29 columns

Figure 6: Boxscore example



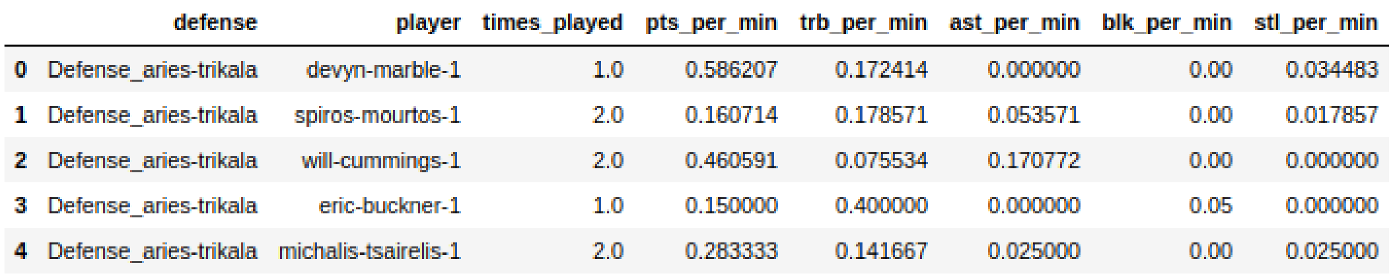
```
In [2]: boxscore = euro.BoxScore('2018-02-01-cska-moscow')

        boxscore.get_home_stats()

Out[2]:
```

	player_id	mp	fg	fga	fg3	fg3a	ft	fta	orb	trb	ast	stl	blk	tov	pf	pts	player_name
0	will-clyburn-1	31.0	4.0	11.0	1.0	4.0	2.0	2.0	0.0	10.0	1.0	0.0	0.0	2.0	3.0	11.0	Will Clyburn
1	nando-de-colo-1	27.0	3.0	9.0	0.0	2.0	4.0	4.0	0.0	0.0	5.0	0.0	0.0	3.0	2.0	10.0	Nando De Colo
2	cory-higgins-1	26.0	7.0	12.0	1.0	2.0	1.0	1.0	0.0	2.0	3.0	1.0	0.0	2.0	3.0	16.0	Cory Higgins
3	sergio-rodriguez-1	25.0	7.0	11.0	4.0	7.0	0.0	0.0	0.0	1.0	5.0	1.0	0.0	3.0	4.0	18.0	Sergio Rodriguez
4	othello-hunter-1	22.0	3.0	5.0	0.0	0.0	5.0	6.0	2.0	4.0	1.0	2.0	0.0	0.0	4.0	11.0	Othello Hunter
5	semen-antonov-1	21.0	1.0	4.0	1.0	2.0	2.0	2.0	1.0	1.0	1.0	0.0	1.0	0.0	2.0	5.0	Semen Antonov
6	nikita-kurbanov-1	17.0	2.0	3.0	2.0	3.0	4.0	4.0	1.0	4.0	0.0	0.0	0.0	1.0	3.0	10.0	Nikita Kurbanov
7	kyle-hines-1	16.0	3.0	5.0	0.0	0.0	1.0	2.0	2.0	3.0	0.0	1.0	0.0	0.0	3.0	7.0	Kyle Hines
8	andrey-vorontsevich-1	8.0	0.0	2.0	0.0	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	Andrey Vorontsevich
9	vitaly-fridzon-1	7.0	1.0	1.0	1.0	1.0	2.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	5.0	Vitaly Fridzon

Source:

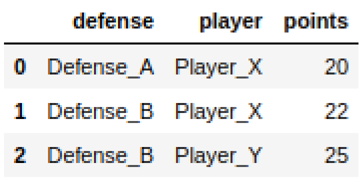
https://www.basketball-reference.com/euro/boxscores/2018-02-01-cska-moscow.html

Figure 7: Cleaned Dataset preview



	defense	player	times_played	pts_per_min	trb_per_min	ast_per_min	blk_per_min	stl_per_min
0	Defense_aries-trikala	devyn-marble-1	1.0	0.586207	0.172414	0.000000	0.00	0.034483
1	Defense_aries-trikala	spiros-mourtos-1	2.0	0.160714	0.178571	0.053571	0.00	0.017857
2	Defense_aries-trikala	will-cummings-1	2.0	0.460591	0.075534	0.170772	0.00	0.000000
3	Defense_aries-trikala	eric-buckner-1	1.0	0.150000	0.400000	0.000000	0.05	0.000000
4	Defense_aries-trikala	michalis-tsairelis-1	2.0	0.283333	0.141667	0.025000	0.00	0.025000

Figure 8: Example Dataset 1



	defense	player	points
0	Defense_A	Player_X	20
1	Defense_B	Player_X	22
2	Defense_B	Player_Y	25

Figure 9: Resulting Network from Dataset 1

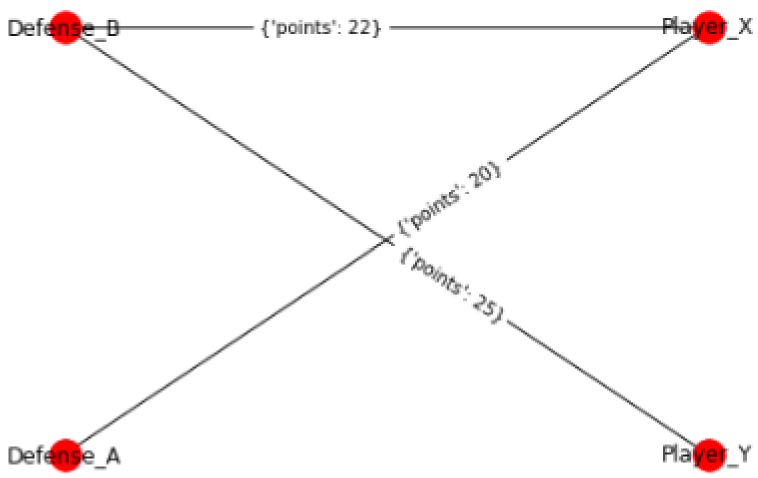


Figure 10: Example Dataset 1 Triangle Component

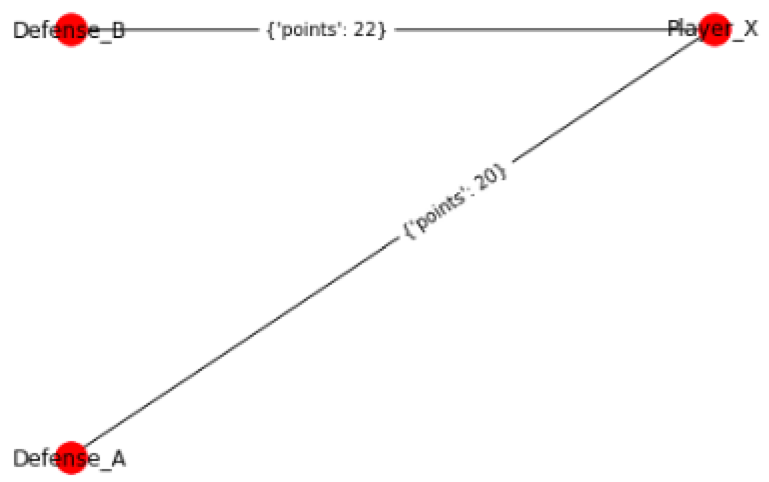
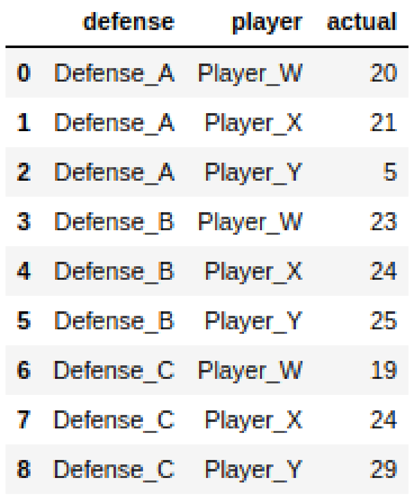


Figure 11: Example Dataset 2 and resulting Network-Based



	defense	player	actual
0	Defense_A	Player_W	20
1	Defense_A	Player_X	21
2	Defense_A	Player_Y	5
3	Defense_B	Player_W	23
4	Defense_B	Player_X	24
5	Defense_B	Player_Y	25
6	Defense_C	Player_W	19
7	Defense_C	Player_X	24
8	Defense_C	Player_Y	29

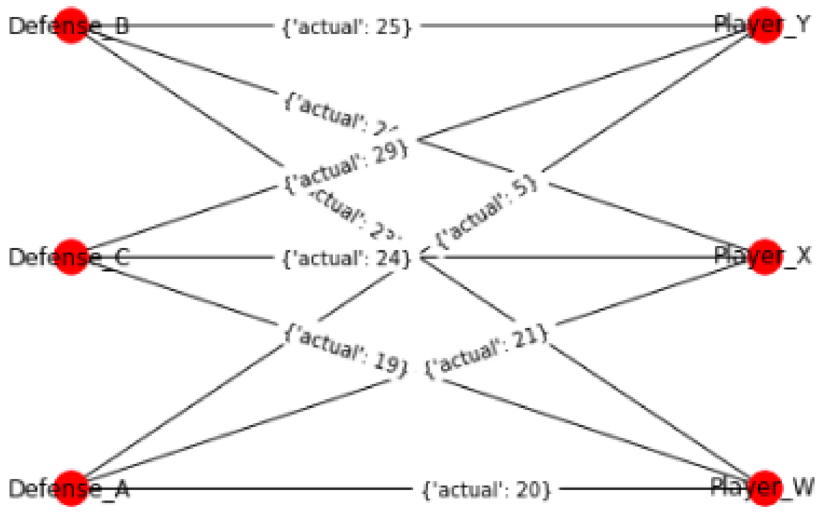


Figure 12: Jagged Matrix (without any Triangle Selection/Ordering)

Potential Matchup 1	T1	T2	…				
Potential Matchup 2							
…							

Figure 13: Matrix Structure - 5 buckets, mean

Potential Matchup 1:	Overall Mean	Overall Median	C1 Mean	C2 Mean	C3 Mean	C4 Mean	C5 Mean
Potential Matchup 2:	…	…	…	…	…	…	…
…							

Note: CX denotes Chunk X

Figure 14: Linear Regression Example (using example Dataset 2)

Feature Matrix

```
[   3.28,    4.6 ,   16.62,   20.12]
[   4.14,    4.8 ,   20.87,   25.26]
[  21.74,   21.88,   25.38,   30.53]
[  16.38,   19.  ,   22.86,  100.  ]
[  22.86,   23.  ,   26.68,  116.  ]
[  20.69,   24.15,   29.05,  105.  ]
[  19.83,   19.95,   27.84,  121.8 ]
[   5.71,    5.75,   29.  ,   35.11]
[   4.75,    5.71,   20.65,   25.  ]
```

corresponding output (actual performances)

```
[20]
[21]
[ 5]
[23]
[19]
[24]
[24]
[25]
[29]
```

Resulting prediction from trained model

```
[ 22.11]
[ 23.32]
[  5.4 ]
[ 24.12]
[ 18.68]
[ 23.64]
[ 23.67]
[ 26.32]
[ 22.75]
```

*Model is trained with above features/output

Resulting intercept of line:

```
[ 13.58]
```

Resulting feature weights:

```
[-2.1 ,  0.94,  0.47,  0.16]
```

*In practice, training data and test data are split – see Cross-Validation section

Figure 15: Random Forest Regressor Example

*Using same training data as Figure 14

Resulting prediction from trained model:

```
21.8
20.9
15.0
23.9
16.2
20.2
22.4
24.7
27.7
```

Figure 16: Residuals vs Actual across iterations – 2 examples

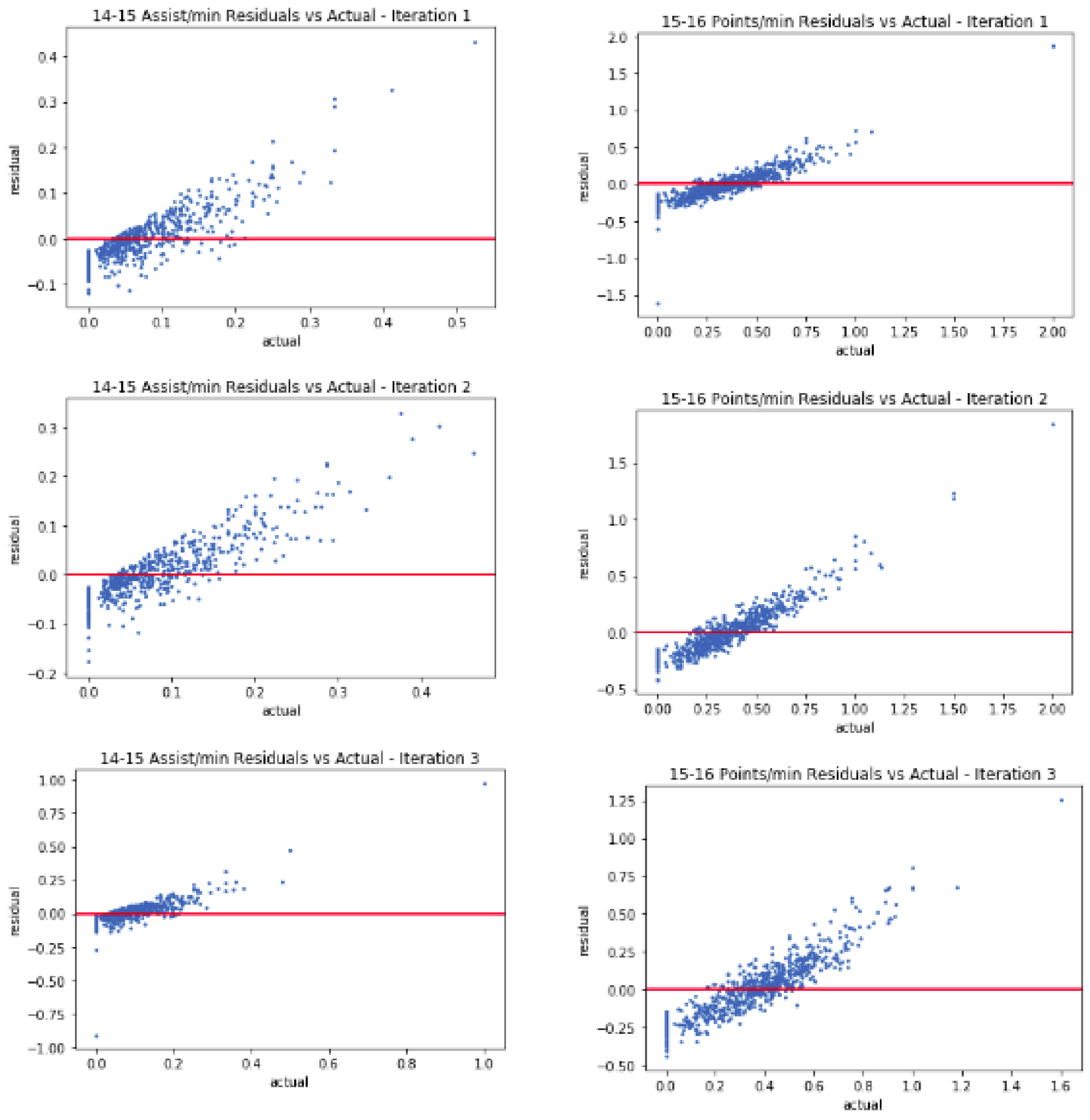


Figure 18: MSE (averaged across all 3 iterations)

	linreg_null	linreg_median_5	linreg_median_10	linreg_mean_5	linreg_mean_10	rf_null	rf_median_5	rf_median_10	rf_mean_5	rf_mean_10
pts_per_min2015	0.1483	0.1484	0.1482	0.1484	0.1484	0.1668	0.1595	0.1546	0.1628	0.1582
trb_per_min2015	0.0734	0.0732	0.0734	0.0732	0.0732	0.0811	0.0788	0.0783	0.0785	0.0788
ast_per_min2015	0.0440	0.0438	0.0439	0.0438	0.0438	0.0497	0.0473	0.0463	0.0471	0.0476
stl_per_min2015	0.0287	0.0287	0.0286	0.0287	0.0287	0.0339	0.0326	0.0319	0.0329	0.0329
blk_per_min2015	0.0158	0.0158	0.0158	0.0158	0.0158	0.0164	0.0161	0.0162	0.0164	0.0163
pts_per_min2016	0.1486	0.1474	0.1475	0.1474	0.1474	0.1663	0.1564	0.1571	0.1580	0.1563
trb_per_min2016	0.0708	0.0707	0.0707	0.0707	0.0707	0.0792	0.0764	0.0756	0.0768	0.0761
ast_per_min2016	0.0425	0.0420	0.0422	0.0420	0.0420	0.0472	0.0451	0.0455	0.0457	0.0452
stl_per_min2016	0.0309	0.0308	0.0309	0.0308	0.0308	0.0373	0.0358	0.0345	0.0358	0.0358
blk_per_min2016	0.0140	0.0140	0.0140	0.0140	0.0140	0.0151	0.0149	0.0151	0.0151	0.0154
pts_per_min2017	0.1387	0.1384	0.1384	0.1384	0.1384	0.1557	0.1489	0.1459	0.1477	0.1494
trb_per_min2017	0.0685	0.0681	0.0681	0.0681	0.0681	0.0781	0.0734	0.0732	0.0731	0.0719
ast_per_min2017	0.0409	0.0409	0.0410	0.0409	0.0409	0.0486	0.0461	0.0445	0.0458	0.0458
stl_per_min2017	0.0265	0.0265	0.0265	0.0265	0.0265	0.0299	0.0295	0.0294	0.0296	0.0299
blk_per_min2017	0.0130	0.0130	0.0130	0.0130	0.0130	0.0143	0.0142	0.0143	0.0141	0.0143

Figure 19: RMSE (averaged across all 3 iterations)

	linreg_null	linreg_median_5	linreg_median_10	linreg_mean_5	linreg_mean_10	rf_null	rf_median_5	rf_median_10	rf_mean_5	rf_mean_10
pts_per_min2015	0.2151	0.2158	0.2157	0.2158	0.2158	0.2351	0.2273	0.2163	0.2310	0.2264
trb_per_min2015	0.1141	0.1141	0.1142	0.1141	0.1141	0.1245	0.1212	0.1181	0.1211	0.1198
ast_per_min2015	0.0679	0.0677	0.0684	0.0677	0.0677	0.0729	0.0698	0.0686	0.0696	0.0700
stl_per_min2015	0.0453	0.0453	0.0451	0.0453	0.0453	0.0520	0.0505	0.0501	0.0506	0.0506
blk_per_min2015	0.0313	0.0313	0.0313	0.0313	0.0313	0.0347	0.0342	0.0343	0.0345	0.0342
pts_per_min2016	0.2081	0.2068	0.2071	0.2068	0.2068	0.2317	0.2195	0.2234	0.2264	0.2244
trb_per_min2016	0.1146	0.1145	0.1146	0.1145	0.1145	0.1255	0.1207	0.1201	0.1211	0.1208
ast_per_min2016	0.0653	0.0651	0.0653	0.0651	0.0651	0.0728	0.0699	0.0703	0.0705	0.0700
stl_per_min2016	0.0507	0.0507	0.0508	0.0507	0.0507	0.0622	0.0637	0.0588	0.0610	0.0612
blk_per_min2016	0.0264	0.0263	0.0263	0.0263	0.0263	0.0318	0.0315	0.0316	0.0317	0.0322
pts_per_min2017	0.1965	0.1964	0.1966	0.1964	0.1964	0.2240	0.2115	0.2112	0.2111	0.2134
trb_per_min2017	0.1059	0.1051	0.1056	0.1051	0.1051	0.1151	0.1098	0.1071	0.1099	0.1091
ast_per_min2017	0.0598	0.0598	0.0598	0.0598	0.0598	0.0691	0.0664	0.0649	0.0664	0.0664
stl_per_min2017	0.0346	0.0346	0.0346	0.0346	0.0346	0.0416	0.0412	0.0411	0.0412	0.0418
blk_per_min2017	0.0226	0.0226	0.0226	0.0226	0.0226	0.0281	0.0274	0.0278	0.0275	0.0282

Figure 20: Scaled MAE (averaged across all 3 iterations)

	linreg_null	linreg_median_5	linreg_median_10	linreg_mean_5	linreg_mean_10	rf_null	rf_median_5	rf_median_10	rf_mean_5	rf_mean_10
pts_per_min2015	0.4337	0.4341	0.4332	0.4341	0.4341	0.4878	0.4663	0.4522	0.4760	0.4626
trb_per_min2015	0.4598	0.4587	0.4594	0.4587	0.4587	0.5080	0.4936	0.4906	0.4914	0.4936
ast_per_min2015	0.6061	0.6037	0.6054	0.6037	0.6037	0.6849	0.6517	0.6382	0.6489	0.6562
stl_per_min2015	0.8592	0.8578	0.8550	0.8578	0.8578	1.0123	0.9756	0.9534	0.9830	0.9850
blk_per_min2015	1.1874	1.1873	1.1873	1.1873	1.1873	1.2309	1.2129	1.2170	1.2342	1.2233
pts_per_min2016	0.4247	0.4213	0.4216	0.4213	0.4213	0.4755	0.4471	0.4492	0.4518	0.4469
trb_per_min2016	0.4555	0.4548	0.4549	0.4548	0.4548	0.5099	0.4914	0.4867	0.4941	0.4895
ast_per_min2016	0.6276	0.6210	0.6234	0.6210	0.6210	0.6974	0.6673	0.6734	0.6751	0.6680
stl_per_min2016	0.9071	0.9064	0.9068	0.9064	0.9064	1.0970	1.0506	1.0125	1.0517	1.0533
blk_per_min2016	1.2866	1.2839	1.2835	1.2839	1.2839	1.3882	1.3649	1.3844	1.3852	1.4150
pts_per_min2017	0.4039	0.4030	0.4028	0.4030	0.4030	0.4533	0.4337	0.4249	0.4301	0.4350
trb_per_min2017	0.4516	0.4489	0.4487	0.4489	0.4489	0.5153	0.4841	0.4827	0.4822	0.4742
ast_per_min2017	0.6091	0.6085	0.6099	0.6085	0.6085	0.7232	0.6867	0.6625	0.6819	0.6815
stl_per_min2017	0.8932	0.8944	0.8919	0.8944	0.8944	1.0082	0.9959	0.9914	0.9992	1.0085
blk_per_min2017	1.2297	1.2297	1.2299	1.2297	1.2297	1.3538	1.3375	1.3487	1.3341	1.3539

Figure 21: Scaled RMSE (averaged across all 3 iterations)

	linreg_null	linreg_median_5	linreg_median_10	linreg_mean_5	linreg_mean_10	rf_null	rf_median_5	rf_median_10	rf_mean_5	rf_mean_10
pts_per_min2015	0.6289	0.6310	0.6308	0.6310	0.6310	0.6875	0.6648	0.6327	0.6756	0.6620
trb_per_min2015	0.7147	0.7146	0.7149	0.7146	0.7146	0.7800	0.7588	0.7399	0.7585	0.7502
ast_per_min2015	0.9362	0.9327	0.9430	0.9327	0.9327	1.0054	0.9618	0.9456	0.9589	0.9646
stl_per_min2015	1.3530	1.3530	1.3487	1.3530	1.3530	1.5545	1.5091	1.4980	1.5128	1.5130
blk_per_min2015	2.3528	2.3528	2.3528	2.3528	2.3528	2.6075	2.5662	2.5747	2.5903	2.5708
pts_per_min2016	0.5950	0.5912	0.5920	0.5912	0.5912	0.6625	0.6276	0.6386	0.6471	0.6416
trb_per_min2016	0.7372	0.7368	0.7374	0.7368	0.7368	0.8077	0.7766	0.7729	0.7791	0.7771
ast_per_min2016	0.9658	0.9620	0.9657	0.9620	0.9620	1.0762	1.0328	1.0388	1.0420	1.0350
stl_per_min2016	1.4901	1.4899	1.4920	1.4899	1.4899	1.8273	1.8727	1.7277	1.7918	1.7993
blk_per_min2016	2.4173	2.4161	2.4161	2.4161	2.4161	2.9170	2.8913	2.9027	2.9037	2.9550
pts_per_min2017	0.5722	0.5717	0.5724	0.5717	0.5717	0.6522	0.6158	0.6150	0.6147	0.6214
trb_per_min2017	0.6981	0.6930	0.6962	0.6930	0.6930	0.7587	0.7241	0.7062	0.7250	0.7191
ast_per_min2017	0.8910	0.8901	0.8908	0.8901	0.8901	1.0293	0.9894	0.9662	0.9895	0.9891
stl_per_min2017	1.1656	1.1680	1.1665	1.1680	1.1680	1.4018	1.3906	1.3865	1.3902	1.4095
blk_per_min2017	2.1335	2.1335	2.1336	2.1335	2.1335	2.6548	2.5891	2.6287	2.5992	2.6668