Simulated Season

David Liedtka
dliedtka@stanford.edu

1 Introduction

I have always been interested in the way that statistics quantify athletic performance, particularly in the United States' three largest professional sports leagues, the National Football League, Major League Baseball, and the National Basketball Association. In recent years, the emphasis on advanced analytics and the use of machine learning has been most visible in basketball in my opinion, with the rise of controversial figures in the NBA such as Sam Hinkie, currently a guest lecturer at the Stanford Graduate School of Business. Many advanced metrics exist today, such as win shares or value over replacement player, that quantify a player's performance much more completely than simple stats that have been relied upon heavily in the past, such as points or rebounds per game. However, these metrics quantify a players past performance. For my project, I wanted to see if I could effectively predict a player's future performance by projecting a variety of statistics. Additionally, because in the NBA what ultimately matters most are wins and losses, I wanted to see if I could use those projections to better predict game outcomes.

2 Task Definition

Broadly, my project involves two goals:

1. Statistical projection
2. Game outcome prediction

I sought initially to frame the first goal, statistical projection, as a regression problem using stochastic gradient descent using a multitude of features, including past performance data for each player. However, this problem is more complex than it would appear, as training is dependent on each individual player. I eventually broke this goal down into two sub-tasks, predictions for veteran players and prediction for rookie players. For a veteran player, with many seasons of NBA experience, past performance is perhaps the best indicator of future performance. For a rookie player, playing in his first professional season, we do not have past performance data to rely on for training, creating a dependence on less telling characteristics, such as height and weight or college statistics, which can be unreliable. Even more challenging, many players come from an international background and do not even have American college basketball experience for our consideration. Thus, for veteran players, I perform a standard regression using stochastic gradient descent to minimize squared loss, but for rookie players I implement the K-nearest neighbors algorithm to find several players from the past that the rookie is expected to perform similarly to. The second outcome, game

prediction, is a more straightforward task that I formulated as a classification problem using support vector machines.

Game prediction is difficult, particularly in the NBA, because (according to extremes in winning percentage) the "better" team does in fact win more often than in sports such as baseball, but "upsets" (when the team that is not expected to win does win) still occur. If you develop a probabilistic model that says, for example, Team A will beat Team B 51% of the time, then the optimal prediction is to choose Team A to win 100% of the time unless you are able to develop intuition about when an upset is likely to occur. I sought to develop this intuition by including features in my classification that encompassed wear and tear on a team and individual position match-ups with the opponent.

3 Infrastructure

As my project is a task in machine learning, I first needed to collect data for training, validation, and testing. Basketball Reference (basketball-reference.com) contains comprehensive data relevant to both of my goals of statistical projection and game outcome prediction. To compile this data, I built a web scraper and used regular expression matching to download the "totals" and "advanced" career statistics for every player that played in an NBA game in the 2013-14, 2014-15, 2015-16, 2016-17, and 2017-18 seasons. This meant, for example, that for a player like LeBron James, who first appeared in the 2003-04 season and is still an active players, I compiled each of his season statistics from 2003-04 to 2017-18 into my dataset. I also downloaded each player's height, weight, handedness, when they were selected in the NBA draft, teams played for each season, position played each season, and the player's career college statistics if they attended American college. For a full list of statistics that I downloaded, view the statistics that I reported results on in the Appendix. Additionally, to enable game prediction, I used the web scraper to compile each team's regular season schedule from the 2014-15, 2015-16, 2016-17, and 2017-18 seasons. Additionally, I tracked which team was home for each game, which team was away, the date the game occurred, and which team won.

Having compiled the data, I experimented with normalizing the statistical data in a number of ways. First, I normalized all statistics that were not already rates on a per minute basis. For example, rather than storing how many points a player scored in a season, I stored his points scored per minute played. The only statistics I did not normalize were already rates (such as minutes per game or win shares per 48 minutes) or did not make sense to normalize (such as games played or minutes played). Next I experimented with removing outliers. A player who appeared only very briefly in a season and recorded one very positive (or one very negative) play could have an impact on statistics. For example, if a player appeared in only one minute of one game all year but made his only shot, his shooting percentage of 1.000% could lead him to be artificially overrated. Thus, for each player who appeared in less than ten minutes of game action in a season, I zeroed out all statistics. Finally, I experimented by scaling each statistic by different ranges (I would undo this scaling before computing prediction error). I tried leaving statistics as they were, scaling

from -1 to 1 (meaning out of all players in the dataset, the player with the lowest points per game over the course of a season would be scaled to -1, and the player with the highest would be scaled to 1), and scaling from 0 to 1. Interestingly, I found that removing outliers and the type of scaling I used did not have a significant impact on any results, so by default I removed outliers and scaled from -1 to 1.

4 Approach

4.1 Statistical Projection

Having decided to model statistical projection as a regression problem, I broke the task down into two sub-tasks, using k-nearest neighbors for rookie players and linear regression for veteran players. Before implementing these models, I developed a baseline and an oracle. A simplified approach for evaluating players is to judge them based on their past statistics, so for my baseline for veterans for predicting 2017-18 season statistics, I predict using their 2016-17 statistics (the previous season) exactly. For rookie players, as a previous season is unavailable, I predict each player's statistics as an average of every non-rookie player's (in my dataset) rookie season statistics, meaning each rookie receives the same prediction. As my oracle, training on all data from seasons prior to 2017-18, I develop a linear regression model for each statistic with each other statistic as a feature. For example, when testing, to predict a player's points per minute in the 2017-18 season, I use all other statistics as features for that season, such as the players rebounds per minute, win shares per minute, and shooting percentage. Clearly, this type of prediction is at an advantage because if you are trying to predict statistics for a season that has yet to occur, you will not have access to a partial set of statistics from that future season. I use the same oracle for rookies and veterans. The baseline and oracle results can be found with the rest of the results in the appendix.

For rookies, I implement K-nearest neighbors. Using normalized features such as height, weight, handedness, draft position, rookie age, position played, college attended (if applicable), and a weighted average of college statistics (if applicable; accomplished by averaging statistics over all years played in college, weighing each more recent season twice as much as the previous), for each rookie player I find the eight non-rookie players with the closest feature-set Euclidean distance. Then, weighted by each of the eight players' Euclidean distance from the rookie, I predict the rookie's statistics using a weighted average of the neighbors' rookie seasons. As there is no "training" that occurs here, I do not measure training error, but I do record validation error. I run ten trials, each time choosing a different random set of 10% of players to serve as a validation set. However, when it came time for testing, as no parameters are learned and I thought it beneficial to have as many options for "neighbors" as possible, I include those in the validation sets as possible "neighbors".

For veterans, I implement stochastic gradient descent, minimizing square loss and using regularization. For each statistic I predict, I train a model on all seasons up to and including the 2015-16 season. I use the same set of features that I used for rookie predictions but with

the addition of a weighted average of past professional season statistics. For example, if a player has played in three seasons prior to the one being predicted, as a feature for his career points per minute, I sum 1/7 times his points per minute in his first season, 2/7 times his total in his second season, and 4/7 times his total in his third season. The intention is to give recent performance the most weight, but also to acknowledge trends in a player's career and to safeguard to a degree from factors such as injuries. For validation, I predict statistics from the 2016-17 season, and for testing I predict 2017-18 statistics. Using the validation set, I experimented with multiple eta and regularization parameters.

For both rookies and veterans, I use root mean square error as the measure of error I attempt to minimize. I compute the root mean square error for each statistic by comparing the predicted 2017-18 statistic for each player to the true 2017-18 statistic. Because the validation set was not randomly generated (except for K-nearest neighbors) and therefore would be the same for each trial, running multiple trials produces identical results. Thus, I ran only one trial for statistical projection.

4.2 Game Outcome Prediction

I implement multiple classification models useful for gaining insight into game prediction. For my baseline, I simply choose a random winner, expecting to choose correctly half of the time. My oracle is more involved and mirrors my implementation. My intuition was that exploiting features such as fatigue and individual players match-ups would make upset prediction easier. Therefore, for my oracle, I used the same features that I will describe below, but I used the true future (2017-18) statistics for predicting futurue (2017-18) game outcomes. Again, baseline and oracle results can be found in the appendix.

"Home-court advantage" is a commonly-discussed phenomena in sports, and it is especially relevant in basketball. I wanted to provide several models that could provide insight into game outcome prediction, so the first I implement simply predicts the home team as the winner in each game.

Next, I wanted to incorporate factors that could encompass team fatigue from game to game. It is often surmised that NBA teams do not perform as well if they have played games in back-to-back nights or have played three games in four nights. Therefore, for this model, I implemented classification using support vector machines (minimizing hinge loss with regularization) and included as features the home team, the away team, the date, whether the home team was playing in the second of games on back to back nights (and the same for the away team), and whether the home team was playing in the third of games in four nights (and the same for the away team). Using the validation set, I experimented with multiple regularization and eta parameters.

Next, to develop some intuition about including features to exploit player match-ups, I include the same fatigue features but also introduce features based on the previous season's statistics. For example, for a game taking place during season n between teams A and B, I first generate a list of each player who played for team A and team B during season n. Then for each position (point guard, shooting guard, small forward, power forward, and center) for each team, I generate a feature for each statistic by taking a minutes played-weighted

average of that statistic (from season $n-1$ for each player at that position. For example, if team A had two players that played point guard in 2017-18, and if player 1 played twice as much as player 2 during 2016-17, then the feature for team A's point guard points per minute in 2017-18 would be 2/3 times player 1's 2016-17 points per game plus 1/3 times player 2's points per game in 2016-17. This model can be viewed as another baseline, providing a comparison to develop intuition about the value of using projected statistics rather than past statistics.

Finally, I repeated implementing the model described above, but I used projected statistics for 2017-18 and actual statistics for all previous seasons rather than the previous season's statistics. I used my projected statistics from statistical projection as input features for these outcome predictions.

My original plan was to use all seasons prior to 2016-17 as training data, all 2016-17 seasons as a validation set, and all 2017-18 seasons as the test set, as I did with statistical projection. However, because game prediction included features based on what teams were playing, I thought it would be beneficial to train on outcomes that occurred most recently compared to the season I would eventually try to predict, 2017-18. Thus, instead of using 2016-17 outcomes as a validation set, I used a random sample (of 10%) of games over the 2014-15, 2015-16, and 2016-17 seasons as the validation set.

For each of these models, I used accuracy (number of games predicted correctly out of number of games predicted total) as my measure of accuracy that I attempted to maximize. For each model, I ran 100 trials (differing randomly generated validation sets lead to different results for each trial).

5 Literature Review

Applying machine learning to predict sports outcomes or statistics has become increasingly popular, especially with the advent of fantasy sports and the rise in popularity of sports betting. Porter suggests viewing the progression of career statistics on a year-to-year basis as a time series, and ARIMA to make fantasy football predictions[2]. It is intuitive to view season-by-season statistical projection as a time series. Hermann and Ntoso formulated the problem differently to optimize winnings on DraftKings, using regression for game prediction rather than classification[1]. Wheeler attempts a players points scored prediction that he ties into game outcome prediction, a sentiment similar to my attempts to predict game outcomes using statistics, but I use a wider set of statistics as features[3].

These previous studies provide valuable insight into past approaches towards similar problems, but my task differs from each of these studies in a variety of ways.

6 Results

A full presentation of statistical projections root mean square error, including comparison to the baseline and oracle, is available in Appendix A.1.

The following tables summarizes the accuracy of the different game prediction models:

Table 1: Game prediction accuracy for different models over 100 trials.

	Accuracy		
Model	**Training**	**Validation**	**Testing**
Random (Baseline)	0.5008	0.4994	0.5013
Home	0.5773	0.6284	0.5788
SVM-F	0.5625	0.5464	0.5138
SVM-F,LP	0.6311	0.6912	0.6186
SVM-F,PP	0.6771	0.6994	0.4512
SVM-F,TP (Oracle)	0.6771	0.6994	0.6634

Where "Random" is the model that chooses a winner randomly, "Home" always chooses the home team, "SVM-F" is the model using support vector machines and team fatigue features, "SVM-F,LP" uses support vector machines with last year's player statistics in addition to team fatigue features, "SVM-F,PP" uses projected 2017-18 statistics, and "SVM-TP" uses the true 2017-18 statistics.

7 Analysis

7.1 Statistical Projections

My results prove that both goals, statistical projection and game outcome prediction are difficult tasks to accomplish. For statistical projection, in Tables 3 and 4, in all cases we see that the oracle outperforms our models, and usually does so substantially. Still, from a practical purpose, future results are never readily available. Therefore, compared to the alternative of using the previous season's statistics (the veteran baseline) or averaging multiple players' performances (the rookie baseline), our projections are valuable if they outperform the baseline.

We find generally mixed results when it comes to outperforming the baseline. For rookie results, our test results outperform the baseline in 53.1% of measured statistics. For veteran results, however, out test results outperform the baseline in only 12.8% of measured statistics.

After initially experiencing overfitting, regularization was added to veteran projections, and it appears to have worked, as there is no discernible pattern between veteran training, validation, and test error. However, the validation error for K-nearest neighbors is consistently and noticeably lower than the test accuracy. This is likely because ten trials were run measuring validation accuracy for rookies, using a random subset of players who were not rookies in 2017-18 as the validation set every time. Meanwhile, for testing the set of rookies in 2017-18 was used. The 2017-18 rookie class was wildly unpredictable, with Ben Simmons, Jayson Tatum, and Donovan Mitchell surpassing expectations with players like Markelle Fultz and Josh Jackson falling far short. It is possible that the set of potential "neighbor" for 2017-18 rookies was not as representative as it was for players who were rookies in any number of seasons before.

Regarding the lack of success in surpassing the baseline for veteran results, using the previous year's statistics is a simple strategy, but it is a relatively effective strategy. Most players tend to perform similarly over the course of their careers, but every year a few players rapidly ascend or tail off in performance. Using linear regression, we hoped to capture those trends. This largely failed for a number of reasons.

First and foremost, the use of a linear predictor for such a complex problem is not ideal. Linear prediction limited my use of past performance, as I had to maintain identical feature sets for each player. This led to me needing to use a weighted average of career statistics rather than having features tied to each season. For example, I could not include a set of features for a player's third career season, because then a player who had only played two seasons would be interpreted as having recorded statistics of zero throughout a third season they had yet to play. I briefly considered creating different models depending on how many seasons a player had played (i.e.- training on all players entering their second season, then all players entering their third season, etc.), but I thought this would create dependencies on lesser sets of training data (not many players have played 10 seasons, for example). For these reasons, if I continue working on this problem in the future, I would revamp my infrastructure to address this problem as a time series problem.

Additionally, the structure of my validation set may have been problematic. Because I used all 2016-17 seasons as validation data, there was at least a two year disconnect between each of my training and testing samples. Basketball is a game in which play style changes dramatically, and this gap between training and testing may have accounted for some degree of inaccuracy. Looking back, I may have been better off structuring my validation data similar to my validation data for rookie players, using a random subset of all seasons instead of all seasons from 2016-17.

7.2 Game Outcome Prediction

In Table 1, our baseline performs as expected, predicting correctly half of the time. Interestingly, we notice that choosing the home team is a simple but relatively well-performing method of outcome prediction.

Beginning with our support vector machines, SVM-F performs relatively well for training but under-performs for testing, despite the use of regularization. While the intention of SVM-F was to exploit possible team fatigue, the model instead relied more heavily (by virtue of feature weights) on which teams were playing. Because training occurred on outcomes dispersed among three seasons, it is likely that those weights were not as dependable when it came time to predict the 2017-18 season, as none of the 2017-18 games were in the training data. Again, the NBA is a league that changes rapidly, and our training failed to capture that.

The results for SVM-F,LP and SVM-F,TP were encouraging, but not initially so. Originally, I fed features related to every collected statistic into the models. This was clearly an example of using too many features, as prediction accuracy fell under 50% in both cases. However, I later decided only to include features related to win shares and value over replacement player, two popular advanced analytical statistics. With this change, SVM-F,LP

showed that even using the previous years statistics could lead to a significant rise in prediction accuracy. SVM-F,TP showed that in an ideal scenario, prediction power would increase even more.

My hope was that projected statistics, SVM-F,PP, would find a middle ground between using past statistics and future statistics. However, because my predictions generally were worse than the baseline, the results of SVM-F,PP were disappointing. An additional source of error was likely another disconnect between training and testing. For training, I used true statistics, but for testing I used projected statistics. For consistency, I should have used projected statistics throughout, but this would have required me to have saved my training predictions for statistical projection.

8 Future Work

While my results were mixed, I have identified areas that if addressed could lead to substantial performance increases in the future. First, instead of using linear regression, inspired by Porter, I considered using an auto-regressive moving average (ARMA) model and treating season-by-season data as a time series. At the time, implementation was hamstrung because I thought project requirements prohibited the use of external libraries, and implementing models such as ARMA myself seemed too daunting. However, in the future, treating my data as a time series would allow me to better leverage career data.

Additionally, as I mentioned when testing my support vector machine models, I think my inclusion of too many features may have caused decreases in performance. I considered using scikit-learn's recursive feature elimination, but the structure of my algorithm made late-stage integration infeasible. Ideally, I would have been able to eliminate statistical features into the support vector machines that were not beneficial, instead of going through myself and choosing impactful features such as win shares. I also would have been able to apply this technique to statistical projection, allowing me to eliminate features that were more harmful than beneficial.

9 Conclusion

Although I failed to achieve all my goals to the degree I had hoped, my experimentation in different areas allowed for a number of takeaways. We found K-nearest neighbors to be an effective way to model performance for rookies, a difficult task because historical data does not exist. The implementation of K-nearest neighbors we used was relatively simple and still provided encouraging performance, meaning that refining the implementation could lead to even greater gains.

Statistical projection proved to be too complex a problem for simple linear models, but representing the data as a time series is a source of optimism for future testing. Although projected statistics failed to positively influence game outcome prediction, we saw that true statistics (whether they are from a previous season or from the current season) were sig-

nificantly beneficial. By further refining the features included in outcome prediction and my methods of statistical projection, I hope to see greater prediction accuracy achieved. I still believe projection-based outcome predictions can be an insightful way to look at sports analytics, and it is a problem I look forward to continuing to work on in the future.

Acknowledgments

I would like to thank Professor Percy Liang and Kelvin Guu for leading a thought-provoking and rewarding class, and I would like to thank my project mentor, Jacob Hoffman for his insightful and personalized feedback. I would also like to thank Basketball Reference for compiling an organized and comprehensive set of historical NBA data and making it accessible to the public.

References

[1] Hermann, E., and Ntoso, A. Machine learning applications in fantasy basketball, 2015.

[2] Porter, J. W. Predictive analytics for fantasy football: Predicting player performance across the nfl.

[3] Wheeler, K. Predicting nba player performance.

A Appendix

A.1 Statistical Projection Results

These tables (on the next page) provide measures of root mean square error for statistical projections. Table 3 has information for rookies, while table 4 has information for veterans. Below, each statistic's abbreviation is defined.

Abbreviation	Statistic	Abbreviation	Statistic
2PAPM	2 point attempts per minute	GS	Games started
2PP	2 point percentage	GSPG	Games started per game
2PPM	2 pointers per minute	MP	Minutes played
3PAPM	3 point attempts per minute	MPG	Minutes per game
3PAr	3 point attempt rate	OBPM	Offensive box plus minus
3PP	3 point percentage	ORBP	Offensive rebound percentage
3PPM	3 pointers per minute	ORBPM	Offensive rebounds per minute
ASTP	Assist percentage	OWSPM	Offensive win shares per minute
ASTPM	Assists per minute	PER	Player efficiency rating
BLKP	Block percentage	PFPM	Personal fouls per minute
BLKPM	Blocks per minute	PPM	Points per minute
BPM	Box plus minus	STLP	Steal percentage
DBMP	Defensive box plus minus	STLPM	Steals per minute
DRBP	Defensive rebound percentage	TOVP	Turnover percentage
DBRPM	Defensive rebounds per minute	TOVPM	Turnovers per minute
DWSPM	Defensive win shares per minute	TRBP	Total rebound percentage
FGAPM	Field goal attempts per minute	TRBPM	Total rebounds per minute
FGP	Field goal percentage	TSP	True shooting percentage
FGPM	Field goals per minute	USGP	Usage rate
FTAPM	Free throw attempts per minute	VORP	Value over replacement player
FTP	Free throw percentage	WSP48	Win shares per 48 minutes
FTPM	Free throws per minute	WSPM	Win shares per minute
FTr	Free throw rate	eFGP	Effective field goal percentage
G	Games played		

Table 2: Root mean square error of statistical projection results for rookies.

Rookies	Evaluation Type			
Statistic	**Baseline**	**Validation**	**Test**	**Oracle**
2PAPM	0.1271822	0.0853715	0.1102053	0.0694203
2PP	0.1946147	0.1233054	0.1988918	0.1714309
2PPM	0.0766426	0.0512767	0.0702685	0.0526191
3PAPM	0.1466386	0.0496596	0.1373063	0.0699842
3PAr	0.2794030	0.1414404	0.2195793	0.1492821
3PP	0.1906027	0.1454785	0.1726784	0.1504758
3PPM	0.0940896	0.0180304	0.0925701	0.0568467
ASTP	8.9039544	6.9212909	7.7787120	2.0827287
ASTPM	0.0595627	0.0390106	0.0515991	0.0190485
BLKP	1.9794341	2.0075799	1.7763658	1.1247150
BLKPM	0.0228316	0.0256372	0.0202953	0.0106629
BPM	8.7912321	4.3037364	9.0176848	5.8715883
DBPM	3.3867500	2.2747163	3.3372224	1.5917227
DRBP	11.041144	6.7298811	12.000342	4.4509754
DRBPM	0.1061800	0.0609821	0.1159320	0.0455188
DWSPM	0.0006130	0.0005876	0.0006288	0.0005123
FGAPM	0.1432276	0.0899041	0.1493192	0.0409580
FGP	0.1503975	0.1121458	0.1561641	0.0550816
FGPM	0.1056569	0.0509165	0.1098624	0.0397406
FTAPM	0.0484582	0.0520891	0.0417629	0.0267278
FTP	0.3069334	0.2193311	0.2954026	0.2411895
FTPM	0.0343632	0.0370646	0.0309403	0.0186353
FTr	0.1648663	0.1952504	0.1594535	0.1374298
G	29.622986	23.259968	24.057657	14.002168
GS	20.023374	20.587360	16.901033	6.9376808
GSPG	0.2875709	0.2890411	0.2559861	0.1179766
MP	739.98331	679.92364	566.58426	201.10051
MPG	8.7570880	7.8488205	7.5465832	5.0445241
OBPM	8.1605222	3.4112209	8.3592637	4.5871242
ORBP	5.0699508	3.5259134	4.1896664	4.3683680
ORBPM	0.0450752	0.0310175	0.0370074	0.0402946
OWSPM	0.0093863	0.0019046	0.0094029	0.0071993
PER	13.714340	5.9554850	13.901074	7.9073441
PFPM	0.0722107	0.0633766	0.0706218	0.0754276
PPM	0.2852356	0.1187052	0.2947476	0.1259680
STLP	1.4830499	0.9937589	1.5055381	0.5735240
STLPM	0.0297845	0.0193033	0.0302545	0.0125445
TOVP	7.1790493	7.1789439	7.3680450	5.3070822
TOVPM	0.0407013	0.0296250	0.0387871	0.0261915
TRBP	6.7991677	4.2162519	6.8480785	1.5977862
TRBPM	0.1216386	0.0746141	0.1224897	0.0326993
TSP	0.1723815	0.1157552	0.1825699	0.0540597
USGP	7.0959000	4.7224607	7.1254070	2.5184761
VORP	0.6455558	0.7788045	0.6401059	0.7880474
WSP48	0.2803634	0.0919097	0.2811465	0.1768436
WSPM	0.0093179	0.0019398	0.0093852	0.0073103
eFGP	0.1746248	0.1172719	0.1886646	0.0489920

Table 3: Root mean square error of statistical projection results for veterans.

Veterans	Evaluation Type				
Statistic	**Baseline**	**Training**	**Validation**	**Test**	**Oracle**
2PAPM	0.0750096	0.0587292	0.0735368	0.0757096	0.0315622
2PP	0.1192205	0.0873232	0.1236591	0.1262712	0.0939476
2PPM	0.0330569	0.0378121	0.0585330	0.0336728	0.0211716
3PAPM	0.0447230	0.0299724	0.0366667	0.0508383	0.0290293
3PAr	0.1263996	0.0866152	0.0948589	0.1411627	0.0876196
3PP	0.1463931	0.1466732	0.1327629	0.1455416	0.1142450
3PPM	0.0210366	0.0139880	0.0177344	0.0232544	0.0167801
ASTP	5.4245708	4.4136420	4.9909833	5.9047276	1.5063530
ASTPM	0.0340482	0.0270521	0.0322467	0.0376707	0.0174221
BLKP	0.8550122	0.9511799	1.2180082	0.9260505	0.7388880
BLKPM	0.0103499	0.0122863	0.0153432	0.0111023	0.0072530
BPM	4.1469622	3.1691736	3.4190071	4.3036611	1.9647486
DBPM	1.8598675	1.5373425	1.9664839	1.9656078	0.9960145
DRBP	5.8442759	3.7147716	3.8067021	5.9581942	2.1710099
DRBPM	0.0564616	0.0381099	0.0339158	0.0570095	0.0190879
DWSPM	0.0004844	0.0004650	0.0004860	0.0005140	0.0003538
FGAPM	0.0762289	0.0613936	0.0787891	0.0816012	0.0277220
FGP	0.0924126	0.0741414	0.0892850	0.0982641	0.0364519
FGPM	0.0381748	0.0388822	0.0593331	0.0397320	0.0148375
FTAPM	0.0423525	0.0360058	0.0387168	0.0443252	0.0207639
FTP	0.2046918	0.1507218	0.1849054	0.2196255	0.1715088
FTPM	0.0334023	0.0267327	0.0305434	0.0371491	0.0189899
FTr	0.2803156	0.1655936	0.1393127	0.2861662	0.2383455
G	23.871467	21.060589	20.331142	23.794483	10.204553
GS	24.677925	23.985940	21.319717	22.273573	7.5045891
GSPG	0.3271101	0.3083840	0.2978070	0.2998471	0.1179909
MP	686.88900	656.31637	572.06215	647.01905	148.67105
MPG	6.1933996	5.7979847	5.4845185	6.2782114	2.8692306
OBPM	3.4675242	2.7815793	3.6429275	3.6342035	1.4925360
ORBP	2.2132313	3.3765336	5.3354179	2.6589175	1.8569007
ORBPM	0.0199487	0.0325654	0.0526570	0.0239066	0.0164142
OWSPM	0.0013526	0.0014585	0.0013019	0.0017692	0.0014169
PER	5.4179313	5.0377285	7.3385562	5.5701364	2.2747830
PFPM	0.0302994	0.0334963	0.0325314	0.0358807	0.0351431
PPM	0.1016158	0.0921533	0.1299677	0.1115868	0.0327425
STLP	0.7924065	0.6429261	0.8305095	0.9744914	0.2845687
STLPM	0.0158309	0.0126099	0.0166845	0.0193595	0.0068952
TOVP	4.6926668	4.7566650	4.6844650	6.3726775	4.2592407
TOVPM	0.0199476	0.0236505	0.0197564	0.0259002	0.0128672
TRBP	3.3449512	2.6329178	3.5421741	3.3710199	0.8700354
TRBPM	0.0599710	0.0489876	0.0589802	0.0605516	0.0234744
TSP	0.1109419	0.0763429	0.0880070	0.1191824	0.0446556
USGP	3.8851923	3.3544472	4.0760959	4.0968989	1.3018100
VORP	1.0380392	1.0385961	0.9048020	0.9632704	0.7988298
WSP48	0.0927613	0.0863651	0.1250682	0.0967065	0.0409234
WSPM	0.0013395	0.0015363	0.0014712	0.0016757	0.0012436
eFGP	0.1117355	0.0806755	0.0977359	0.1197191	0.0430093