RPP Technical Writeup

Ian Padovani

The algorithm and theory behind it:

RPP stands for Relative Performance Potential and can be broken into two sub categories ROP (Relative Offensive Potential) and RDP (Relative Defense Potential). Since every competition is different, a universal value can not be applied nor can this be used for a universal ranking.

The algorithm is based on the idea of shown potential and consistency. At a competition, the robot will perform at a given level and steadily improve or worsen as the competition goes on. This is due to comfort levels and design reliability changing as time goes on. The idea is that at some point after enough matches, an approximation could be made regarding the relative performance of the robot. A universal approximation of potential would take too many matches to be accurate and is impossible considering a given robot competes against a fraction of the total bots. In order to make a good approximation the robot needs to play with the majority of other robots as the reason will be explained later.

ROP is to account and measure the offensive potential of the robot based on points scored and the consistency of those points. Each robot will have a range of performance that can be categorized by a normal distribution assuming no major outliers. Major outliers define themselves and refer to events that are unforeseen and are non-repeatable extremes. For example a robot that gets heavily defensed one round and scores minimal points is a major outlier. The drive team will learn what happened and not let themselves do as poorly the second time even if they got defensed the exact same amount. By taking into account major outliers and recognizing them as such we can approximate more effectively and even branch into match predictions. This all put together is how we account for human variation in ability match to match. That being said, ROP is different per game as rules and point values change. The general formula is as follows:

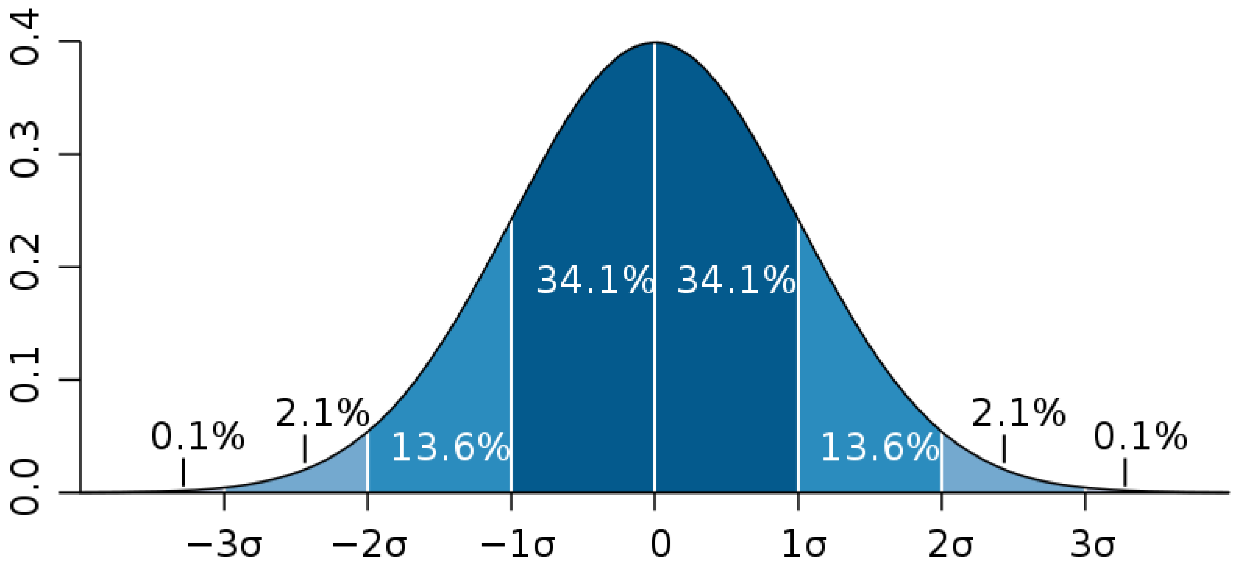
$$ROP = \frac{r_{n\,avg}}{r_{max\,avg}}$$

$$r_{n\,avg} = Robot\ \#n\ point\ range\ avg = \left(\frac{P_{n\,high} - P_{n\,low}}{2}\right) + P_{n\,low}$$

$$r_{max\,avg} = Highest\ point\ range\ avg\ at\ comp$$

Point ranges are what RPP, ROP, and RDP are all based on. The idea is to define a range or what is reasonably possible for a given robot to accomplish in a random match ignoring major outliers. The next section will explain why these are more complex than simply a high and low.

In a given match a robot fires a ball at a target. They either miss or score with no other outcome. So this means that you can record both the number scored and number missed. To expand the data further you can record these two metrics for every match that robot plays. If you graph the points scored according to the equation of a bell curve you will get something similar to this:



This is a graph of a normal distribution. A normal distribution simply states that a given datapoint will fall somewhere in this range with a great percentage close to the middle. The equation for this curve is as follows below:

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

$$\sigma = standard\ deviation = \sqrt{\frac{\Sigma(x_i-\mu)^2}{N}}$$

$$x_i = each\ value\ in\ a\ population$$

$$\mu = the\ average\ of\ a\ population$$

$$N = the\ size\ of\ the\ population$$

With this newly expanded data you can make predictions about robot performance with two major limitations. You will only be able to predict performance for matches that have no outside interference or dependence on non-consistent variables. These non-consistent variables are events such as a different robot ally every run or a different opponent every match. Additionally, recording scored and missed becomes tiring for the people scouting. On larger teams this can be fixed but on smaller teams like my own this is unreasonable. Simplifying it down to just scored can accomplish very similar results and in testing made no noticeable difference. This is realized because a mechanically inaccurate scoring mechanism will always be mechanically inaccurate. Even if they pull off a match of high accuracy at its root it is still inaccurate. However we can not stop there, you need to expand your data further for it to actually be useful in predictions.

To expand it further we first set a constant environment. This is an environment that has constant rules that never change. Doing so enables you to make more accurate guesses and cut down on instances of randomness (Even still this method will yield a ~52% confidence but is comparably better than other methods). The rules for an example FRC environment are set below:

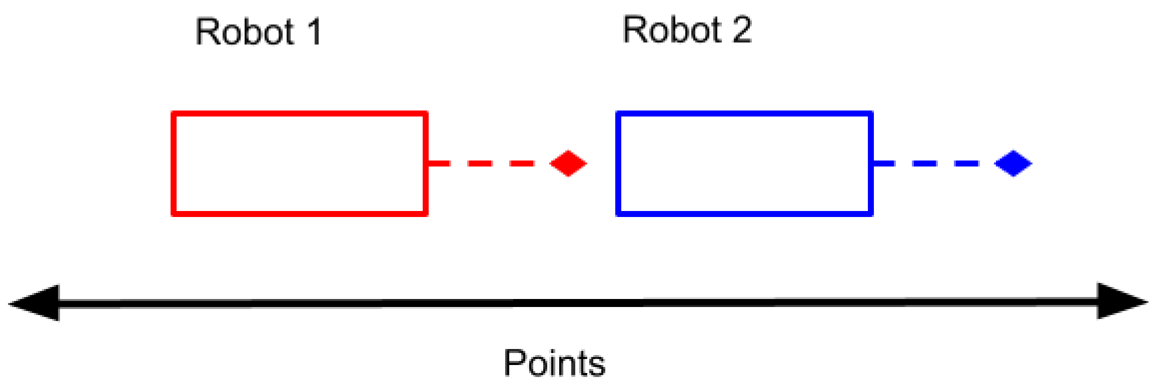
1.) There is a set list of teams that get rotated through such that every robot plays the vast majority of other robots competing.
2.) This list does not change under any circumstance once the competition has started.
3.) The drivers will be of roughly consistent caliber every match even if they change every match. Since they constantly change we assume they all get similar levels of practice and as such will all perform similarly.
4.) The game rules, point values, penalty values, and size never change throughout the competition and will be consistent for each and every robot.
5.) Referees will score with no bias and 100% accuracy all in accordance to the rules.
6.) Scouters input data with no bias and 100% accuracy all in accordance to the rules.
7.) Time fatigue does not exist and drivers will perform the same any number of times in a row due to assumed similar adrenaline levels match to match for the sake of uniformity. (More adrenaline does not always mean drivers score better so this rule works out)
8.) The spreadsheet will accurately calculate values with no error each and every time.

Now that we have set our environment we can begin to expand our data. Before this expansion we only had every ball scored for every match played. Now we do the same but for every robot that competes. Moreover, we collect the same or similar info for all other point scoring means and observe defense. Defense needs to be observed due to robot on robot contact being inevitable.

To observe and collect data regarding defense it becomes rather tricky. Defense does not inherently score points but instead denies the acquisition rate of points. While we could note down who defensed who, this becomes arduous in that one robot could defense multiple other robots. Instead, we can simply form a matrix with the list of teams on both the columns and the rows. The intersection of two points is an instance of the column team playing against the row team. Then the point delta value is calculated for that match where we find how much the row robot was varied from their average. This value tells us how effective defense, if any, was at reducing the point acquisition rate of the opposing robot. Doing this means that we can catch a robot playing a perceived "good" defense but could be realistically doing nothing for the score.

This is much simpler to record and is fully independent of scouters further reducing the load for smaller teams. Even this however has issues as defense does not impact one robot. Defense impacts the entire alliance as it ruins flow, causes stress, and changes strategies. So instead the metric will be based on the avg point delta denied on the opposing alliance every match. The lack of any defense will be shown and negatively affect the team's RDP. While this can be seen as encouraging aggressive play, I see it as the algorithm recognizing the lack of good strategy.

Now with this new expansion of our data we can make some pretty interesting observations and calculations in order to rank robots. To visualize how we will do this observe the pictures below:



Each robot will have a range that they will reasonably score in a match. For FRC, a safe percentage is dropping a team’s worst match since it can be put off to being an outlier and is a way of giving benefit of the doubt to teams. This is better than cutting off their best simply because if they were able to do that then they are able to do it again even if the chances are low. These reasonable score ranges are the solid boxes. The extra dotted line is the theoretical max of the team relative to shown performance. An example of this is if in your best match you miss some balls, this dotted line represents a carbon copy match where you don’t miss those balls. It is important to note that this metric can only be done if balls missed were being recorded.

These ranges overlap with each other going robot to robot allowing you to make predictions and comparisons in robot performance. Keep in mind that even though the ranges are depicted as rectangles they still observe the normal distribution pattern shown earlier. Meaning in 100 matches you will not score in the far right side of the box more than you will in the middle. In fact, in a 100 matches you will most likely end up close to the middle roughly 66% of the time. The use of rectangles is to simplify the visuals for understanding.

Now that all of this is discussed we can break down how to calculate the other two metrics as they are dependent on this expansion of data.

RDP is calculated simply by looking at how many points the team is denying the other team on average relative to the team with the highest point scoring value. This is similar to ROP in calculation and means the same thing in essence. You end up with an average in a range with a high and low. Note that if a robot never plays defense but still receives penalties they can get a negative RDP. Another good note is even if a robot plays defense but gets enough penalties to negate the effectiveness of the defense they can still have a 0 or negative RDP which shows they are not favorable at playing defense.

$$RDP = \frac{d_{n\ avg}}{r_{max\ avg}}$$

$$d_{n\ avg} = Robot\ \#n\ point\ denial\ range\ avg = \left(\frac{D_{n\ high} - D_{n\ low}}{2}\right) + D_{n\ low} - p$$

$$p = avg\ penalties\ received$$

Now that we have both ROP and RDP the last metric is RPP. RPP is a bit more strange as it is an overall rating and talks about how much potential a team shows in performance relative to other teams at the competition. Adding ROP and RDP will tell you the combined effectiveness of the robot's performance even taking into account penalties. Dividing this by the highest robot's combined ROP and RDP will tell you a robot's RPP. This is shown below:

$$RPP = \frac{ROP_n + RDP_n}{ROP_{max} + RDP_{max}}$$

$$ROP_n = Robot\ \#n\ ROP$$

$$ROP_{max} = Highest\ ROP\ at\ comp$$

$$RDP_n = Robot\ \#n\ RDP$$

$$RDP_{max} = Highest\ RDPat\ comp$$

Now that we have all three of our metrics we can discuss how to interpret them. The closer any of these values are to 1 the better the robot is. No robot will ever have a value higher than 1, as a value of 1 signifies that they are the best at the competition.

Summary of Inputs:
- Individual points scored per robot (6 people)
    - Does not include team based things or events
    - Penalties Scored
- Managing spreadsheet and correcting data (1 person)

Overall this method takes a minimum of 6 people however more is recommended so that people can be rotated out and take breaks. Truthfully it is a very boring task and after a few hours can be mentally draining.

Data collection will be done in a google form that is attached to the spreadsheet. This keeps it simple and familiar for all team members. Internet connection is not always available at competitions and in the event that this is the case, paper scouting should be done and the spreadsheet filled out when internet is available.

Data processing will be handled by a single person who during matches will take the data from the previous match and put it into the correct rows and columns for the algorithm to do its thing. This person will also be responsible for observing the data and checking for errors or faults in the algorithm. With how tired scouting people can get, this becomes a full time job.

This will all be done in google sheets so that the human player can access it and strategize depending on teams in upcoming matches and so it can be shared among a ton of people.

How match predictions work:

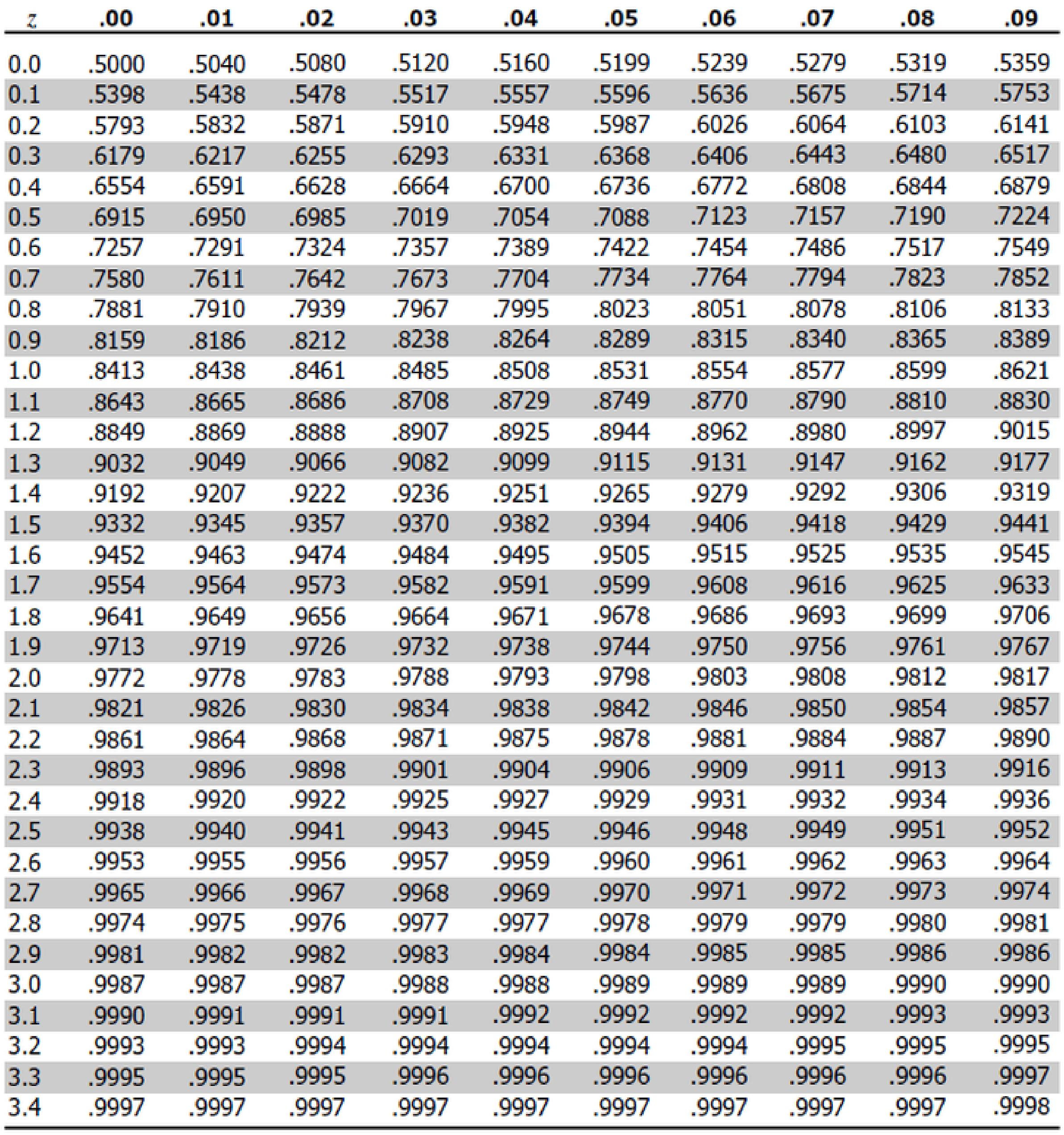
Match predictions will be done by adding all bounds of the ROP with all members of the same team and subtracting these bounds by the bounds of the added RDP of the opposing team. This means you will have a point max and point minimum in the range of reasonable scores accounting for defense options. You will also get an average point score that you can expect them to score most of the time. Compare this range to the opposing team's range and find the overlap. Using the properties of a normal distribution, you can calculate a win chance.

<u>Win%:</u>
Assume the overlap has values that follow a normal distribution. There exists a property called z-score that will be very important to us for calculating win chances. Z-score is calculated by the following equation:

$$z = \frac{x - \mu}{\sigma}$$

$$x = point\ value\ we\ are\ calculating\ for$$

This z-score makes the point values something we can turn into percentages that follow a normal distribution. To turn them into percentages we normally use a table like the one below. However since we are using spreadsheets, we use the command NORMSDIST which does the hard work for us.

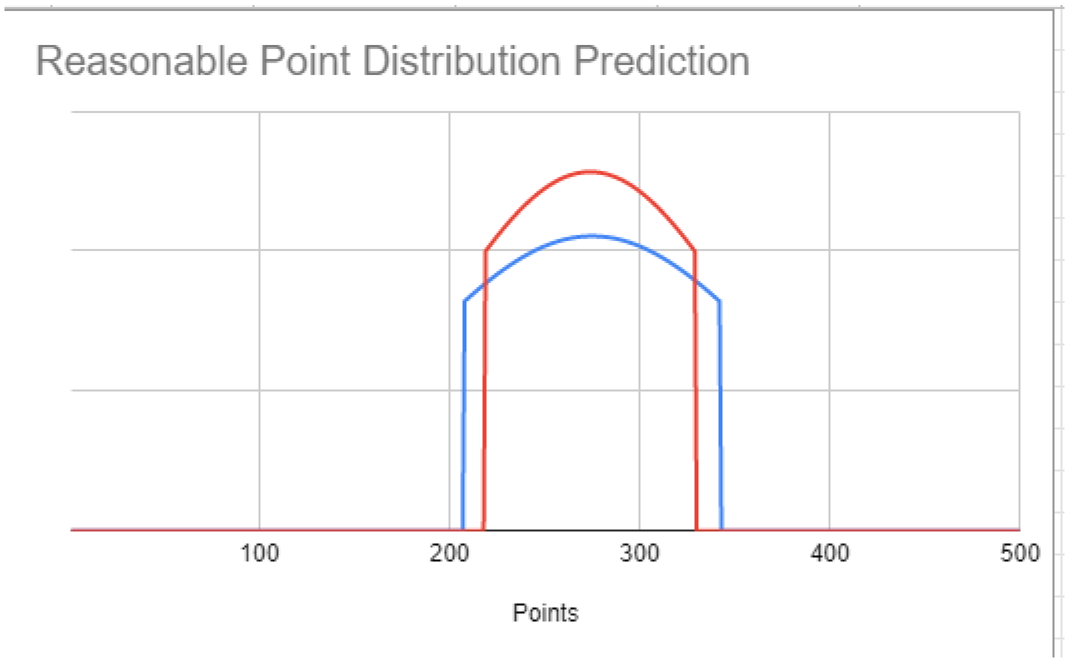


z	.00	.01	.02	.03	.04	.05	.06	.07	.08	.09
0.0	.5000	.5040	.5080	.5120	.5160	.5199	.5239	.5279	.5319	.5359
0.1	.5398	.5438	.5478	.5517	.5557	.5596	.5636	.5675	.5714	.5753
0.2	.5793	.5832	.5871	.5910	.5948	.5987	.6026	.6064	.6103	.6141
0.3	.6179	.6217	.6255	.6293	.6331	.6368	.6406	.6443	.6480	.6517
0.4	.6554	.6591	.6628	.6664	.6700	.6736	.6772	.6808	.6844	.6879
0.5	.6915	.6950	.6985	.7019	.7054	.7088	.7123	.7157	.7190	.7224
0.6	.7257	.7291	.7324	.7357	.7389	.7422	.7454	.7486	.7517	.7549
0.7	.7580	.7611	.7642	.7673	.7704	.7734	.7764	.7794	.7823	.7852
0.8	.7881	.7910	.7939	.7967	.7995	.8023	.8051	.8078	.8106	.8133
0.9	.8159	.8186	.8212	.8238	.8264	.8289	.8315	.8340	.8365	.8389
1.0	.8413	.8438	.8461	.8485	.8508	.8531	.8554	.8577	.8599	.8621
1.1	.8643	.8665	.8686	.8708	.8729	.8749	.8770	.8790	.8810	.8830
1.2	.8849	.8869	.8888	.8907	.8925	.8944	.8962	.8980	.8997	.9015
1.3	.9032	.9049	.9066	.9082	.9099	.9115	.9131	.9147	.9162	.9177
1.4	.9192	.9207	.9222	.9236	.9251	.9265	.9279	.9292	.9306	.9319
1.5	.9332	.9345	.9357	.9370	.9382	.9394	.9406	.9418	.9429	.9441
1.6	.9452	.9463	.9474	.9484	.9495	.9505	.9515	.9525	.9535	.9545
1.7	.9554	.9564	.9573	.9582	.9591	.9599	.9608	.9616	.9625	.9633
1.8	.9641	.9649	.9656	.9664	.9671	.9678	.9686	.9693	.9699	.9706
1.9	.9713	.9719	.9726	.9732	.9738	.9744	.9750	.9756	.9761	.9767
2.0	.9772	.9778	.9783	.9788	.9793	.9798	.9803	.9808	.9812	.9817
2.1	.9821	.9826	.9830	.9834	.9838	.9842	.9846	.9850	.9854	.9857
2.2	.9861	.9864	.9868	.9871	.9875	.9878	.9881	.9884	.9887	.9890
2.3	.9893	.9896	.9898	.9901	.9904	.9906	.9909	.9911	.9913	.9916
2.4	.9918	.9920	.9922	.9925	.9927	.9929	.9931	.9932	.9934	.9936
2.5	.9938	.9940	.9941	.9943	.9945	.9946	.9948	.9949	.9951	.9952
2.6	.9953	.9955	.9956	.9957	.9959	.9960	.9961	.9962	.9963	.9964
2.7	.9965	.9966	.9967	.9968	.9969	.9970	.9971	.9972	.9973	.9974
2.8	.9974	.9975	.9976	.9977	.9977	.9978	.9979	.9979	.9980	.9981
2.9	.9981	.9982	.9982	.9983	.9984	.9984	.9985	.9985	.9986	.9986
3.0	.9987	.9987	.9987	.9988	.9988	.9989	.9989	.9989	.9990	.9990
3.1	.9990	.9991	.9991	.9991	.9992	.9992	.9992	.9992	.9993	.9993
3.2	.9993	.9993	.9994	.9994	.9994	.9994	.9994	.9995	.9995	.9995
3.3	.9995	.9995	.9995	.9996	.9996	.9996	.9996	.9996	.9996	.9997
3.4	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9997	.9998

Looking at the table you can see how much work it would be to go through manually and find all of the z-score percentages you need. Once you have all of these percentages you have to interpret them a certain way to make sense.

For simplicity we will call the percentages you get from the z-score P1 and P2 with P1 being the larger of the two. If you wanted to find the percentage chance of the red team scoring a point value in between say 100 and 200 you start by finding the associated percentage with each point value using z-score. 200 will have the larger percentage so it will be the point value associated with P1 and 100 will be for P2. For simple numbers we will say P1 equals 0.5 and P2 equals 0.125. We now have everything we need to find what percent chance the red team has to score a point value between 100 and 200. Simply subtract P1 from P2 and multiply by 100. In our case that would be 0.375 or 37.5% chance of the red team scoring between 100 and 200 points.

This basic principle is what is used to calculate the win percent of a team. To start calculating it, we find the definite loss and win percent chance for both red and blue teams. These bounds are defined as areas where there is no overlap and thus no reasonable chance for a different result. For example on the graph below:



From the graph you can see each team's bounds for how many points they can reasonably be predicted to get. (We got this from the bell curve equation mentioned a while back). We can observe that the blue team has both the lowest and highest bound when it comes to points, so to find our definite loss percent we need to use our P1 and P2 method from earlier. To calculate definite loss percent we want to find the probability blue will score in the zone between its lower bound and red's lower bound. So P1 would be the lower bound of red and P2 would be the lower bound of blue. Since red does not have a range of points where they will outright lose, their definite loss percent is 0. So for simple variables to remember for later DLB and DLR will be placeholders for each respective loss percentage.

To calculate definite win percent, it is the same as before but on the other end of the graph. For this graph blue has the higher bound so P1 is blue’s maximum bound and P2 is red’s maximum bound. Rinse and repeat what you did for definite loss and you get DWB and DWR.

Now to find the probability of the inside it gets strange, for uniformity we will call this region the upset region. This is the region where both blue and red exist. A strange property we can take advantage of is how the equations are set up. We can logically take the largest upset bound (in this case the maximum red bound) and subtract its percentage value from the smallest upset bound (in this case the minimum red bound). However, the P1 and P2 values change depending on which team you are looking at since the red team and blue team have different bounds from each other. So let's break down this problem.

In this example the bounds of our upset region will be 300 and 200 (just for some easy numbers). 300 and 200 equate to complete different percentages when comparing red team to blue team. 300 may turn P1 into 0.75 on the red graph while 300 may turn P1 into 0.6 on the blue graph. So frame of reference matters when trying to find the win percent. To find the red team win percent in the upset region we can simply find the P1 value of the upper bound of the upset region on the blue graph And subtract from P2 on the blue graph placed at the upset region minimum. In other words, P1 at 300 on the blue graph minus P2 at 200 on the blue graph. The reason we use the blue graph to find a red value, is we need to find the percentage of time blue will actually score less than red at a given score. We use the bounds because they come out to a cleaner and closer percent chance but like everything this is not perfect. It is an approximation. Do the same method for the blue team and you will have all the percentages you need. We will call them UWB and UWR.

To find our overall win chance percent for blue we add UWB, DWB, and DLR. Take that sum and subtract it by UWR, DLB. Now you have your overall win chance for blue. The reason we do all of that is because a direct win for blue will add to the UWB (since UWB does not account for them) and a DLB will take away from the percent. The reason behind DLR being added to blue is due to any time red loses blue wins. The same reasoning goes for why we subtract by UWR. Any time red wins blue loses. Repeat this same calculation for the red team by switching blue with red and you have both win chances. Keep in mind, that these do not account for ties, unforeseen events, etc. We know this because we are only looking at a small range of numbers. To find how confident we are in our results we can jank it with the method described below.

In order to find confidence you see what percent chance you have of ever having a value land in your accounted for range. This is found by setting P1 to your uppermost bound of the two graphs, and P2 to your lower most bound of the two graphs. In other words subtract the P value at the two extremes of the two curves and the resulting percentage is how confident you are. If you want to increase this confidence, account for more data. But with the method listed above you should get around 52-78% confidence every time depending on how close the match is. The closer it is, the smaller your confidence.

The following method works for matches into the future. With just 3 data points per robot, some events get to around 30 matches into the future before the predictions start to lose the high level of accuracy and the lack of data begins to show. It all depends on how wildly robots improve or worsen as the competition goes on.

Why this works:

There is an interesting phrase that helps explain some of this: “A group will act logically and an individual will act randomly”. This algorithm is different from others because it normalizes everything to the rest of the competition or “group”. If we keep it specific to the robot then you run into the issue of the randomness involved in how a driver approaches a match since they will never play the exact same every single time. They will have a degree of randomness in what they do making certain things inconsistent. OPR, DPR, CCWM try to fix this by observing it through all matches played but at its foundation, is still focusing on the individual. This is what I believe is part of the inconsistent nature behind OPR, DPR, and CCWM that many complain about. That being said, RPP is not something that can be done with a script and some api calls like OPR, DPR, and CCWM can be. It needs the actual scores of the robots which makes it only better than traditional OPR scouting at the individual competition level.