

Some Worked Examples

In the examples below, it is shown how the problem scoreboard is constructed and what the answer comes out to be after numerical solution of the equations which use the scoreboard as an input data set. When the scoreboard is of size n (i.e., with n rows and columns) there will be n coupled nonlinear equations. A separate document on this site [[Derivation](#)] gives the derivation of the set of equations.

1. A record of wins and losses in a group of athletic teams.

The contest consists of seven teams playing each other to determine the season champion. After only part of the season of play, it is desired to find the team standings at that point. The table below shows the scoreboard for this problem (the team vs. team wins and losses) plus extra descriptive data about the contest (total wins and losses of each team and fraction of games won). Only the scoreboard data are used in the analysis.

| | <u>1</u> | <u>2</u> | <u>3</u> | <u>4</u> | <u>5</u> | <u>6</u> | <u>7</u> | <u>Wins</u> | <u>Games</u> | <u>Wins/games</u> |
|--------|----------|----------|----------|----------|----------|----------|----------|-------------|--------------|-------------------|
| 1 | - | 0 | 4 | 0 | 3 | 5 | 2 | 14 | 24 | .583 |
| 2 | 0 | - | 4 | 0 | 2 | 5 | 3 | 14 | 24 | .583 |
| 3 | 4 | 3 | - | 2 | 1 | 2 | 1 | 13 | 30 | .433 |
| 4 | 2 | 2 | 1 | - | 3 | 4 | 2 | 14 | 20 | .700 |
| 5 | 1 | 4 | 5 | 3 | - | 4 | 3 | 20 | 34 | .588 |
| 6 | 2 | 0 | 2 | 1 | 4 | - | 1 | 10 | 32 | .312 |
| 7 | <u>1</u> | <u>1</u> | <u>1</u> | <u>0</u> | <u>1</u> | <u>2</u> | <u>-</u> | 6 | 18 | .333 |
| Losses | 10 | 10 | 17 | 6 | 14 | 22 | 12 | | | |

Teams 1 and 2 would normally be considered tied in the standings because they are equal in fraction of games won. But team 1 has won more against team 5 and less against team 7, and team 5 is far stronger than team 7. This should give an edge to team 1 over team 2, and the analysis by the vector method will bear this out.

If all teams had played the same number of games, the scoreboard could be used in the algorithm just as it stands. But the example has been picked to illustrate the scoreboard changes that are needed when unequal numbers of games are played. This problem is just like the battlefield problem, in which one compares weapon systems on the basis of their records of eliminating opposing weapons. Here we use the numbers of games as the group size instead of the numbers of weapons in an engagement and we use the groupability property of the algorithm. The team is assumed to be made up of a lot of sub-teams, each of which plays one game and either wins or loses. The sub-teams are then averaged, so that all of them are identical, meaning that they can be formed into a single group of sub-teams for processing by the algorithm. The first step in the processing is to divide each of the above scoreboard entries by the size of the defending group--column entries divided by the column's group size. Running the algorithm on that revised scoreboard will give the strengths of the teams as groups of attackers (winners)--the groups in the rows. We will use the applied strengths--the projections of the attack vectors on the contest vector--for scalar designators of strength. Then, the last step is to divide the team groups' applied strengths by the group sizes to obtain the single-game applied strengths of the teams. What we did was construct a single average game, with all teams participating in it as if it were a military engagement, and then we handle it as a military-analysis problem.

Making the above changes to the scoreboard and running the algorithm gives the results shown on the table below in the 'applied strength' column. The 'wins/games' column was copied from the previous table for easy side-by-side comparison. The applied strength is for a single average game, as described in the last paragraph. To facilitate comparison between the two columns, the applied-strength numbers were scaled to give both columns the same sum. The two columns of numbers correlate very well, and it was mentioned above why team 1 should come out ahead of team 2. Teams 6 and 7 switch their positions at the bottom of the rankings.

| | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 11 | -- | .4 | -- | .2 | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| 12 | -- | .1 | -- | -- | .1 | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| 13 | -- | -- | .4 | .3 | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| 14 | -- | -- | .3 | -- | .4 | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| 15 | -- | -- | -- | .4 | .3 | -- | -- | -- | -- | -- | -- | -- | -- | -- |

Notice the scores 0.4 and 0.3 repeated four times in the entries for the pairings (lower left part of the array). The entries in the upper-right section are for equations giving the strengths of the candidates; these strengths will enter the equations of the lower-left (for the strengths of the pairings) and prevent ties from forming. The next step is to run the contest algorithm program on this scoreboard to obtain the set of vectors for the problem--candidate vectors pointing at pairings and pair vectors pointing at candidates. When this is done, every occupied cell in the last array will contain a vector component--a component of the row-participant's vector pointing in the direction of the column participant. Each row of the solution represents the strength vector of a participant in the "contest".

Every potential pair will have a *quality* which I define as the product of the magnitudes of the two orthogonal components of the *pair's* vector (which point in the directions of the *pair's* candidates). Thus the quality number is the area of the rectangle formed by the two components of the pair's strength vector. The raw data for rows 6-15 of the above array show products of the row entries that are the same for four pairings. A run of the algorithm eliminates ties, producing the quality numbers shown on the following table:

| <u>Pair</u> | <u>Quality of the pair</u> |
|-------------|----------------------------|
| 1,2 | 0.0329 (2nd pair) |
| 1,3 | 0.0094 |
| 1,4 | 0.0066 |
| 1,5 | 0.0058 |
| 2,3 | 0.0142 |
| 2,4 | 0.0299 |
| 2,5 | 0.0033 |
| 3,4 | 0.0511 (1st pair) |
| 3,5 | 0.0450 |
| 4,5 | 0.0473 |

The pairs selected are 3,4 and 1,2. Although candidate 5 was able to make strong pairings, she lost out because her best potential partners became the first pair selected.

Extending this to triple rooms--three roommates to a room--is straightforward. First we have our candidates express preferences for *combinations* (the room with its occupants) instead of for individuals. (The single-roommate problem is a room choice with a two-person combination in it.) Then we recognize that the preference distribution functions can be treated like probability distribution functions in that the three preferences for a combination can be multiplied together to form the joint preference for a room containing that combination. Then the problem is set up just as it was for pairs, except that the 'other party' that is expressing a preference for a particular candidate is a two-person combination whose preference for the candidate is a combination of two preferences for the room. That is, if candidates A, B, and C are interested in becoming roommates, the preference for A will be the square root of the product of the preferences of B and C for the room. To illustrate this mathematically, say that candidates A, B, and C have preferences p_a , p_b , and p_c for a room with A, B, and C as roommates. Then $(p_a)(p_b)(p_c)$ is the joint preference for the room and that can be split into factors $[(p_b)(p_c)]^{1/2}$, $[(p_a)(p_c)]^{1/2}$, and $[(p_a)(p_b)]^{1/2}$. These factors will be the deduced preferences of the 'room' for A, B, and C, respectively. Entering these deduced preferences into the upper-right section of the scoreboard will permit the solution of the three-roommate problem. The assumption behind this is a very reasonable one: that a candidate's preference for a triple room is the product of equal preferences for the two other roommates who will be occupying the room. (Extending to higher-order combinations will proceed in the same way. Problems with mixed-order combinations are soluble, too.)

3. An auction with sealed bids.

One place where preference numbers on choices are clearly evident is in an auction, where the monetary bid is a direct statement of how much the bidder desires the object. This example is a sealed-bid type of auction, one in which all of the bids are submitted in advance and the winners are announced later. A strong bidder would be one who bids large amounts on popular objects; popular objects are those receiving large bids from strong bidders. If two bids are tied for the maximum on a particular object, the strength of the bidder will enter into the decision on who wins. From this it follows that strength differences between bidders may let a bidder win with less than the largest bid. (This is something like a "volume discount" offered to a major player in the auction.) Let our example consist of 3 bidders and 4 objects to be sold in the auction. The bidders are labeled 1,2,3 and the objects are labeled A,B,C,D. The bids are as follows:

| Bidder | A | B | C | D | Total bids |
|--------|----|---|---|---|------------|
| 1 | 10 | 6 | 8 | 7 | 31 |
| 2 | 10 | 4 | 3 | 6 | 23 |
| 3 | 0 | 0 | 9 | 2 | 11 |

The scores look like this:

| | 1 | 2 | 3 | A | B | C | D |
|---|----|----|---|----|---|---|---|
| 1 | - | 0 | 0 | 10 | 6 | 8 | 7 |
| 2 | 0 | - | 0 | 10 | 4 | 3 | 6 |
| 3 | 0 | 0 | - | 0 | 0 | 9 | 2 |
| A | 10 | 10 | 0 | - | 0 | 0 | 0 |
| B | 6 | 4 | 0 | 0 | - | 0 | 0 |
| C | 8 | 3 | 9 | 0 | 0 | - | 0 |
| D | 7 | 6 | 2 | 0 | 0 | 0 | - |

The 1,2,3 rows show the relative desires of the bidders for the objects; the A,B,C,D rows show that the objects favor the bidders by the same amount as the bidders favor the objects. These rows are not normalized in an auction because neither the bidders nor the objects are *a priori* equal.

For this type of problem, the objective is different than for the roommates problem, where we wanted to group together participants who have similar preferences for their groups. Here we just want to connect two participants which have identical preferences for each other and together make the biggest contribution to the overall contest, then go on to connect the next pair, and so on. The *contest vector* represents the contest as a whole, the *applied strength* represents the contribution of a participant to the contest, and the *partial applied strength* represents the part of the participant's contribution which is attributable to direct interaction with another participant. The applied strength is the projection of the participant's strength vector upon the contest vector, and the partial applied strength is the projection (upon the contest vector) of the component of that strength vector which lies in the direction of another participant. A pair of participants will have a pair of associated partial applied strengths (from two vector components) and the sum of these two will represent the contribution of the pair to the contest.

The computer program solves the problem and produces a list of sums of pairs of partial applied strengths. On the list below, the pairs designated as 'winner' show which bidder gets the object. The sum of the column equals the magnitude of the contest vector.

| <u>Pair</u> | <u>Sum of partial applied strengths of the pair</u> |
|-------------|---|
| 1,A | 0.180 (winner) |
| 1,B | 0.084 (winner) |
| 1,C | 0.144 (winner) |
| 1,D | 0.107 (winner) |
| 2,A | 0.160 |
| 2,B | 0.048 |
| 2,C | 0.048 |
| 2,D | 0.080 |
| 3,A | 0 |
| 3,B | 0 |
| 3,C | 0.125 |
| 3,D | 0.023 |

Bidder 1 gets all of the objects, even though object C received a higher bid from bidder 3 and there was a tie bid for object A. This softening of the criticality of a bid shows the tendency of a winner to have more in his favor than the precise amount of a single bid: He should be rewarded if his other bids push up the general level of the auction, and his reward is in the form of a bit of leeway.

In auctions there is often a 'reserve' or minimum bid--an amount which the seller must receive before he will let the object go. It is a simple matter to have the inputs be the above-the-reserve amounts instead of the absolute amounts. In that situation, all bids below the minimum will be given the value zero and those above the minimum will have the reserve amount subtracted from them. Then the solution follows as shown above.

4. Bidding on contracts to do a set of jobs.

Suppose a major body of work can be broken up into pieces with the pieces contracted out separately. Awarding contracts for the jobs on the basis of sealed bids is like an auction except that the winners are supposed to be the participants with the lowest bids. So how can a scoreboard reflect the situation in which preference goes to smaller numbers instead of larger? The answer here is that the scoreboard cannot distinguish between the two, but the strength computation can make the distinction. All one need do is enter the data as if this were an auction and then compute normally. The lower bidders in the auction picture will have to be considered the stronger ones when low bidding is a criterion for winning. Selection of the winners would proceed as in the auction, except that we select pairs from the bottom up, instead of the top down, on the list of paired partial applied strengths. A bidder may win with a bid slightly higher than the minimum on a particular job if he has a record of sufficiently low bids throughout the rest of the competition.

5. A diving or figure skating contest judged by a panel of judges.

In the judged contest of this section, we have 4 contestants whose performances are scored by a panel of 3 judges, with the results shown in the 'raw' columns on the table below. As is typical in these contests, the judges give very high scores on a 0-10 scale. The first step will have to be normalization of the raw scores, in this case to a sum of 4. (Any normalization number may be used.) The raw and normalized judges' results are the following for judges A, B, and C with respect to candidates 1, 2, 3, and 4:

| <u>Contestant</u> | <u>Raw</u> | | | | <u>Normalized</u> | | | |
|-------------------|------------|-----|-----|------|-------------------|-------|-------|-------|
| | A | B | C | Avg. | A | B | C | Avg. |
| 1 | 9.5 | 9.3 | 8.1 | 8.97 | 1.003 | 1.005 | 0.994 | 1.001 |
| 2 | 9.1 | 9.0 | 7.9 | 8.67 | 0.960 | 0.973 | 0.969 | 0.967 |
| 3 | 9.7 | 9.2 | 8.1 | 9.00 | 1.024 | 0.995 | 0.994 | 1.004 |
| 4 | 9.6 | 9.5 | 8.5 | 9.20 | 1.013 | 1.027 | 1.043 | 1.028 |

Normalization gets all three judges operating at the same level of participation. It is possible for the raw and

normalized averages to give a different apparent winner, but that did not happen in this example. (I say 'apparent' winner because the problem is far from solved at this point.)

The problem is now treated like a simple contest between 'contestants' and judges, but one with a scoreboard showing all participants as having tie scores with each other. (That is to say, a contestant's preference for a judge is equal to that judge's preference for that contestant.) This leads to the array of input data for the contest algorithm which is shown below. The upper-right section is just the transpose of the lower-left section.

| | 1 | 2 | 3 | 4 | A | B | C |
|---|-------|-------|-------|-------|-------|-------|-------|
| 1 | 0 | 0 | 0 | 0 | 1.003 | 1.005 | 0.994 |
| 2 | 0 | 0 | 0 | 0 | 0.960 | 0.973 | 0.969 |
| 3 | 0 | 0 | 0 | 0 | 1.024 | 0.995 | 0.994 |
| 4 | 0 | 0 | 0 | 0 | 1.013 | 1.027 | 1.043 |
| A | 1.003 | 0.960 | 1.024 | 1.013 | 0 | 0 | 0 |
| B | 1.005 | 0.973 | 0.995 | 1.027 | 0 | 0 | 0 |
| C | 0.994 | 0.969 | 0.994 | 1.043 | 0 | 0 | 0 |

The judges obtain strength when favored by strong contestants and vice versa.

This judging problem is set up for processing by the contest algorithm. Solving the set of 7 simultaneous nonlinear contest equations numerically gives all the strength vectors of the participants and their vector sum (the contest vector). The projection of a participant's strength vector upon the contest vector is the applied strength of the participant, and it is the applied strength of the contestant which determines his rank in the contest. The applied strengths of all participants are given on the next table, normalized to a sum of unity.

| <u>Participant</u> | <u>Applied strength</u> |
|--------------------|-------------------------|
| 1 | 0.137 |
| 2 | 0.132 |
| 3 | 0.138 |
| 4 | 0.141 |
| A | 0.151 |
| B | 0.151 |
| C | 0.151 |

We do not care about the judges (A, B, and C), but they all turn out to be very even here. The winning contestant is 4, and all contestants rank the same as they were in the input data. In most cases this will probably be true, and it was not the intent here to try to find a case where the outcomes would be different.

6. Analyzing the Supreme Court.

In this example we will use a hypothetical Supreme Court with 5 justices to save space in the displays. Say that they have produced opinions on 10 cases in a year's session. Let A,B,C,D,E represent the justices and numbers 1-10 represent the cases. In the data array below, each case will show a marker "1" for a justice who is on the majority side of the opinion. (Thus, each case will have at least three 1's in its column.)

| <u>Justice</u> | <u>1</u> | <u>2</u> | <u>3</u> | <u>4</u> | <u>5</u> | <u>6</u> | <u>7</u> | <u>8</u> | <u>9</u> | <u>10</u> | <u>Total</u> |
|----------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|--------------|
| A | 1 | 1 | | 1 | | 1 | | 1 | | 1 | 6 |
| B | | 1 | 1 | | 1 | | 1 | | 1 | | 5 |
| C | 1 | | 1 | 1 | 1 | 1 | | 1 | | 1 | 8 |
| D | | 1 | 1 | 1 | 1 | | 1 | 1 | 1 | | 7 |
| E | 1 | 1 | 1 | 1 | | 1 | 1 | | 1 | 1 | 8 |

This is a problem in which the justices demonstrate strength by being on the majority side of cases. The more on the majority side and the stronger the cases, the greater is the strength of the justice. The cases are stronger

when they get more votes on the *minority* side, with the stronger justices on the minority side giving more strength to the case. (In other words, the importance of a case is determined by the strength of the opposition to it.) A unanimous decision represents a very weak case and gets a computed strength of zero. The scoreboard for the problem is shown below, where it is seen that the upper-right section is the same as the data set shown above and the lower-left section is just the transpose of the upper-right section, but with the 1s and 0s interchanged. The participants are numbered 1 - 15, with 1 - 5 being the justices and the rest being the cases.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
| 1 | | | | | | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 2 | | | | | | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 3 | | | | | | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 4 | | | | | | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |
| 5 | | | | | | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 0 | 1 | 0 | | | | | | | | | | |
| 7 | 0 | 0 | 0 | 0 | 0 | | | | | | | | | | |
| 8 | 1 | 0 | 1 | 0 | 0 | | | | | | | | | | |
| 9 | 0 | 1 | 0 | 0 | 0 | | | | | | | | | | |
| 10 | 1 | 0 | 0 | 0 | 1 | | | | | | | | | | |
| 11 | 0 | 1 | 0 | 1 | 0 | | | | | | | | | | |
| 12 | 1 | 0 | 1 | 0 | 0 | | | | | | | | | | |
| 13 | 0 | 1 | 0 | 0 | 1 | | | | | | | | | | |
| 14 | 1 | 0 | 0 | 0 | 0 | | | | | | | | | | |
| 15 | 0 | 1 | 0 | 1 | 0 | | | | | | | | | | |

From the raw data, it would appear that justices C and D (3 and 5) are tied for first place, each being in the majority eight times. The problem is solved by finding the applied strengths of the justices, with the contest algorithm operating on the scoreboard.

The table below gives the relative applied strengths of all participants, the first five being the justices. The column marked 'gain' is explained below.

| <u>Participant</u> | <u>App. str.</u> | <u>Gain</u> |
|--------------------|------------------|-------------|
| 1 | 0.092 | 0.835 |
| 2 | 0.083 | 0.759 |
| 3 | 0.128 | 1.166 |
| 4 | 0.117 | 1.067 |
| 5 | 0.129 | 1.173 |
| 6 | 0.056 | |
| 7 | 0 | |
| 8 | 0.058 | |
| 9 | 0.026 | |
| 10 | 0.059 | |
| 11 | 0.056 | |
| 12 | 0.058 | |
| 13 | 0.057 | |
| 14 | 0.027 | |
| 15 | 0.056 | |

Justices C and E (3 and 5) are close, but no longer tied. When a justice is especially likely to be in the majority, like justices C and E in this example, he or she is said to be a *swing* justice. The applied strength is an indicator of the swing tendency of a justice. To isolate the swing effect, I use a quantity--*gain*--shown for the justices in the second column. Gain may be greater or less than 1 and it is the factor of increase over the average applied strength of the justices. The gain quantity is relatively independent of the number of cases. The gain column on the table above shows how justices C and E have gained at the expense of justices A and B, and are therefore more likely to be swing justices. The second case (numbered as participant 7) has a unanimous decision and gets zero strength as a result. (It could have been dropped from the data set at the beginning.)

The program computes the cosines of the angles between attack vectors, and the cosine is zero when the vectors are orthogonal (indicating opposition) and unity when they are parallel (indicating collaboration). For this example, justices A and B have zero for that cosine, and it can be seen from the input data that they are in full opposition to each other when the unanimous case is removed. Justices A and C show the most collaboration, with a cosine of 0.853, but justices B and D are close behind, with 0.849. These numbers, which take account of the strengths of cases, are a better measure of collaboration than just the fraction of majorities in which the justices are on the same side. (The fractions are 5/7 and 4/6, respectively, when the unanimous case is not counted.)

7. The famous circular-preference situation.

It is often stated that we should not be surprised if some social-choice problems turn out to be insoluble, because one very simple situation has no solution. This is the case of three choices with circular preferences, for example choices A, B, and C with votes for them as shown in the following table (3 voters, each distributing 6 votes among the choices):

| <u>Voter</u> | <u>A</u> | <u>B</u> | <u>C</u> |
|--------------|----------|----------|----------|
| 1 | 1 | 2 | 3 |
| 2 | 3 | 1 | 2 |
| 3 | 2 | 3 | 1 |

Each voter prefers a different choice. Two-thirds of the voters reject every choice (as a winner), so how could there be a winner when any one selected would be rejected by 2/3 of the voters?

The answer is that, if this three-way symmetry produces no winner, it must be producing a three-way tie, and that is just the solution which the vector method gives. Ties are very ordinary events and not a cause for declaring a problem insoluble. If another voter is brought into the above picture or if one of the existing three voters is allowed to increase one vote by the slightest amount (while decreasing elsewhere to keep the normalization intact) the solution will go in the expected direction and produce the expected winner.

8. Picking a best pair of players for doubles tennis

The coach of a school tennis team wants to select a pair of players from his team for entry into a state tournament, where they will represent the school in doubles tennis. He will make the selection by forming all possible pairs from his team and having every pair play a match against every other pair. The results will tell him which pair is the best at doubles play, which pair is second best, etc. The team has 6 players, permitting 15 pairings and 90 matches among the pairs. (It would be infeasible to have all possible pairs play against each other for a player population of any appreciable size, because of the explosion in the number of matches needed. The purpose of this example is not to assist tennis coaches in any way but to explore the general philosophy of pair matching, using preference numbers that are observable rather than subjective.)

The following table gives the results of all of the matches. The players are numbered 1 to 6 and the pairs are designated (1,2), (1,3), etc. Next to each pair is a column showing the opposing pairs which have been played against, with the losers among the opposing pairs underlined. The last column gives the score for the designated pair (matches won).

| <u>PAIR</u> | <u>OPPOSING PAIRS</u> | <u>SCORE</u> |
|-------------|---|--------------|
| (1, 2) | (<u>3, 4</u>) (<u>3, 5</u>) (<u>3, 6</u>) (<u>4, 5</u>) (4, 6) (5, 6) | 4 |
| (1, 3) | (<u>2, 4</u>) (2, 5) (<u>2, 6</u>) (4, 5) (<u>4, 6</u>) (<u>5, 6</u>) | 4 |
| (1, 4) | (2, 3) (<u>2, 5</u>) (2, 6) (<u>3, 5</u>) (3, 6) (5, 6) | 2 |
| (1, 5) | (<u>2, 3</u>) (<u>2, 4</u>) (2, 6) (3, 4) (<u>3, 6</u>) (<u>4, 6</u>) | 4 |
| (1, 6) | (2, 3) (2, 4) (<u>2, 5</u>) (3, 4) (<u>3, 5</u>) (4, 5) | 2 |
| (2, 3) | (<u>1, 4</u>) (1, 5) (<u>1, 6</u>) (4, 5) (<u>4, 6</u>) (5, 6) | 3 |

| | | |
|--------|---|----------|
| (2, 4) | (1, 3) (1, 5) (<u>1, 6</u>) (3, 5) (<u>3, 6</u>) (5, 6) | 2 |
| (2, 5) | (<u>1, 3</u>) (1, 4) (1, 6) (<u>3, 4</u>) (<u>3, 6</u>) (<u>4, 6</u>) | 4 |
| (2, 6) | (1, 3) (<u>1, 4</u>) (<u>1, 5</u>) (3, 4) (<u>3, 5</u>) (4, 5) | 3 |
| (3, 4) | (1, 2) (<u>1, 5</u>) (<u>1, 6</u>) (2, 5) (<u>2, 6</u>) (<u>5, 6</u>) | 4 |
| (3, 5) | (1, 2) (1, 4) (1, 6) (<u>2, 4</u>) (2, 6) (4, 6) | 1 |
| (3, 6) | (1, 2) (<u>1, 4</u>) (1, 5) (2, 4) (2, 5) (<u>4, 5</u>) | 2 |
| (4, 5) | (1, 2) (<u>1, 3</u>) (<u>1, 6</u>) (<u>2, 3</u>) (<u>2, 6</u>) (3, 6) | 4 |
| (4, 6) | (<u>1, 2</u>) (1, 3) (1, 5) (2, 3) (2, 5) (<u>3, 5</u>) | 2 |
| (5, 6) | (<u>1, 2</u>) (1, 3) (<u>1, 4</u>) (<u>2, 3</u>) (<u>2, 4</u>) (3, 4) | <u>4</u> |
| | | 45 |

It is clear that scores alone are not going to decide which pair is best, since seven pairs have the highest score of 4. Let us next form a matrix of player scores in which each element (row i , column j) is the score produced by player i when paired with player j . Call it the *scoreboard*.

| | | SCOREBOARD | | | | | | |
|---|--|------------|----------|----------|----------|----------|----------|--------------|
| | | <u>1</u> | <u>2</u> | <u>3</u> | <u>4</u> | <u>5</u> | <u>6</u> | <u>Total</u> |
| 1 | | - | 4 | 4 | 2 | 4 | 2 | 16 |
| 2 | | 4 | - | 3 | 2 | 4 | 3 | 16 |
| 3 | | 4 | 3 | - | 4 | 1 | 2 | 14 |
| 4 | | 2 | 2 | 4 | - | 4 | 2 | 14 |
| 5 | | 4 | 4 | 1 | 4 | - | 4 | 17 |
| 6 | | 2 | 3 | 2 | 2 | 4 | - | 13 |

The scores here are unnormalized preference indicators for partners, not normalized because the participants are not *a priori* equal. It would be reasonable to expect that the cell with the biggest number would indicate the best pair, but this example has seven pairs tied for maximum. If player i has the same pair score with either j or k , but k is a better performer than j in his other pairings, then k should be preferred over j as a partner for i . The contest algorithm brings out this effect for all participants and finds the best pair.

Running the computer program on the contest scoreboard gives the following pair sums. The pair sum is player i 's partial contribution to the contest vector (due to his vector component in direction j) plus player j 's partial contribution (due to his vector component in direction i).

| <u>PAIR</u> | <u>PAIR SUM</u> |
|-------------|----------------------------|
| (1, 2) | 0.093 |
| (1, 3) | 0.089 |
| (1, 4) | 0.044 |
| (1, 5) | 0.097 (winning pair) |
| (1, 6) | 0.042 |
| (2, 3) | 0.066 |
| (2, 4) | 0.044 |
| (2, 5) | 0.096 |
| (2, 6) | 0.063 (third-choice pair) |
| (3, 4) | 0.084 (second-choice pair) |
| (3, 5) | 0.023 |
| (3, 6) | 0.040 |
| (4, 5) | 0.092 |
| (4, 6) | 0.040 |
| (5, 6) | 0.088 |

It is, of course, no surprise that pair (1,5) wins the competition. As a pair they win as many matches as any other pair and as individuals they are on the winning side 16 and 17 times. Other pairs are close behind.

To find the second-ranked pair, we could delete the rows and columns of players 1 and 5 and run the program again, or we could just look at the numbers from the first run and pick the next-lower pair sum which does not involve players 1 and 5. Although the former approach will bring the next pair choice into sharper relief, it does so by discarding data. The better approach (more accurate and easier) is to pick the pair choices from the

original run of the algorithm, as was done above. The pair choice makes use of the strengths of players as determined from performance in all of their pairings. Even if higher-ranking pairs have already been picked, the performance of a lower-ranking player with and against those already-picked players still gives useful data for placing that lower-ranking player into his proper pair.

This is a good place to insert a comment about the traditional method of pair matching, i.e., finding a stable solution and then attempting to justify it. This tennis example has seven pairings in which the two players are paired with a favorite (or with a tie for favorite). If all players are paired simultaneously with a favorite, we must have a stable solution, and there is a solution like that: (1,2), (3,4), and (5,6). From the stability point of view, this is a better solution than what the contest algorithm gave us, because all pairs have a pair score of 4. The contest algorithm solution, while also stable, gave two 4's and a 3 -- lower on the stability scale, we might say. However, the purpose of the competition was not to produce pairs of players satisfied with their partners as well as can be arranged. Rather, it was to produce a best pair, a second-best pair formed from the remaining players, and so on.

9. An election

Say we have an election with 5 electors choosing from among 3 candidates (A, B, and C), and the electors each have 6 votes to spread among the candidates -- 3 for first choice, 2 for second choice, and 1 for third choice. Number the participants in the election from 1 to 8, with 1 - 5 being for the electors and 6 - 8 for the candidates. The candidates also vote for the electors; their votes for electors are the same as the votes they received from them. (Since both electors and candidates are casting votes, the term 'voters' is not used.) The array of input data is as follows:

| | | Election Data | | | | | | | |
|---|---|---------------|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | A | B | C |
| 1 | - | 0 | 0 | 0 | 0 | 0 | 3 | 2 | 1 |
| 2 | 0 | - | 0 | 0 | 0 | 0 | 3 | 2 | 1 |
| 3 | 0 | 0 | - | 0 | 0 | 0 | 3 | 2 | 1 |
| 4 | 0 | 0 | 0 | - | 0 | 0 | 2 | 3 | 1 |
| 5 | 0 | 0 | 0 | 0 | - | 0 | 1 | 3 | 2 |
| A | 3 | 3 | 3 | 2 | 1 | - | 0 | 0 | 0 |
| B | 2 | 2 | 2 | 3 | 3 | 0 | - | 0 | 0 |
| C | 1 | 1 | 1 | 1 | 2 | 0 | 0 | - | 0 |

The numbers in the row of a candidate will tend to predict his chance of winning, along with the strengths of the electors associated with those numbers. The contest algorithm was designed for this interaction between participants; applying it will show how the candidates compare and which one is the winner.

The example has been set up with the first three electors identical, as seen by comparing their rows and columns. Now contest analysis has the feature of *groupability*, by which identical participants can be grouped in any arbitrary way without changing the results. Here we can lump electors 1, 2, and 3 into a single group of three. This leaves us with three elector *groups*, which we can number 1, 3, and 4 (number 1 consisting of three individuals). The contest-analysis rule for combining into groups is to *add the rows together* and *delete the repeated columns*.

The election data array above can be turned into a scoreboard for the election by simply numbering the candidates as 6, 7, and 8. But when electors are grouped, the grouping rule is applied and the scoreboard becomes one with only six groups, numbered 1-3 for the electors and 4-6 for the candidates. The final scoreboard is shown next, with participants 4, 5, and 6 being the candidates. A column has been added to show the size of the group.

| | 1 | 2 | 3 | 4 | 5 | 6 | Size |
|---|---|---|---|---|---|---|------|
| 1 | - | 0 | 0 | 9 | 6 | 3 | 3 |
| 2 | 0 | - | 0 | 2 | 3 | 1 | 1 |
| 3 | 0 | 0 | - | 1 | 3 | 2 | 1 |
| 4 | 3 | 2 | 1 | - | 0 | 0 | 1 |
| 5 | 2 | 3 | 3 | 0 | - | 0 | 1 |
| 6 | 1 | 1 | 2 | 0 | 0 | - | 1 |

The quantity which determines the winner is the applied strength of the candidate, as found by the algorithm. The table below shows the applied strengths (normalized to unity) of all participants, electors as well as candidates, for the example in both its ungrouped and grouped form. (Now the participant numbering reverts to its original 8-participant form.) The grouping makes no difference, as is shown by the grouped applied strength being just the sum of that quantity for the three individuals in the group.

| Participant | Applied strength | |
|-------------|------------------|---------------|
| | Ungrouped | Grouped |
| 1 | .1219 | - |
| 2 | .1219 | .366 |
| 3 | .1219 | - |
| 4 | .1211 | .121 |
| 5 | .1099 | .110 |
| A | .1615 | .161 (winner) |
| B | .1612 | .161 |
| C | .0806 | .081 |

Candidates A and B each received 12 votes, but contest analysis breaks the tie and gives A a slight edge to win the election. It was necessary to go to four significant figures to disclose the winner. Among the electors, 5 is the weakest because he skewed his preferences toward the weakest candidate.

This vote-grouping procedure opens the door to a way for handling the problem of the American Electoral College, the controversial method specified in the Constitution for electing American presidents. The congressional districts can be taken as groups with sizes equal to the votes cast in them. Within each district, there can be a separate election with one vote per voter for candidate of choice, and a group of voters for each candidate. The separate election can then be analyzed with the algorithm to give a candidate weights for the district. These weights can then be used as the preference numbers for another run of the algorithm for the country as a whole, this time with the districts as groups having preference distributions over candidates. If desired, another stage in the aggregation can be inserted, so that the complete election proceeds in three steps--voters in districts, districts in states, and states within the country. This solves the long-standing problem with vote aggregation that has plagued democracies since the days of the French revolution.

10. Baseball: analysis of a partial season of American League games

In this example, the aim is to investigate a planned contest which has not accumulated its planned set of data. The play of a partial American League baseball season was chosen--games played through 1 June 2001. Many of the teams had not yet played each other at that point in the season.

Fourteen teams are in the league, and their indices (1-14) will be the following:

- | | |
|--------------------------|------------------------|
| 1 - Baltimore Orioles | 8 - Detroit Tigers |
| 2 - Boston Red Sox | 9 - Kansas City Royals |
| 3 - New York Yankees | 10 - Minnesota Twins |
| 4 - Tampa Bay Devil Rays | 11 - Anaheim Angels |
| 5 - Toronto Blue Jays | 12 - Oakland Athletics |
| 6 - Chicago White Sox | 13 - Seattle Mariners |
| 7 - Cleveland Indians | 14 - Texas Rangers |

Now a data array will show how many games team i (in the column) won against team j (in the row) during the part of the season completed prior to 2 June. For entry into the contest algorithm, each element of the array is converted into the fraction of games played by team j which are won by team i . Games played are taken as the group sizes, as they were in the first example of this paper, so we are dividing each matrix element by the column's group size to obtain the scoreboard for the contest algorithm.

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|
| 1 | - | 3 | 1 | 8 | 0 | 0 | 1 | 4 | 0 | 3 | 2 | 1 | 0 | 2 |
| 2 | 3 | - | 5 | 6 | 4 | 0 | 0 | 0 | 3 | 3 | 0 | 4 | 2 | 0 |
| 3 | 6 | 6 | - | 0 | 2 | 0 | 2 | 0 | 6 | 2 | 0 | 3 | 2 | 0 |
| 4 | 5 | 0 | 0 | - | 2 | 0 | 1 | 2 | 2 | 0 | 1 | 1 | 0 | 1 |
| 5 | 0 | 1 | 4 | 4 | - | 3 | 0 | 0 | 3 | 0 | 3 | 1 | 2 | 5 |
| 6 | 0 | 0 | 0 | 0 | 3 | - | 4 | 6 | 0 | 0 | 2 | 1 | 1 | 4 |
| 7 | 5 | 0 | 2 | 5 | 0 | 1 | - | 9 | 5 | 0 | 3 | 0 | 0 | 4 |
| 8 | 2 | 0 | 0 | 4 | 0 | 4 | 3 | - | 0 | 0 | 3 | 0 | 0 | 6 |
| 9 | 0 | 3 | 0 | 4 | 4 | 0 | 1 | 0 | - | 5 | 0 | 1 | 0 | 2 |
| 10 | 3 | 3 | 4 | 0 | 0 | 6 | 0 | 5 | 8 | - | 2 | 2 | 1 | 1 |
| 11 | 0 | 0 | 0 | 3 | 3 | 4 | 3 | 3 | 1 | 1 | - | 3 | 1 | 3 |
| 12 | 0 | 2 | 3 | 2 | 5 | 5 | 0 | 0 | 1 | 2 | 3 | - | 1 | 2 |
| 13 | 3 | 4 | 4 | 1 | 4 | 5 | 0 | 0 | 4 | 1 | 6 | 5 | - | 4 |
| 14 | 1 | 0 | 0 | 2 | 1 | 2 | 2 | 0 | 1 | 0 | 3 | 5 | 2 | - |

The next table shows, for each team, the fraction of games won (games won/games played) and the applied strength (as computed by the contest algorithm, but scaled to fit into the same approximate range as the first column). The fraction won is the number traditionally used to rank the team in the league.

| Team | Fraction won | Applied strength |
|------|--------------|------------------|
| 1 | 0.472 | 0.121 |
| 2 | 0.577 | 0.161 |
| 3 | 0.558 | 0.167 |
| 4 | 0.278 | 0.073 |
| 5 | 0.481 | 0.129 |
| 6 | 0.412 | 0.113 |
| 7 | 0.667 | 0.165 |
| 8 | 0.431 | 0.107 |
| 9 | 0.370 | 0.104 |
| 10 | 0.673 | 0.191 |
| 11 | 0.472 | 0.127 |
| 12 | 0.491 | 0.138 |
| 13 | 0.774 | 0.213 |
| 14 | 0.358 | 0.099 |

What is very interesting about these numbers is the high correlation between the two columns. When a regression analysis is run on them (using their full precision, not rounded to the above three decimal places), the coefficient of determination (R-squared) comes out to be 0.96. This seems to say that athletic league play can be adequately handled the traditional way (using win percentages to determine team standings) except when it is necessary to separate teams which are very close in their standings (like teams 1 and 11 above) or when the numbers of games played are quite different from team to team.

Alan E. Johnsrud
(Last revision 18 Oct. 09)

Go to Home page: [Home](#)

Go to Topics page: [Topics](#)