Rank like a Doc: Distance metrics for concordance and consensus among partially-overlapping judgment sets that include ranked pro- and potentially unranked con- lists

Jeff Shrager
Commerce Net
and
Stanford University
Symbolic Systems Program (Adjunct)
Palo Alto, CA
jshrager@stanford.edu

Amy N. Langville
Department of Mathematics
College of Charleston
Charleston, SC
langvillea@cofc.edu.

May 25, 2022

Abstract

Your abstract.

1 Introduction

Because of the enormous uncertainties inherent in most of medicine, physicians with expertise in a particular area (hereafter "specialists") will often produce a short ranked list of recommended or "indicated" treatments ("pros"), and sometimes also a short list of "contra-indications" ("cons"), that is, treatments that the specialist recommends *against* choosing. We call either such a list an "n-rank", where n is the number of indicated or contra-indicated treatments. In the present work n will usually be 3, thus, a "3-rank". When the n-rank contains only pros it is also referred to as a top-n ranking in the literature. Patients also commonly obtain multiple n-ranks from different specialists and must decide whether to believe one over the other, form a consensus, or get yet more opinions.[1]

In addition to the rank order differing, multiple n-ranks may have different elements, that is, specialists may disagree both in the order of relevant treatments, and in what treatments are relevant.[2] We consider a target patient or cohort (hereafter just a "case", c), and a number of n-ranks, R_c, for c from different specialists, (s_i): $R_{c,s_i} = [T_1, T_2, ...]_{c,s_i}$ where T_1 is the highest ranked treatment and so on. The goal of the present work is to create a distance metric, $d(R_{c,s_a}, R_{c,s_b})$ where s_a and s_b are different specialists.[3]

One use of d is to estimate the "*difficulty*" of a particular case, represented, for example, by the average all-way distance between however-many specialists are consulted for opinions on the same case. Another use of d is to estimate the performance of a treatment ranking algorithm. Such an algorithm might be given an input describing c (say as a feature vector), a set of plausible treatments $\{T_i, ...\}$ (the "pharmacopeia"), and "training data", which might be, for example a list of prior cases including what treatments were utilized in those cases and the outcomes, say in terms of quality of life, longevity, blood pressure, etc. The algorithm uses the training data to score each treatment for the new case, and the n-rank ($R_{c,algorithm}$) is simply the best-first sort of the scored treatments.

The typical way that one would go about estimating the performance of such an algorithm is cross validation: We provide the algorithm a subset of the training data set (say 80%), holding the rest (20%) as a "test set", comparing the algorithmically-produced top indicated treatment with the ones actually chosen in the test set, and employing a statistic, such as "Area Under the Curve", to estimate how well the algorithm is doing.

[1] Sometimes multiple specialists will confer among themselves in what is sometime called a "tumor board", and report a consensus. In the present work we do not distinguish this from the n-rank coming from a single specialist.

[2] There is a great deal of complexity related to what counts as a "treatment". Treatments may be single drugs, or combinations of drugs (called "cocktails") that may overlap one another, or even be identical combinations of drugs under different "regimens", that is, given at different dosages or time periods. Even more generally, a "treatment" decision may involve getting a test, or a combination of complex treatment regimens and test, or even no action at all, that is, either "watchful waiting" or, in the worst case recommendation to a hospice. We do not dealing with any of this complexity; in the present work the T_i are considered to be exclusive non-complex treatments, involving single drugs, or at least cocktails and regimens that can be considered as single drugs.

[3] We assume that the opinions are from independent specialists who have not consulted with one another.

There are several problems that make this approach untenable in many areas of medicine. First, obtaining the training data may take years of collecting very noisy outcome measures. Second, because medicine is an ever-changing field, the data collected at one point in time is very likely to be outdated and irrelevant, or worse, confusingly wrong by the time enough data is collected to obtain useful statistics. For example, the "pharmacopeia" (the set of plausible treatments) may change often for important diagnoses that are being actively addressed by pharmaceutical companies. Therefore, it it will often be the case that entirely new treatments have been approved, or are coming into common usage, and older ones going out of favor (sometimes being completely withdrawn from the market, or no longer commonly manufactured). Furthermore, the way existing drugs are used in clinical practice, i.e., the "regimens", can change quite rapidly in active areas. For example, between 2020 and 2022 hundreds, perhaps thousands, of changes were made in best practices for treating patients with Covid-19, and dozens of new drugs and vaccines produced. Much of the data gathered in 2020 became nearly irrelevant in 2021 when vaccines became widely available, and again in 2022 when many new non-vaccine treatments were widely applied. In sum, training data collected over a long period (even sometimes a period of just a few months) will not only fail reflect these changes rapidly enough to be useful, but may actually lead to dangerously wrong predictions.

Although specialist recommendations do not have the ground-truth validity of outcomes, they have significant advantage: Specialists can give opinions at any time while the patient is alive, so that one need not wait until a patient's death to realize an outcome. Specialist opinions can even be obtained for a non-existant pseudo-patient, providing a wide range of experimental opportunities. The specialist can, and often do, report multiple potential indicated treatments, and sometimes also one or more *contra-indicated* treatments. On the other hand, outcomes data tells you only about the single treatment that was used. For all these reasons it is useful to augment cross-validation with continuous specialist validation by comparing the n-ranks produced with those produced for the same case by specialists. Thus motivating the present goal of comparing n-ranks with one another.

2 Background

Fagin, et al. [FKM+04], noting that "there is no well-studied and well-understood method for comparing two top k lists for similarity/dissimilarity", set out to define "reasonable and meaningful distance measures between top k lists?" As Fagin et al. have conducted a detailed analysis of the problem studying a wide range of measures, with citation, we will not extract and re-discuss all of those in the present work, but rather refer the reader to [FKM+04] for this background. Fagin, et al. begin by studying the special case of permutations of all of the objects in the domain, for this standard methods, such as Kendall's tau and Spearman's footrule apply.[4] Fagin et al. consider these methods as one of several classes of distance measures, but modify these to apply to partial lists. These researchers reject the straightforward approach of "symmetric difference (union minus the intersection), appropriately scaled" (e.g., [Lee98]) as "not adequate" because "two top 10 lists that are reverses of each other would be declared to be 'very close'.".

In the present context, the problem declared as "not adequate" by Fagin et al. is actually precisely what we want: Two n-ranks containing the same elements, but in a different order are, indeed, considered to be closer to one another than ranks that contains some non-common elements. For example, we want [a,b,c] to be closer to [c,b,a] that it is to [d,e,f]. At the same time, we would like [a,b,c] to be sensibly closer to [a,d,e] than is it to [d,e,f].

3 Data and Notation

We suppose that for a given case c, a specialist, s_i (whether an individual or team), produces an n-rank, as above: $[T_1, T_2, ...]_{c,s_i}$, where the tuple $[T_1, ...]$ can be further understood as comprised of indicated or "proposed" treatments($P = [T_1, T_2, ...]$ (aka. "pros") and contra-indicated treatments: $C = TC_1, T_2, ...]$, that is, treatments the same specialist indicates *against* using ("cons"). The specific number of pros and cons may vary, and sometimes there will be no cons given at all (i.e., $C = []$). In the first part of this paper we will assume that each specialist, s_i, produces exactly 3 pros for a given case, c, that is, the 3-rank: $P = [T_1, T_2, T_3]_{c,s_i}$, where T_1 is the top indicated treatment according to s_i, T_2 is the second best, and so on and no contra-indications, that is: $C = []$. Later we will discuss how contra-indications should be treated. However, as will be seen, contra-indications introduce additional complexities that we postpone until pros have been handled. To be clear, each proposed/indicated/pro treatment (P), or contra-indicated/con treatment (C_i), is selected from a larger set of all possible (or plausible) treatment: $[T_i, ...]$, called the "pharmacopeia", which is independent of the case or the specialist.

[4] Fagin et al. also look at Goodman and Kruskal's gamma. However, if there are no ties, as is the case in the present work, gamma is equivalent to tau.

4 Simulated 10-specialist 3-Rank Judgment Sets

To simulate the specialist indications described above, we generated a static pharmacopeia consisting of 60 treatments: 20 in each of 3 categories: immunotherapies, targeted therapies, and chemotherapies, or "Ix", "Tx", and "Cx" respectively. Each treatment has a set of "genome targets", created by uniformly choosing a random positive integer below 2^{30}, considered as a 29 binary number. We created 100 simulated cases, we assign each a "genome" by the same method. For example, a treatment whose genome target is randomly assigned the decimal value $715,827,882$, i.e., $0b101010101010101010101010101010$, which match (i.e., have the same "1"s or "0"s between the two binary representations in a given position) in 15 places with a case with genome 429496729, i.e., $0b11001100110011001100110011001$. Thus, for this case, this treatment would receive a score of 15.

We want our algorithmic quality metric to compare how well the algorithm's 3-rank compares with 3-ranks produced by multiple specialist specialists. We compare the algorithm with multiple specialists because, as will be seen shortly below, specialists have several different ways that they can produce the result, whereas the algorithm, in principle, has only one.[5] In the present work, and without loss of generality, we hereafter assume that the algorithm always produces the 3-rank: $[1, 2, 3]$. For each of the 100 simulated cases (i.e., their "genomes") we produce ten 3-ranks, as though ten separate specialists were asked to individually consider each of the 100 cases.

Each specialists can choose among any of the 60 treatments across the three categories: Cx, Tx, Ix. A given specialist's 3-rank for a given case is created as follows. First, all treatments (regardless of category) are scored against the case by the bit-matching algorithm described above, producing a 60-rank of all treatments. Next we introduce "individuality" into the judgments through two perturbation processes that we call "tweaking" and "twiddling". Tweaking slightly changes the score by -1, 0, or $+1$ in two stages. Stage 1 determines if any tweaking will be done (with probability p=0.5. If tweaking was chosen, then in Stage 2, we add or subtract 1. This is repeated for each treatment in the 60-rank. Next these are grouped into equi-score groups – that is, groups of items whose scores are numerically indistinguishable, so that within such a group there is no rationale for ranking items above one another. Twiddling randomly permutes these groups, which are then aggregated into a new 60-rank where there has been tweaking, and then the order within equi-score sets is random. (The final sorting algorithm retains order within equi-score sets, thereby retaining the result of the twiddling.) To create indicateations, one of two algorithms is employed[6]: two-thirds of the time the top 3 treatments from the 60-rank are selected to create the 3-rank, and in the remaining third of

[5] Of course, like a tumor board, there may be variability or other factors that lead the algorithm to internally produce multiple 3-ranks. For example, it might run thousands of internal simulations before reporting a result. However, again like a tumor board, at the end of the day it has to produce a single 3-rank.

[6] This is a slight simplification of what actually happens, but this detail is only relevant later when we consider contra-indications.

cases, the top treatment is selected from each of the three categories of immunotherapies, targeted therapies, and chemotherapies. The 100 × 10 resulting unordered judgment sets is displayed in full in Appendix A: Judgment Sets.

Algorithm 1 Generate 100x10 judgment Sets

```
Create treatments, each has id, category, and gene targets
$Treatments \leftarrow []$, $TxN \leftarrow 1$
for $Cat$ in $[Immunotherapy, Chemotherapy, Targetedtherapy]$ do
    for $iteration = 1, 2, \ldots, 20$ do
        $GeneTargets \leftarrow RandomBelow(2^{30})$
        Push $[ID \leftarrow TxN, Category \leftarrow Cat, GeneTargets]$ onto Treatments
        Increment $TxN$
    end for
end for
Create 100 cases with 10 judgments each
$CasesWithjudgments \leftarrow []$
for $CaseN = 1, 2, \ldots, 100$ do
    Create cases, each has genome and judgments
    $Genome \leftarrow RandomBelow(2^{30})$
    Push $judgmentSet(Genome, Treatments)$ onto $CasesWithjudgments$
end for
Return $CasesWithjudgments$
```

Algorithm 2 $judgmentSet(Genome, Treatments)$: 10 independent judgments for the given case (i.e., Genome)

```
$judgments \leftarrow []$
Score all the treatments against the genome
$ScoredTreatments \leftarrow []$, $BinaryGenome \leftarrow BinaryOf(Genome)$
for $Treatment$ in $Treatments$ do
    Push $Matching01s(BinaryGenome, Treatment.GeneTargets)$ onto $ScoredTreatments$
end for
for $judgmentN = 1, 2, \ldots, 10$ do
    $ScoreSortedTxs \leftarrow Sort(Twiddle(Tweak(ScoredTreatments)))$ by least score first
    Pull the top three either overall, or per category
    if $Random(0 - 1) < 2/3$ then
        Push first three entries from $ScoreSortedTxs$ onto $judgments$
    else
        Push first entry in $ScoreSortedTxs$ from each category onto $judgments$
    end if
end for
Return $judgments$
```

5 Distance metrics

We consider six distance metrics, created by combining one of two counting algorithms with one of three position weighting functions.

5.1 Basic scoring algorithms

Each of the metrics simply counts the number of out-of-order elements that occur between two rankings, weighting them in different ways. The rankings being compared must be of the same length, and contain the same elements. This is *not true* of the rankings that we are comparing in this work. Our case is more complicated as will be discussed in the next section. In this section we assume that the two rankings being compared are the same length and have all the same elements. For example, we might compare $[a, b, c]$ with $[c, b, a]$. The most basic desiderata is that the distance between a list and itself (e.g., $[l, m, n]$ vs $[l, m, n]$) is zero. All of these algorithms satisfy this requirement. Beyond that, however, they may differ, in some cases substantially.

5.1.1 `ssfr`: Symmetric Spearman FootRule

The *Symmetric Spearman FootRule* (`ssfr`) sums the differences in rank position of identical elements, multiplied by the weight of the position of the higher ranked element.

5.1.2 `ltgt`: Less Than, Greater Than

The *Less Than, Greater Than* (`ltgt`) method is the same as `ssfr` except that the difference in the rank positions is not counted, that is, when positions differ, one is added to the distance, again multiplied by the weight of the position of the higher ranked element.

To see the difference between `ssfr` and `ltgt`, consider the comparison of the two ranked lists $[a, b, c]$ and $[c, b, a]$. The swap of a with c would receive a 2 in the `ssfr` algorithm as opposed to just a 1 in the `ltgt` algorithm. (Further, these weights, 1 or 2, would then be multiplied by the position weight of the higher ranked element, which in both cases, is position 1.)

5.2 Position weighting functions

As previously mentioned, when a swap in rank positions is scored, the score is multiplied by the position weight of the higher ranked element. (Note that 1 is the highest ranked position.)

5.2.1 `tailharm`: (Tail) Harmonic

The *Harmonic* position weighting function simply assigns a weight of $1/2^j$ where j is the position $j \in (1, 2, ...)$. (The Tail part of the name will be explained below when describe how we deal with sets that do not contain the same elements.)

5.2.2 `all1`: All One (1)

The *All One* position weighting function simply assigns a weight of 1 to every position.

5.2.3 `rand`: Random

The *Random* position weighting function assigns a random weight between 0 and 1 to each position and this is computed anew for each sub-comparison.

5.3 Ranks with non-identical elements

Because the basic scoring algorithms described above require lists with identical elements (though not in the same order), in order to compare two rankings that do not have identical elements, we compute *the mean score of all possible tail-permuted pairs of "composed" rankings.* First we compute the intersection ($I = R1 \cap R2$) and union ($U = R1 \cup R2$) of the two specialist's rankings $R1$ and $R2$. Then, we compute the difference for each ranking of the ranking with U do that $D1 = U - R1$ and $D2 = U - R2$. Next, we use a weighted footrule between the rankings concatenated with every permutation of Di. Using $Fw(r1, r2)$ as the footrule (where w is the weight vector, which will be described shortly), we have the mean of a large set of Fw for $r1 = I|perm(D1)$, and $r2 = I|perm(D2)$.

Consider the example where the treatments are a through e, and one specialist reports: $R1 = [a, b, c]$ and the other reports: $R2 = [b, d, e]$. $I = \{b\}$, $U = \{a, b, c, d, e\}$. $D1 = [d, e]$ and $D2 = [a, c]$. The entire comparison set is the permutations of $D1$ ($[d, e]$ and $[e, d]$ appended to $r1$, creating $[a, b, c, d, e]$ and $[a, b, c, e, d]$, and the permutations of $D2$ $[a, c]$ and $[c, a]$ appended to $r2$, creating $[b, d, e, a, c]$, and $[b, d, e, c, a]$. Now the four ranked lists contain the same five elements and, as a result, a distance can be computed by any of the six distance functions described in the previous section. The resulting judgment score is then the mean of the scores on the complete (one way) set of combinations of these, that is, the mean of $\binom{4}{2} = 6$ scores. There is one additional minor adjustment for the *Harmonic* (`harm`) position weighting function: the position weights given to the appended elements (the last two in the above above) are equal to $1/(2^{(l+2)})$, where l is the last position of the original base set, 3 in this case. For the above example, the harmonic position weights for $[a, b, c, d, e]$ are: $[0.5, 0.25, 0.125, 0.03125, 0.03125]$.

6 Judgment Concordance

Finally, given that we can create a distance between specialists that have non-identical elements, the concordance over the 10 judgments for the case is simply the mean of all possible judgments scored against one another. The results of this computation are reported below each set in Appendix A: Judgment Sets. Note the legend indicating which cell in the scores arises from which combination of counting algorithm and position weight function. For example, `tailharmpws.ssfr`, the upper-left score, which has a value of 0.49 for judgment set 0 (the first set), is the *Symmetric Spearman Footrule* with the `tailharm` position weight function.

The judgment sets with the lowest concordance score, called "Near", vs. the highest concordance score, called "Far" are summarized in a table just after the long table. This enables one to quickly compare the `footrule` and `ltgt` counting algorithms with one another over the three different position weighting functions. One can immediately see that in all cases, set 77 has the lowest concordance score, and in examining this set in the table one can see why this makes sense, because the judgments are identical. Less obvious are why the sets with "Far" concordances have their values. Although there is some disagreement between methods at this end, there is also commonality. For example, set 76 (conveniently although entirely coincidentally, next to 77) is the most common "farthest" set. In looking at that set one can see that it is, to use a technical term, "all over the place".

7 Interpretation and Use of Concordances

The reason for computing a concordance score over multiple specialists is to get a sense of the extent to which specialists agree about the case. This, then, can be used to modulate our assessment of the degree to which the ranking algorithm should be penalized when it disagrees with a given judgment (or a consensus judgment which a tumor board of specialists typically produces). If the specialists cannot agree with one another, then we should not expect our algorithm to agree with any one of them, nor should we give strong credence to a consensus ranking that was reported from a team that cannot agree. In this latter sense, the concordance score is akin to the σ in a Normal distribution which gives us a sense of how much weight we should attach to the mean. In the next section we examine the application of these methods to this problem, and in a later section, fold in the concordance score developed above.

8 Assessing Algorithmic Ranking Agreement with judgment Consensus

Recall that we asserted that, without loss of generality, we could assume the algorithm always returned the 3-rank: $[1, 2, 3]$. We assume further, that in order to assess how well the algorithm is doing in agreeing with specialist rankings, we compare the algorithm's 3-rank with a single specialist-derived 3-rank, which may have been reported by a single specialist, yet more likely is the work of multiple specialists (e.g., a tumor board). Thus, we compute a concordance score, from the previous section, between the algorithm's 3-rank and the one specialist's 3-rank. Since we are only comparing two 3-ranks at a time, we assume, again without loss of generality as the actual numbers are treatment IDs and have no information content in-and-of themselves, that the specialist 3-rank is selected randomly without replacement from the set: $[1, 2, 3, 4, 5, 6]$. Indeed, in order to get a visual sense of the difference between the algorithm's 3-rank and the specialist's 3-rank, we create a "exchange icon" (hereafter "xcon") telling us which element from one 3-rank went to which position in the other. For example, the *xcon* describing the comparison between $[a, b, c]$ and itself is just "x123", and the xcon describing the comparison between $[a, b, c]$ and $[c, b, a]$ is "x321". When an element is missing (or, conversely, a new one is added on the opposite 3-rank) we use a dash (`-`), so that the xcon describing the comparison between $[a, b, c]$ and $[c, e, d]$ is `x3--`, and if all three elements differ: `x---`.

There are a finite and not very large set of possible xcons for a given n-rank, and, indeed, for 3-ranks there are 34. Appendix B: Permutation Rankings despicts these 34 xcons as well as the head-to-head comparison score for the six methods described in detail above. In this case, each pair of columns is sorted from lowest (nearest) to highest (farthest) score. A quick scan of this table shows that the distance between a 3-rank and itself is 0.0. This data comes not from enumerating the permutations and scoring each once, but instead from choosing 10,000 random permutations and reporting the mean of the scores for each xcon. We use 10,000 random permutations because in the random position weight case, the computation is not deterministic. Indeed, one can easily see in Fig. 1 that the number of equi-score sets, sorted by score, is not evenly distributed between the cases.[7]

The `random pws` weighting function, the red line of Fig. 1, provides a rather continuous function as expected. The remaining five weighting functions are deterministic and thus created step functions as expected. Nevertheless, Fig. 1 shows that these step functions vary in terms of the number of values they take on, and in the size of the steps, i.e., the equi-score sets, as tabulated below:

The method to obtain the best distinction among xcons is one of the random methods, which simulate continuous distance. However, in a real application this may not be advisable for an n-rank

[7] For display purposed, Fig. 1 is down-sampled by a fifth 1/5 to plot a more manageable number of points.

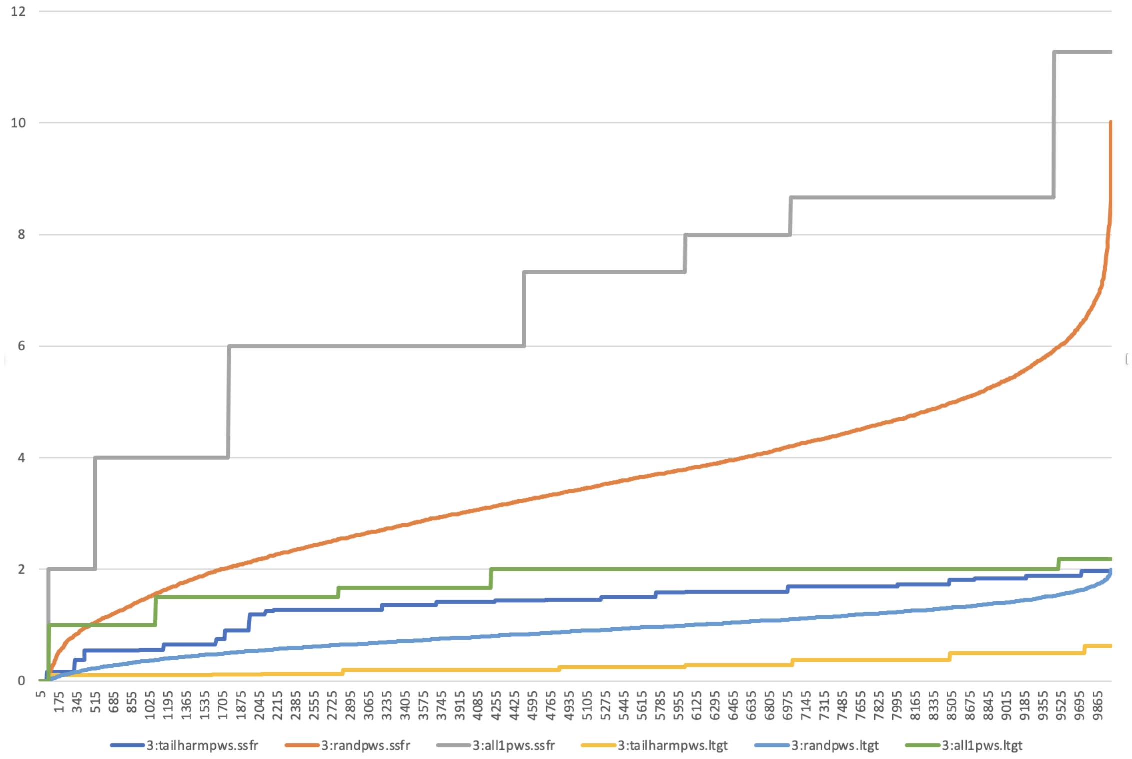

Figure 1: Sorted Concordances for 100 specialist Sets at 3-,4-, and 5-Ranks. Dependent axis is unnormalized score by each algorithm (colors, details of algorithms in text). As points are sorted per Y min-to-max, the X axis simply denotes final ordinal position, and is labeled for reference value only.

method	**no. equi-score sets**	**size of largest set**
tailharmpws.ssfr	26	2
all1pws.ssfr	8	10
tailharmpws.ltgt	10	7
all1pws.ltgt	6	15

Table 2: Equi-Score and Size of Largest Sets for each Method

with small n, because it is less likely one could do enough computations to smooth the presence of coincidental ties which throw off the scores. When $n \geq 4$, there are enough comparisons in the cross-product. On the other hand, one could, recompute the score many times for the same set in order to smooth out anomalous ties. Overall, then, if one is able to use a random algorithm, the best dynamic range[8] is obtained with `randpws.ssfr` weighting. The `tailharmpws.ssfr` appears to be the preferred method if one desires more and finer-grained equi-score sets.

[8] The dynamic range is merely a function of scaling of the parameters in the algorithms, but the number of size of equi-score sets is not a simple function of scale.

8.1 Ordering of the "Far" 3-Rank Contrasts

We now discuss the ordering of the "far" cases. We have the intuition that `x---` should be the worst case, after all, the consensus ranking disagreed with the algorithm entirely, choosing a completely different set of treatments. Unfortunately, the distances delivered by our preferred method, `tailharm.ssfr`, places `x---` fourth from the end, rating `x-31`, `x3-2`, and `x-32` worse than complete disagreement (i.e., `x---`). What is `tailharm.ssfr` capturing (or not capturing) here? Digging deeper reveals that for `tailharm.ssfr`, the cases that scored worse than `x---` are cases where the last (i.e., third) option was brought to the front, and/or above the top-ranked (first) item. Contrast this with the top of the `tailharm.ssfr` list: `x123`, `x120`, `x132`, `x13-`, `x1-2`, `x1-3`, `x1--`. In all these cases the top-ranked treatment matches. The `tailharm.ssfr` weighting emphasizes the top of the ranked list. It is less concerned with the bottom of the lists (in this 3-rank case, rank positions third and beyond). Are there applications for which "breaking rank" (i.e., putting the top-ranked treatment in some place other than the top rank) is considered worse than choosing a completely different set of treatments?

9 Consensus vs. Concordance

In the introduction we described an application of this measure to the problem of estimating how good an AI-based treatment ranking algorithm is. To reiterate, such an algorithm might be given an input describing the case, a list of plausible treatments, and training data in the form of a list of prior cases including what treatments were utilized in prior cases and the outcomes, if available. The algorithm will use the training data to produce a 3-rank for the patient case.

To simulate this situation, we created Algorithm 3, which creates is a heuristic consensus 3-rank from 100 3-rank judgment sets from ten specialists.

Algorithm 3 *Consensus*(*JudgmentSet*)

```
function consensus(jset)
dict=Empty Dictionary
n = length(jset[1]) # Number of elements in each judgement (i.e., n-rank)
for ranking in jset do
    for i in 1...n do
        tx = ranking[i] # ith treatment in this ranking
        inc=(n-i) # Position-adjusted increment for this tx
        dict[tx]++=1+inc # (dict[tx]=0 if not yet encountered)
    end for
end for
Return dict (i.e., tx's) sorted by sum of position adjusted inc's
```

Figure 2 depicts the mean (and standard error) of the `tailharmpws.ssfr` distance between this consensus and each of the ten judgments. Unsurprisingly, the mean consensus distance is highly correlated with the internal concordance distance ($R^2 = 0.964$).

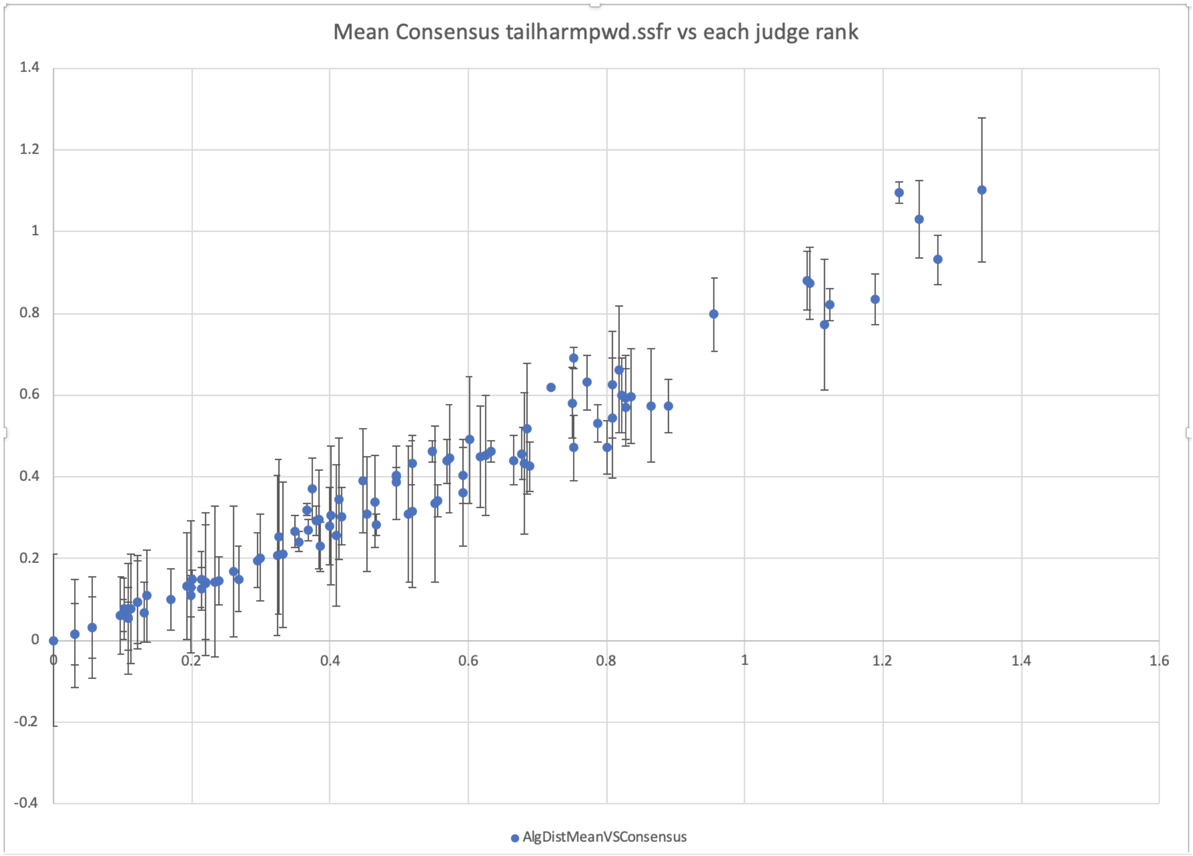

Figure 2: Comparison of Internal Concordance Score vs Mean and Standard Error of Consensus Distance for each of the 100 ten-specialist 3-rank judgment sets.

10 Summary and Contra(Indication) Considerations

Non-parametric rank correlation measures, such as the Kendall-Tau or Spearman coefficients are commonly used to compare two ranked lists, but in many areas, such as medicine, the rankings often will not contain all the same elements. There are several ways that one could deal with this. One is to assign unshared items to the $n+1^{st}$ rank, another is to assign such items to the bottom of all possibilities. Neither of these options seems right for our application of treatment ranking. The former applies a very minimal cost to non-appearance, and the latter is impracticable because the list of possible, or even plausible, treatments is vague, and potentially very long.

Another complexity is that treatments can be understood in a multi-dimensional space. The simplest form of this multi-dimensionality arises in terms of different sorts of treatments, for example, in cancer there are chemotherapies, immunotherapies, radiotherapies, targeted therapies, and combinations of these.[9] A specialist might use these categorizations in various ways. For example, a specialist might first rank the appropriateness of a category for a given patient at the high level, e.g.,"For this patient we would consider chemotherapy, then immunotherapy, then targeted therapy, in that order of preference." The specialist then ranks the specific treatments within those categories

Assume for ease of exposition that there are 100 plausible treatment regimens for a given diagnosis. From this 100, there are many ways to create a list of 3 pros and 2 cons. The list below contains some common ways, although it is certainly not exhaustive.

1. Rank all 100; Report the top 3 and the bottom 2 as a single ranking.
2. Rank all 100; Report the top 3 and the bottom 2 as separate pro and con lists.
3. Rank all 100; Select the top n (e.g., 20); then apply methods 1 or 2, as above.
4. Filter, based on rule(s) to n (e.g., $n = 20$) creating an unranked subset of treatments. Then apply methods 1 or 2.
5. As above but consider "categorical confluxity", e.g., "don't do any chemo".
6. Rank the pros, and then re-rank the whole set for cons (report 3 of each as 1 or 2).
7. As any of the above, but the con list is not reported as ranked (regardless of how it is selected).
8. As above but also produce a "middle ground" list containing all (or some of) the items that are neither good enough to go into the pro list, nor bad enough to go into the con list. This middle ground list may contain all remaining (95) items, or, more reasonably, a set of plausible treatments that are not good enough to be in the top-3, nor bad enough to be considered contra-indicated.
9. Applying any of the above methods to all treatments *in a category*, reporting out 3 pros and 2 cons for each cateogry.

At the outset we described both an *indicated* list (i.e., pros) and a *contra-indicated* list (i.e., cons). Thus far we have dealt only with the pros. There are several reasons for this. First, usually the con

[9] We have assumed throughout that combinations are simply another separate treatment regimen, although this is clearly not a fully satisfactory way to understand and treat combinations.

list will not be ranked so it is unordered. It is not worth deciding which of a set of contra-indicated treatments is worse than others, whereas it *is* worth doing so with the pro choices, since presumably one of them, the top one, will be acted upon. On the other hand, the assumption is that none of the con options will be selected, so it isn't worth ranking the cons.

Another even more complex difficulty arises from how the con list is actually created. One way to think about it is that there is a much longer list of, say, 100 possible treatment options. The specialist ranks the entire list, and reports two sets: the top n (again, in our application, n is usually 3), forming the 3-rank and the bottom b (say, 2) cons. The result is a merged list of 5 where the first 3 are very good and the last 2 are very poor, leaving an invisible quality gap of unknown size between these.

However, it is unlikely, due to the cognitive load and time requirement, that specialists rank all 100 options. In a different, more reasonable process, the specialist first approximately ranks all 100 options, and excludes from further consideration all but, say, 20 reasonable ones. The specialist then undertakes the process described in the previous paragraph, but only on the 20 reasonable options. In this case, the list of 3 pros plus 2 cons has a narrower invisible, unknown quality gap. A further complication is that same specialist might just report 5 pro options and no con options.

There is an additional problem when considering the presentation of cons. Whereas the sense of what "pro" means is relatively consistent across different decision processes, the meaning of con can vary. This does not matter so much in practice because, as mentioned above, the patient presumably ends up doing something that is pro for them, not something that is bad (con) for them (leaving aside complexities introduced by combination treatments). However, in the example goal given at the very beginning of this long run-on paragraph of validating that an algorithm gives rankings that are comparable to that of a specialist, even assuming that the algorithm uses a comparable decision process, the validation statistic must be designed to score "correct" rankings better than "incorrect" ones, and retain appropriate properties.

As a result of this complexity, we have chosen above, and here to treat the con list as a completely separate and unranked set of contra-indicated treatments. In other words, we assume that in determining appropriate treatments for a patient, a specialist has a process for creating the pro list and a separate process for creating the con list. Since cons are, by assumption, unranked, we use the `all1` versions of our distance metric to compare them, because this does not deferentially consider the position of elements in the ranking.

There may be applications which require the evaluation of *one* AI model that creates both a pro list of indicated treatments *and* a con list of contra-indicated treatments. The two scalar measures, one that measures how far the AI's pro list is from the expert recommendations and one that measures how far the AI's con list is from the expert recommendations, must be combined. A few basic methods for combining the pro score and the con score follow. Method 1: ask the application user for convex weights λ_1 and λ_2 such that $\lambda_1 + \lambda_2 = 1$, where λ_1 is the weight associated with the score for the pro list and λ_2 is the weight associated with the score for the pro list. For example, $\lambda_1 = .5$ and $\lambda_2 = .5$ means the pro and con scores are given equal weight. Whereas $\lambda_1 = .75$ and $\lambda_2 = .25$ means the AI system places three times as much priority on getting the pro list correct as it does on the con list. Method 2: weight each score (pro, con) by the number of items in the list. For example, since the pro list from our running examples contains 3 items while the con list contains only 2, weight $\lambda_1 = \frac{3}{3+2} = \frac{3}{5}$ and $\lambda_2 = \frac{2}{3+2} = \frac{2}{5}$.

Finally, we note that the types of errors made by the AI system might carry different penalties that can be differentiated by more sophisticated weighting schemes. There are at least six different types of errors that an AI system might make when comparing its rankings to the “correct” rankings provided by the expert lists.

- A *pro inclusion error* occurs when the AI’s Pro-3 list contains an item, i.e, a treatment, that does not appear on the expert’s Pro-3 list.

- A *con inclusion error* occurs when the AI’s Con-2 list contains an item that does not appear on the experts Con-2 list.

- A *pro exclusion error* occurs when the AI’s Pro-3 list does not contain an item that does appears on the expert’s Pro-3 list.

- A *con exclusion error* occurs when the AI’s Con-2 list does not contain an item that appears on the expert’s Con-2 list.

- An *above-the-fold positional error* occurs when the AI Pro-3 list contains an item from the expert Pro-3 list, but it is in an incorrect rank position.

- A *below-the-fold positional error* occurs when the AI Pro list contains an item from the expert Pro-3 list, but it does not appear in the top 3 rank positions.

11 Acknowledgements

We acknowledge ideas and feedback from our xCures colleagues Mark Shapiro, Asher Wasserman, Glenn Kramer, and Jameson Quinn.

References

[FKM$^+$04] Ronald Fagin, Ravi Kumar, Mohammad Mahdian, D. Sivakumar, and Erik Vee. Comparing and aggregating rankings with ties. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems - PODS '04*, page 47, Paris, France, 2004. ACM Press.

[Lee98] Joon Ho Lee. Combining the evidence of different relevance feedback methods for information retrieval. *Information Processing & Management*, 34(6):681–691, November 1998.

A 100 judgment Sets with Concordance and Consensus Scores

Each block (0, 1, ..., 100) is a judgment set. The top row (with the set number in the left column) displays the (unordered) 3-ranks composing that judgment set. Just below each judgment set are the concordances computed by each of the six distance metrics (two: ssfr vs ltgt, by three: tailharmpws vs all1pws vs randompws).

To the right of the six concordance scores in the main table is the consensus 3-rank, and the mean and standard deviation of the consensus vs each of the specialist 3-ranks, by tailharmpws.ssfr.

Following the main table is a summary table giving the nearest/farthest judgments sets by each distance metric.

Legend for concordance results subtable						
	tailharmpws.ssfr	all1pws.ssfr	randpws.ssfr	Consensus	CxMean	CxDev
	tailharmpws.ltgt	all1pws.ltgt	randpws.ltgt		(above: tailharmpws.ssfr)	

0	[2, 3, 4] [2, 3, 1]	[2, 21, 41] [2, 21, 41]	[2, 3, 4] [2, 31, 41]	[2, 28, 41] [2, 31, 41]	[2, 3, 21] [2, 31, 41]	
	0.495	4.311	2.076	[2, 3, 41]	0.4	0.21
	0.112	1.237	0.581			
1	[10, 9, 48] [10, 32, 48]	[10, 43, 50] [9, 32, 48]	[32, 4, 10] [10, 32, 43]	[10, 48, 9] [10, 32, 9]	[10, 43, 4] [10, 32, 48]	
	0.889	5.449	2.781	[10, 32, 9]	0.573	0.51
	0.197	1.523	0.718			
2	[4, 28, 12] [28, 4, 55]	[28, 4, 55] [28, 4, 2]	[28, 4, 12] [28, 4, 55]	[28, 4, 12] [28, 4, 2]	[28, 4, 55] [28, 4, 12]	
	0.261	1.822	0.93	[28, 4, 55]	0.169	0.271
	0.189	0.911	0.406			
3	[9, 45, 37] [45, 9, 37]	[9, 37, 45] [9, 45, 37]	[9, 37, 45] [45, 37, 9]	[9, 37, 45] [9, 37, 45]	[45, 37, 9] [37, 9, 45]	
	0.75	2.578	1.275	[9, 45, 37]	0.581	0.469
	0.306	0.956	0.484			
4	[33, 36, 20] [36, 33, 6]	[36, 20, 45] [33, 36, 29]	[33, 36, 6] [36, 7, 45]	[33, 36, 6] [33, 36, 29]	[33, 20, 45] [33, 7, 45]	
	0.809	5.022	2.492	[33, 36, 20]	0.544	0.514
	0.216	1.493	0.745			
5	[51, 14, 36] [51, 13, 14]	[51, 14, 36] [51, 14, 45]	[51, 14, 13] [51, 14, 27]	[51, 14, 13] [51, 14, 13]	[51, 14, 36] [51, 14, 45]	
	0.199	1.956	0.843	[51, 14, 13]	0.131	0.113
	0.131	0.911	0.495			
6	[22, 17, 43] [22, 9, 43]	[22, 43, 17] [22, 43, 17]	[22, 43, 9] [22, 17, 43]	[22, 9, 17] [22, 17, 43]	[22, 9, 43] [22, 43, 17]	
	0.38	2.667	1.287	[22, 43, 17]	0.294	0.235
	0.172	0.989	0.505			
7	[20, 33, 57] [20, 33, 12]	[20, 33, 12] [20, 33, 55]	[20, 33, 12] [20, 33, 57]	[20, 33, 12] [20, 33, 57]	[20, 33, 12] [20, 12, 33]	
	0.17	1.6	0.758	[20, 33, 12]	0.1	0.124
	0.114	0.756	0.367			
8	[23, 41, 5] [23, 41, 36]	[23, 41, 36] [23, 41, 5]	[23, 41, 5] [23, 41, 5]	[23, 41, 52] [23, 41, 17]	[23, 41, 5] [23, 41, 5]	
	0.101	1.289	0.664	[23, 41, 5]	0.062	0.081
	0.081	0.644	0.32			
9	[53, 20, 32] [53, 45, 54]	[53, 45, 14] [53, 20, 24]	[53, 46, 45] [53, 46, 54]	[53, 10, 24] [53, 54, 46]	[53, 54, 46] [53, 20, 23]	
	0.548	4.889	2.282	[53, 20, 54]	0.463	0.215

	0.138	1.474	0.724			
10	[3, 48, 23] [3, 19, 8]	[3, 19, 8] [3, 8, 19]	[3, 19, 48] [3, 19, 48]	[3, 19, 43] [3, 19, 43]	[3, 19, 5] [3, 19, 43]	
	0.294	2.756	1.317	[3, 19, 8]	0.196	0.189
	0.137	1.081	0.538			
11	[22, 8, 45] [22, 45, 8]	[22, 8, 41] [22, 41, 8]	[8, 22, 45] [22, 8, 45]	[22, 8, 41] [22, 8, 45]	[22, 8, 41] [22, 8, 41]	
	0.385	2.267	1.108	[22, 8, 41]	0.23	0.299
	0.225	1.0	0.49			
12	[17, 9, 24] [17, 9, 46]	[17, 9, 46] [9, 24, 46]	[17, 9, 46] [9, 24, 46]	[9, 17, 46] [9, 17, 46]	[17, 24, 46] [17, 24, 46]	
	0.828	3.778	1.967	[17, 9, 46]	0.569	0.555
	0.258	1.089	0.459			
13	[17, 39, 60] [17, 39, 26]	[17, 39, 26] [17, 39, 60]	[17, 39, 60] [17, 39, 26]	[17, 39, 60] [17, 39, 26]	[17, 26, 39] [17, 26, 39]	
	0.219	1.778	1.001	[17, 39, 26]	0.138	0.145
	0.133	0.8	0.383			
14	[1, 8, 39] [1, 8, 54]	[8, 1, 3] [8, 45, 30]	[8, 1, 3] [1, 54, 31]	[1, 45, 31] [1, 8, 3]	[1, 8, 3] [1, 54, 39]	
	0.787	5.049	2.461	[1, 8, 54]	0.53	0.417
	0.234	1.534	0.769			
15	[41, 39, 19] [41, 39, 19]	[41, 39, 19] [41, 39, 19]	[41, 39, 19] [41, 39, 18]	[41, 39, 19] [41, 39, 19]	[41, 39, 19] [41, 39, 19]	
	0.031	0.4	0.206	[41, 39, 19]	0.016	0.049
	0.025	0.2	0.128			
16	[31, 2, 27] [31, 2, 27]	[31, 2, 43] [31, 2, 43]	[31, 2, 51] [31, 2, 43]	[31, 2, 51] [31, 2, 51]	[31, 2, 43] [31, 2, 43]	
	0.108	1.378	0.754	[31, 2, 43]	0.078	0.082
	0.086	0.689	0.398			
17	[7, 20, 19] [7, 33, 55]	[7, 19, 33] [7, 33, 20]	[7, 33, 55] [7, 33, 19]	[7, 33, 55] [7, 33, 55]	[7, 33, 18] [7, 19, 18]	
	0.354	3.156	1.652	[7, 33, 19]	0.242	0.187
	0.137	1.126	0.548			
18	[43, 19, 17] [43, 17, 32]	[43, 19, 17] [43, 19, 17]	[43, 19, 17] [43, 17, 19]	[43, 17, 19] [43, 19, 18]	[43, 19, 17] [43, 19, 17]	
	0.239	1.733	0.816	[43, 19, 17]	0.145	0.209
	0.136	0.748	0.398			
19	[16, 35, 58] [16, 35, 58]	[16, 58, 35] [16, 35, 58]	[16, 35, 58] [58, 16, 35]	[7, 35, 58] [16, 35, 58]	[35, 58, 16] [16, 58, 35]	
	0.801	3.022	1.557	[16, 35, 58]	0.472	0.616
	0.261	0.967	0.513			
20	[19, 3, 40] [19, 40, 3]	[19, 40, 48] [19, 20, 40]	[19, 40, 3] [19, 40, 20]	[19, 20, 40] [19, 40, 20]	[19, 40, 5] [19, 40, 55]	
	0.323	2.622	1.428	[19, 40, 20]	0.208	0.173
	0.175	1.1	0.519			
21	[54, 36, 45] [54, 36, 45]	[54, 45, 39] [54, 36, 13]	[54, 39, 51] [54, 51, 36]	[54, 51, 39] [54, 51, 36]	[54, 39, 1] [54, 40, 39]	
	0.518	4.4	2.236	[54, 36, 39]	0.434	0.192
	0.159	1.404	0.785			
22	[8, 3, 1] [8, 3, 1]	[3, 8, 31] [8, 27, 59]	[8, 31, 59] [8, 31, 51]	[8, 3, 1] [8, 3, 31]	[8, 31, 41] [3, 8, 31]	
	0.688	4.37	2.17	[8, 3, 31]	0.426	0.279
	0.259	1.478	0.721			
23	[24, 18, 26]	[18, 24, 55]	[24, 26, 18]	[18, 28, 41]	[26, 24, 28]	

	[18, 28, 55]	[24, 26, 28]	[18, 24, 41]	[24, 18, 55]	[28, 24, 18]	
	1.094	5.615	2.965	[24, 18, 28]	0.875	0.506
	0.31	1.685	0.887			
24	[11, 40, 55] [51, 11, 40]	[11, 51, 40] [40, 51, 11]	[11, 40, 51] [11, 40, 51]	[55, 11, 40] [55, 11, 40]	[11, 51, 55] [51, 55, 11]	
	1.116	4.756	2.248	[11, 51, 40]	0.772	0.562
	0.364	1.511	0.729			
25	[25, 17, 54] [25, 54, 17]	[25, 17, 31] [25, 54, 31]	[25, 39, 31] [25, 17, 54]	[25, 17, 54] [25, 17, 54]	[25, 17, 54] [25, 54, 17]	
	0.331	2.578	1.243	[25, 17, 54]	0.21	0.254
	0.14	0.937	0.49			
26	[23, 44, 6] [44, 23, 6]	[23, 44, 6] [23, 44, 6]	[23, 44, 6] [23, 44, 6]	[44, 23, 6] [23, 44, 6]	[23, 44, 6] [23, 44, 6]	
	0.267	0.711	0.427	[23, 44, 6]	0.15	0.316
	0.178	0.356	0.182			
27	[17, 22, 52] [17, 22, 55]	[17, 22, 8] [17, 22, 41]	[17, 22, 41] [17, 22, 41]	[17, 22, 41] [17, 22, 52]	[17, 22, 52] [17, 22, 55]	
	0.122	1.556	0.791	[17, 22, 41]	0.094	0.081
	0.097	0.778	0.409			
28	[44, 45, 40] [44, 45, 53]	[45, 44, 40] [44, 45, 53]	[45, 44, 53] [44, 45, 40]	[44, 1, 36] [44, 1, 40]	[44, 1, 36] [44, 1, 36]	
	0.633	4.207	2.094	[44, 45, 1]	0.463	0.298
	0.216	1.352	0.674			
29	[24, 55, 32] [24, 55, 50]	[24, 32, 55] [24, 32, 55]	[24, 50, 32] [24, 55, 50]	[24, 50, 6] [24, 55, 32]	[24, 55, 6] [24, 32, 55]	
	0.399	3.156	1.44	[24, 55, 32]	0.28	0.222
	0.171	1.156	0.603			
30	[54, 20, 28] [54, 43, 19]	[54, 19, 28] [54, 19, 28]	[54, 19, 28] [54, 20, 28]	[54, 20, 28] [54, 20, 28]	[54, 43, 20] [54, 20, 43]	
	0.416	3.378	1.524	[54, 20, 28]	0.304	0.292
	0.123	1.037	0.508			
31	[53, 40, 20] [20, 53, 40]	[20, 40, 53] [40, 20, 53]	[53, 40, 20] [40, 53, 20]	[20, 40, 53] [20, 40, 53]	[20, 40, 53] [53, 20, 40]	
	0.822	2.756	1.303	[20, 40, 53]	0.6	0.582
	0.328	1.0	0.508			
32	[54, 53, 29] [54, 29, 5]	[54, 53, 29] [54, 29, 53]	[54, 29, 53] [54, 29, 53]	[54, 29, 5] [54, 53, 29]	[54, 29, 53] [54, 29, 53]	
	0.233	1.644	0.892	[54, 29, 53]	0.144	0.171
	0.144	0.756	0.363			
33	[44, 48, 33] [45, 44, 48]	[44, 33, 3] [44, 45, 33]	[44, 45, 22] [44, 45, 33]	[44, 33, 3] [44, 3, 33]	[44, 33, 3] [44, 33, 3]	
	0.569	3.793	1.941	[44, 33, 45]	0.438	0.367
	0.208	1.267	0.611			
34	[3, 59, 47] [59, 3, 47]	[3, 47, 29] [3, 47, 32]	[59, 3, 47] [3, 47, 32]	[59, 47, 29] [3, 47, 59]	[3, 47, 59] [59, 47, 3]	
	0.835	3.793	1.802	[3, 47, 59]	0.597	0.604
	0.277	1.285	0.692			
35	[2, 55, 8] [2, 55, 37]	[2, 55, 8] [2, 55, 37]	[2, 55, 8] [2, 55, 37]	[55, 4, 2] [2, 55, 8]	[2, 55, 37] [55, 17, 2]	
	0.551	2.933	1.544	[2, 55, 8]	0.334	0.545
	0.225	1.089	0.634			
36	[11, 45, 37] [11, 45, 37]	[11, 45, 37] [45, 11, 37]	[11, 45, 37] [45, 11, 37]	[11, 45, 37] [11, 45, 42]	[45, 11, 43] [11, 45, 37]	
	0.409	1.689	0.895	[11, 45, 37]	0.256	0.382

	0.281	0.844	0.424			
37	[3, 45, 27] [3, 45, 36]	[3, 36, 27] [3, 27, 36]	[3, 36, 27] [3, 36, 27]	[3, 27, 36] [3, 45, 36]	[3, 27, 45] [3, 27, 36]	
	0.383	2.711	1.435	[3, 27, 36]	0.295	0.237
	0.172	1.0	0.508			
38	[33, 44, 57] [33, 44, 57]	[33, 44, 12] [33, 44, 57]	[33, 44, 57] [33, 44, 12]	[33, 44, 12] [33, 44, 12]	[33, 44, 27] [33, 44, 12]	
	0.101	1.289	0.639	[33, 44, 12]	0.078	0.082
	0.081	0.644	0.324			
39	[7, 4, 2] [4, 38, 55]	[7, 4, 2] [4, 7, 38]	[7, 38, 55] [7, 4, 2]	[4, 7, 38] [4, 38, 55]	[7, 4, 2] [7, 4, 38]	
	0.753	4.059	1.971	[4, 7, 38]	0.691	0.44
	0.282	1.381	0.671			
40	[24, 1, 27] [1, 27, 24]	[1, 24, 27] [1, 27, 24]	[27, 24, 3] [1, 24, 27]	[1, 24, 52] [27, 24, 1]	[1, 24, 52] [24, 27, 1]	
	0.865	3.704	1.862	[1, 24, 27]	0.575	0.559
	0.322	1.307	0.704			
41	[11, 26, 55] [24, 28, 26]	[11, 26, 55] [26, 11, 39]	[11, 24, 55] [41, 26, 24]	[55, 39, 24] [24, 41, 55]	[11, 55, 39] [24, 41, 11]	
	1.343	7.107	3.558	[11, 24, 26]	1.102	0.588
	0.245	1.794	0.907			
42	[39, 50, 16] [39, 50, 32]	[50, 39, 32] [50, 39, 32]	[39, 50, 32] [39, 50, 15]	[39, 50, 32] [39, 50, 5]	[39, 50, 32] [50, 32, 39]	
	0.519	2.4	1.156	[39, 50, 32]	0.316	0.423
	0.303	1.089	0.495			
43	[36, 6, 43] [36, 6, 9]	[36, 6, 10] [36, 6, 54]	[36, 6, 9] [36, 6, 10]	[36, 6, 10] [36, 6, 9]	[36, 6, 10] [36, 6, 10]	
	0.111	1.422	0.69	[36, 6, 10]	0.078	0.082
	0.089	0.711	0.4			
44	[44, 38, 7] [38, 44, 7]	[44, 38, 35] [38, 7, 44]	[44, 38, 7] [44, 38, 7]	[44, 38, 7] [44, 38, 7]	[38, 44, 7] [44, 38, 7]	
	0.466	1.733	0.869	[44, 38, 7]	0.284	0.441
	0.267	0.778	0.328			
45	[31, 6, 52] [31, 6, 52]	[31, 6, 20] [31, 6, 23]	[31, 6, 20] [31, 6, 52]	[31, 6, 22] [31, 6, 22]	[31, 23, 20] [31, 6, 52]	
	0.22	2.311	1.298	[31, 6, 52]	0.143	0.193
	0.104	0.922	0.478			
46	[41, 29, 5] [19, 29, 60]	[19, 29, 60] [19, 29, 41]	[19, 60, 5] [19, 60, 5]	[5, 60, 41] [41, 19, 5]	[29, 19, 60] [29, 5, 19]	
	1.189	5.852	3.075	[19, 29, 5]	0.834	0.597
	0.284	1.63	0.726			
47	[16, 50, 35] [16, 47, 50]	[16, 47, 50] [16, 47, 50]	[16, 47, 27] [16, 50, 47]	[16, 47, 36] [16, 50, 47]	[16, 50, 27] [16, 50, 47]	
	0.325	2.578	1.173	[16, 50, 47]	0.253	0.218
	0.166	1.044	0.554			
48	[6, 16, 56] [6, 16, 44]	[6, 16, 44] [6, 16, 12]	[6, 56, 35] [6, 16, 12]	[6, 16, 56] [6, 16, 44]	[6, 16, 56] [6, 16, 44]	
	0.213	2.222	1.075	[6, 16, 56]	0.148	0.158
	0.104	0.9	0.451			
49	[47, 41, 23] [47, 23, 1]	[47, 41, 23] [47, 41, 23]	[47, 41, 23] [47, 41, 23]	[47, 23, 5] [47, 41, 23]	[47, 41, 23] [47, 41, 23]	
	0.198	1.467	0.647	[47, 41, 23]	0.109	0.231
	0.092	0.556	0.345			
50	[6, 19, 20]	[20, 19, 6]	[19, 45, 22]	[19, 45, 22]	[6, 20, 19]	

	[19, 20, 6]	[6, 45, 22]	[20, 45, 22]	[19, 20, 6]	[6, 19, 20]	
	1.091	5.526	2.951	[19, 6, 20]	0.88	0.412
	0.232	1.463	0.739			
51	[23, 48, 43]	[23, 48, 19]	[23, 48, 19]	[23, 48, 19]	[23, 48, 43]	
	[23, 48, 19]	[23, 43, 48]	[23, 48, 52]	[23, 48, 52]	[23, 48, 52]	
	0.192	1.867	1.063	[23, 48, 19]	0.133	0.165
	0.122	0.856	0.403			
52	[31, 9, 56]	[31, 20, 47]	[31, 20, 45]	[31, 20, 45]	[31, 20, 56]	
	[31, 20, 47]	[31, 20, 26]	[31, 20, 47]	[31, 20, 47]	[31, 20, 47]	
	0.214	2.267	1.093	[31, 20, 47]	0.127	0.198
	0.093	0.874	0.44			
53	[27, 15, 56]	[35, 27, 16]	[27, 35, 56]	[27, 15, 56]	[27, 35, 16]	
	[27, 43, 17]	[27, 15, 56]	[27, 35, 16]	[27, 35, 17]	[35, 27, 15]	
	0.678	4.459	2.114	[27, 35, 15]	0.457	0.281
	0.236	1.444	0.7			
54	[40, 29, 20]	[29, 20, 40]	[20, 40, 29]	[29, 20, 40]	[29, 20, 45]	
	[40, 29, 20]	[20, 29, 45]	[20, 29, 45]	[40, 29, 20]	[40, 20, 29]	
	0.956	3.778	1.748	[29, 20, 40]	0.797	0.529
	0.361	1.333	0.647			
55	[7, 27, 5]	[7, 23, 27]	[7, 27, 23]	[27, 7, 23]	[7, 27, 23]	
	[27, 7, 23]	[27, 7, 23]	[7, 27, 47]	[7, 27, 41]	[7, 27, 23]	
	0.514	2.4	1.157	[7, 27, 23]	0.309	0.324
	0.325	1.133	0.58			
56	[15, 39, 19]	[20, 15, 21]	[15, 39, 44]	[20, 39, 41]	[20, 15, 19]	
	[15, 20, 39]	[15, 20, 19]	[20, 39, 53]	[20, 15, 39]	[20, 15, 39]	
	0.828	4.622	2.367	[20, 15, 39]	0.594	0.541
	0.29	1.544	0.698			
57	[26, 47, 2]	[26, 47, 2]	[26, 38, 35]	[26, 35, 38]	[26, 47, 2]	
	[26, 38, 35]	[26, 38, 35]	[26, 38, 35]	[26, 35, 38]	[26, 35, 38]	
	0.401	3.333	1.617	[26, 38, 35]	0.306	0.286
	0.12	1.044	0.511			
58	[58, 48, 60]	[58, 23, 1]	[58, 23, 1]	[58, 48, 23]	[58, 60, 48]	
	[58, 30, 23]	[58, 23, 60]	[58, 60, 30]	[58, 23, 30]	[58, 30, 14]	
	0.496	4.222	2.19	[58, 23, 60]	0.386	0.238
	0.163	1.385	0.658			
59	[16, 23, 58]	[16, 58, 23]	[16, 30, 58]	[16, 30, 58]	[16, 23, 58]	
	[16, 58, 23]	[16, 30, 23]	[16, 23, 58]	[16, 30, 58]	[16, 23, 58]	
	0.374	2.622	1.159	[16, 58, 23]	0.37	0.212
	0.147	0.889	0.496			
60	[26, 28, 55]	[28, 26, 37]	[28, 26, 37]	[26, 28, 37]	[26, 55, 2]	
	[26, 55, 2]	[28, 55, 7]	[28, 26, 55]	[28, 26, 37]	[26, 28, 39]	
	0.772	4.385	2.129	[26, 28, 55]	0.631	0.442
	0.295	1.5	0.815			
61	[28, 1, 30]	[28, 1, 51]	[28, 1, 51]	[28, 51, 30]	[28, 31, 51]	
	[28, 1, 51]	[28, 31, 30]	[28, 32, 30]	[28, 30, 1]	[28, 1, 51]	
	0.454	3.689	1.744	[28, 1, 51]	0.31	0.3
	0.142	1.163	0.6			
62	[38, 1, 50]	[38, 1, 50]	[38, 1, 50]	[38, 1, 47]	[38, 1, 50]	
	[38, 1, 23]	[38, 1, 23]	[38, 1, 50]	[38, 1, 47]	[38, 1, 50]	
	0.097	1.244	0.592	[38, 1, 50]	0.062	0.081
	0.078	0.622	0.347			
63	[35, 26, 40]	[35, 55, 12]	[55, 35, 40]	[26, 55, 35]	[26, 35, 40]	
	[40, 55, 7]	[55, 35, 26]	[40, 55, 26]	[26, 40, 55]	[26, 55, 14]	
	1.224	5.911	2.809	[55, 26, 35]	1.095	0.415

	0.312	1.704	0.766			
64	[39, 4, 59] [39, 32, 4]	[32, 39, 4] [39, 4, 48]	[32, 39, 25] [39, 32, 4]	[39, 4, 48] [39, 32, 4]	[39, 32, 25] [39, 32, 25]	
	0.592	3.556	1.889	[39, 32, 4]	0.361	0.338
	0.267	1.322	0.758			
65	[50, 3, 19] [50, 3, 19]	[50, 3, 13] [50, 3, 13]	[50, 3, 19] [50, 3, 41]	[50, 3, 29] [50, 54, 3]	[50, 41, 13] [50, 3, 41]	
	0.298	2.756	1.344	[50, 3, 41]	0.203	0.192
	0.138	1.074	0.463			
66	[5, 28, 50] [13, 5, 50]	[13, 50, 36] [13, 5, 50]	[13, 5, 50] [13, 5, 50]	[13, 5, 47] [5, 13, 50]	[5, 13, 50] [5, 13, 50]	
	0.666	3.022	1.523	[13, 5, 50]	0.441	0.492
	0.297	1.081	0.497			
67	[19, 30, 23] [30, 19, 56]	[19, 30, 56] [19, 30, 23]	[30, 19, 56] [19, 30, 56]	[19, 30, 23] [30, 19, 23]	[30, 23, 19] [30, 19, 5]	
	0.601	2.8	1.418	[30, 19, 23]	0.491	0.355
	0.367	1.289	0.619			
68	[30, 46, 10] [30, 26, 36]	[30, 36, 29] [30, 26, 36]	[30, 36, 29] [30, 46, 26]	[30, 29, 46] [30, 46, 13]	[30, 46, 36] [30, 46, 26]	
	0.464	3.956	1.918	[30, 46, 36]	0.339	0.23
	0.161	1.33	0.611			
69	[31, 47, 54] [31, 54, 47]	[31, 54, 47] [31, 54, 47]	[31, 54, 47] [31, 54, 47]	[31, 54, 47] [31, 54, 47]	[31, 54, 19] [31, 54, 19]	
	0.131	1.111	0.534	[31, 54, 47]	0.069	0.126
	0.089	0.533	0.319			
70	[35, 19, 53] [35, 38, 30]	[30, 19, 38] [35, 19, 53]	[30, 35, 19] [38, 30, 35]	[35, 30, 38] [35, 30, 53]	[38, 19, 53] [30, 38, 35]	
	1.123	5.674	2.873	[35, 30, 38]	0.822	0.549
	0.274	1.596	0.858			
71	[48, 50, 38] [9, 48, 50]	[48, 9, 50] [48, 50, 38]	[48, 7, 50] [48, 9, 23]	[48, 50, 38] [48, 9, 38]	[9, 48, 38] [48, 9, 23]	
	0.681	4.074	2.133	[48, 9, 50]	0.433	0.298
	0.263	1.378	0.656			
72	[40, 22, 26] [52, 26, 22]	[26, 2, 52] [22, 40, 26]	[40, 26, 22] [40, 4, 52]	[2, 22, 52] [26, 2, 52]	[40, 26, 22] [26, 2, 52]	
	1.253	6.281	3.063	[26, 40, 22]	1.031	0.389
	0.243	1.596	0.776			
73	[56, 31, 30] [31, 56, 30]	[56, 31, 30] [56, 27, 30]	[31, 56, 35] [56, 31, 27]	[31, 56, 16] [31, 56, 16]	[56, 31, 35] [56, 31, 35]	
	0.618	3.333	1.623	[56, 31, 30]	0.45	0.393
	0.346	1.407	0.615			
74	[23, 41, 29] [23, 41, 19]	[23, 41, 29] [23, 41, 19]	[23, 41, 19] [23, 41, 19]	[23, 41, 19] [23, 41, 19]	[23, 41, 19] [23, 41, 19]	
	0.056	0.711	0.37	[23, 41, 19]	0.031	0.066
	0.044	0.356	0.209			
75	[22, 9, 43] [22, 43, 9]	[22, 43, 9] [22, 9, 43]	[22, 9, 43] [22, 43, 9]	[22, 9, 43] [22, 43, 9]	[22, 43, 9] [22, 43, 9]	
	0.2	1.067	0.583	[22, 43, 9]	0.15	0.194
	0.133	0.533	0.246			
76	[56, 47, 38] [28, 26, 45]	[38, 43, 45] [28, 43, 47]	[28, 56, 26] [28, 56, 43]	[56, 28, 26] [28, 43, 6]	[26, 47, 6] [38, 43, 9]	
	1.28	7.527	3.718	[28, 56, 43]	0.931	0.663
	0.187	1.822	0.906			
77	[31, 51, 10]	[31, 51, 10]	[31, 51, 10]	[31, 51, 10]	[31, 51, 10]	

	[31, 51, 10]	[31, 51, 10]	[31, 51, 10]	[31, 51, 10]	[31, 51, 10]	
	0.0	0.0	0.0	[31, 51, 10]	0.0	0.0
	0.0	0.0	0.0			
78	[15, 53, 38] [53, 15, 46]	[9, 53, 38] [15, 53, 37]	[53, 15, 46] [15, 53, 46]	[53, 15, 9] [53, 15, 9]	[15, 53, 37] [15, 53, 38]	
	0.72	3.57	1.742	[53, 15, 9]	0.62	0.489
	0.353	1.444	0.734			
79	[48, 41, 22] [48, 22, 23]	[48, 22, 7] [23, 41, 7]	[48, 41, 23] [48, 22, 41]	[48, 22, 23] [48, 23, 7]	[23, 41, 7] [48, 41, 23]	
	0.818	4.8	2.505	[48, 23, 41]	0.663	0.431
	0.175	1.352	0.705			
80	[48, 38, 31] [48, 38, 31]	[48, 38, 31] [48, 38, 31]	[48, 38, 31] [48, 31, 38]	[48, 38, 31] [48, 38, 4]	[48, 38, 31] [48, 38, 31]	
	0.107	0.8	0.442	[48, 38, 31]	0.053	0.123
	0.072	0.389	0.213			
81	[38, 27, 3] [38, 27, 26]	[38, 27, 26] [38, 3, 44]	[38, 3, 44] [38, 3, 44]	[38, 27, 26] [38, 3, 44]	[38, 27, 3] [38, 27, 3]	
	0.349	3.067	1.574	[38, 27, 3]	0.266	0.25
	0.122	1.044	0.556			
82	[9, 54, 37] [6, 54, 29]	[6, 9, 29] [9, 6, 54]	[9, 54, 6] [9, 56, 39]	[9, 6, 54] [9, 6, 54]	[9, 6, 56] [9, 56, 36]	
	0.753	4.782	2.455	[9, 6, 54]	0.471	0.467
	0.233	1.508	0.769			
83	[2, 4, 14] [2, 23, 52]	[2, 14, 4] [2, 23, 52]	[2, 4, 14] [14, 23, 52]	[2, 23, 52] [2, 23, 52]	[2, 4, 14] [2, 23, 52]	
	0.625	4.207	2.158	[2, 23, 14]	0.452	0.419
	0.101	1.096	0.503			
84	[44, 59, 7] [59, 44, 7]	[59, 44, 7] [59, 44, 7]	[44, 59, 7] [44, 7, 37]	[44, 59, 52] [59, 44, 7]	[44, 59, 7] [59, 44, 7]	
	0.572	2.311	0.952	[44, 59, 7]	0.445	0.358
	0.319	0.981	0.59			
85	[30, 50, 46] [30, 50, 46]	[30, 50, 1] [30, 50, 29]	[30, 50, 1] [30, 50, 1]	[30, 50, 16] [30, 50, 16]	[30, 50, 16] [30, 50, 1]	
	0.122	1.556	0.688	[30, 50, 1]	0.094	0.081
	0.097	0.778	0.421			
86	[22, 9, 21] [22, 45, 9]	[22, 45, 9] [22, 21, 45]	[22, 21, 9] [22, 9, 21]	[22, 9, 45] [22, 9, 21]	[22, 9, 45] [22, 21, 9]	
	0.368	2.667	1.403	[22, 9, 21]	0.27	0.235
	0.178	1.033	0.55			
87	[22, 9, 20] [22, 9, 19]	[22, 9, 19] [22, 9, 20]	[22, 9, 20] [22, 9, 20]	[22, 9, 20] [22, 9, 20]	[22, 9, 20] [22, 9, 20]	
	0.056	0.711	0.337	[22, 9, 20]	0.031	0.066
	0.044	0.356	0.196			
88	[26, 51, 43] [26, 60, 1]	[26, 43, 1] [26, 60, 46]	[26, 60, 51] [26, 60, 1]	[26, 43, 1] [26, 46, 1]	[26, 46, 43] [26, 51, 60]	
	0.496	4.178	2.161	[26, 60, 43]	0.404	0.215
	0.146	1.304	0.658			
89	[1, 30, 56] [1, 30, 54]	[1, 56, 30] [56, 1, 54]	[1, 60, 54] [1, 60, 3]	[1, 30, 60] [1, 30, 54]	[1, 56, 30] [1, 30, 3]	
	0.592	4.089	2.036	[1, 30, 56]	0.403	0.4
	0.208	1.367	0.795			
90	[17, 55, 14] [17, 55, 24]	[55, 17, 14] [17, 55, 14]	[55, 17, 14] [55, 17, 14]	[55, 17, 14] [55, 17, 14]	[17, 55, 14] [17, 55, 14]	
	0.448	1.511	0.781	[55, 17, 14]	0.391	0.414

	0.303	0.756	0.433			
91	[45, 40, 6] [51, 40, 6]	[45, 40, 6] [45, 51, 44]	[51, 45, 40] [45, 51, 44]	[51, 40, 6] [45, 51, 44]	[51, 45, 44] [45, 51, 44]	
	0.809	4.519	2.176	[45, 51, 40]	0.625	0.506
	0.235	1.348	0.689			
92	[9, 26, 56] [9, 26, 56]	[26, 9, 10] [26, 22, 10]	[26, 9, 22] [9, 26, 56]	[26, 9, 10] [9, 22, 56]	[9, 26, 22] [9, 26, 56]	
	0.685	3.674	1.975	[9, 26, 22]	0.517	0.471
	0.312	1.37	0.669			
93	[5, 38, 47] [5, 6, 16]	[5, 38, 47] [5, 6, 47]	[5, 38, 47] [5, 38, 47]	[5, 6, 16] [5, 47, 27]	[5, 6, 16] [5, 6, 38]	
	0.413	3.6	1.892	[5, 6, 38]	0.346	0.239
	0.121	1.137	0.601			
94	[6, 52, 30] [6, 52, 23]	[6, 52, 23] [6, 52, 23]	[6, 52, 23] [6, 52, 23]	[6, 52, 23] [6, 52, 23]	[6, 52, 23] [6, 52, 23]	
	0.031	0.4	0.216	[6, 52, 23]	0.016	0.049
	0.025	0.2	0.083			
95	[48, 25, 38] [48, 25, 38]	[48, 25, 38] [48, 12, 38]	[48, 12, 38] [48, 38, 12]	[48, 12, 25] [48, 12, 38]	[48, 25, 38] [48, 12, 25]	
	0.366	2.711	1.395	[48, 12, 25]	0.32	0.246
	0.144	0.933	0.428			
96	[60, 1, 26] [60, 1, 26]	[60, 1, 26] [60, 26, 1]	[60, 1, 26] [60, 1, 26]	[60, 1, 26] [60, 1, 26]	[60, 1, 54] [60, 1, 26]	
	0.107	0.8	0.441	[60, 1, 26]	0.053	0.123
	0.072	0.389	0.188			
97	[42, 8, 55] [8, 60, 31]	[42, 8, 60] [8, 42, 31]	[8, 42, 60] [8, 42, 31]	[8, 42, 60] [42, 8, 60]	[8, 42, 22] [8, 42, 60]	
	0.555	2.919	1.445	[8, 42, 60]	0.342	0.357
	0.323	1.27	0.682			
98	[9, 40, 7] [9, 40, 7]	[9, 40, 6] [9, 40, 7]	[9, 40, 55] [9, 40, 46]	[9, 40, 36] [9, 40, 49]	[9, 40, 36] [9, 40, 36]	
	0.135	1.733	0.844	[9, 40, 7]	0.109	0.075
	0.108	0.867	0.411			
99	[26, 9, 43] [26, 9, 28]	[26, 9, 28] [26, 9, 28]	[26, 9, 28] [26, 9, 43]	[26, 9, 43] [26, 9, 46]	[26, 9, 28] [26, 9, 43]	
	0.101	1.289	0.632	[26, 9, 28]	0.078	0.082
	0.081	0.644	0.276			

all1pws.ltgt		
	Near	77=0.0, 94=0.2, 15=0.2, 87=0.356, 74=0.356, 26=0.356, ...
	Far	... 72=1.596, 46=1.63, 23=1.685, 63=1.704, 41=1.794, 76=1.822,
all1pws.ssfr		
	Near	77=0.0, 94=0.4, 15=0.4, 87=0.711, 74=0.711, 26=0.711, ...
	Far	... 70=5.674, 46=5.852, 63=5.911, 72=6.281, 41=7.107, 76=7.527,
randpws.ltgt		
	Near	77=0.0, 94=0.083, 15=0.128, 26=0.182, 96=0.188, 87=0.196, ...
	Far	... 89=0.795, 60=0.815, 70=0.858, 23=0.887, 76=0.906, 41=0.907,
randpws.ssfr		
	Near	77=0.0, 15=0.206, 94=0.216, 87=0.337, 74=0.37, 26=0.427, ...
	Far	... 50=2.951, 23=2.965, 72=3.063, 46=3.075, 41=3.558, 76=3.718,
tailharmpws.ltgt		
	Near	77=0.0, 94=0.025, 15=0.025, 87=0.044, 74=0.044, 96=0.072, ...

	Far	... 31=0.328, 73=0.346, 78=0.353, 54=0.361, 24=0.364, 67=0.367,
tailharmpws.ssfr		
	Near	77=0.0, 94=0.031, 15=0.031, 87=0.056, 74=0.056, 62=0.097, ...
	Far	... 70=1.123, 46=1.189, 63=1.224, 72=1.253, 76=1.28, 41=1.343,

B Permutation Rankings

All possible 3-rank vs 3-rank permutations, ordered by each of the six distance metrics. The ”x” code indicates the permutation. For example, if [a,b,c] goes to [a,b,c] that is ”x123”, whereas if [a,b,c] goes to [c,b,a] that is ”x321”. (Note that the ”b” is in the same place in both of these.) A dash (-) indicates that an element does not appear in both 3-ranks. For example, [a,b,c] [a,b,d] is ”x12-”, and [a,b,c] goes to [d,e,f] is ”x—”.

Note that some of the metrics are ”fine grained”, others have many equi-score sets. Also, as expected, all the distance metric agree that x123 represents the best possible agreement. However, they disagree about which are the worst. Although under one intuition, x— should be the worst possible disagreement, under another interpretation, putting one ranking's top score at the bottom (or v.v.) may be considered worse than a disagreement about the elements. These points are discussed in more detail in the text.

tailharmpws.ssfr		**all1pws.ssfr**		**randpws.ssfr**		**tailharmpws.ltgt**		**all1pws.ltgt**		**randpws.ltgt**	
x123	0.00	x123	0.00	x123	0.00	x123	0.00	x123	0.00	x123	0.00
x12-	0.16	x132	2.00	x-23	0.36	x–3	0.11	x132	1.00	x12-	0.13
x132	0.38	x213	2.00	x231	0.58	x1–	0.11	x213	1.00	x-23	0.16
x13-	0.55	x12-	2.00	x132	0.74	x-2-	0.11	x-23	1.00	x2-3	0.21
x1-2	0.55	x21-	4.00	x213	0.83	x—	0.12	x1-3	1.00	x213	0.23
x1-3	0.56	x13-	4.00	x12-	1.18	x-23	0.13	x12-	1.00	x321	0.28
x1–	0.65	x231	4.00	x1-2	1.53	x12-	0.13	x321	1.00	x3-2	0.31
x213	0.75	x1-2	4.00	x1-3	1.62	x1-3	0.13	x2-3	1.50	x–3	0.38
x21-	0.91	x1-3	4.00	x32-	1.70	x3–	0.20	x32-	1.50	x–1	0.40
x231	1.19	x321	4.00	x23-	2.41	x-3-	0.20	x-21	1.50	x132	0.43
x312	1.19	x312	4.00	x21-	2.64	x–2	0.20	x-13	1.50	x–2	0.46
x321	1.25	x2-3	6.00	x-21	2.68	x–1	0.20	x13-	1.50	x13-	0.47
x2–	1.28	x-23	6.00	x-2-	2.73	x132	0.25	x1-2	1.50	x3–	0.55
x-1-	1.28	x32-	6.00	x13-	2.79	x32-	0.25	x312	1.50	x231	0.61
x2-1	1.36	x-21	6.00	x2-3	2.94	x-21	0.25	x231	1.50	x2-1	0.64
x31-	1.36	x-13	6.00	x312	3.05	x13-	0.25	x–3	1.67	x32-	0.64
x-2-	1.42	x2-1	6.00	x-12	3.05	x1-2	0.25	x1–	1.67	x1-3	0.66
x-12	1.44	x1–	6.00	x31-	3.07	x321	0.25	x-2-	1.67	x-31	0.69
x23-	1.44	x23-	6.00	x-13	3.25	x2–	0.28	x2–	2.00	x-21	0.70
x2-3	1.45	x-12	6.00	x2-1	3.28	x-1-	0.28	x3-1	2.00	x1–	0.79
x-13	1.45	x31-	6.00	x321	3.30	x2-3	0.38	x-32	2.00	x-1-	0.83
x32-	1.50	x2–	7.33	x–3	3.48	x3-1	0.38	x3–	2.00	x1-2	0.91
x-21	1.50	x-2-	7.33	x3-1	3.57	x-32	0.38	x-3-	2.00	x-3-	0.94
x-23	1.59	x-1-	7.33	x3–	3.64	x-13	0.38	x21-	2.00	x-13	1.00
x3–	1.60	x3-1	8.00	x-3-	3.70	x-31	0.38	x2-1	2.00	x3-1	1.00
x–1	1.60	x-32	8.00	x-31	3.96	x3-2	0.38	x23-	2.00	x23-	1.01
x-3-	1.70	x3-2	8.00	x2–	3.98	x213	0.50	x-31	2.00	x-2-	1.05
x–2	1.70	x-31	8.00	x-1-	4.61	x2-1	0.50	x-12	2.00	x—	1.12
x–3	1.73	x–3	8.67	x3-2	4.82	x231	0.50	x–2	2.00	x2–	1.15
x3-1	1.81	x3–	8.67	x–1	4.91	x312	0.50	x-1-	2.00	x-32	1.26
x—	1.84	x-3-	8.67	x–2	5.08	x31-	0.50	x3-2	2.00	x312	1.33
x-31	1.89	x–2	8.67	x1–	5.10	x-12	0.50	x–1	2.00	x31-	1.35
x3-2	1.89	x–1	8.67	x—	5.77	x23-	0.50	x31-	2.00	x21-	1.39
x-32	1.97	x—	11.27	x-32	6.29	x21-	0.63	x—	2.18	x-12	1.65

C A Bayesian Approach (Jameson's Comments)

In the Bayesian way of thinking, models, priors, and likelihoods are the core entities upon which all else rests. In this way of thinking, distance metric is epi-phenomenal, or, more generously, a computed measure which must be computed from the core entities. One way to do this might be to compute the distance measure from the likelihood. More precisely: We design (or assume) a model that is comprised of a generating function for judgments (i.e., n-ranks for particular cases), and appropriate hyper-parameters. This model can generate the 100x10 example specialistment sets, and the hyper-parameters would start from some priors, and can learn from real examples to generate n-ranks like the cases+specialistment sets that is has learned from. For each new case, the distance could be thought of as, for example, the log, of the difference between the priors (hyper-parameter values) of the model before seeing a particular case (i.e., $log(|NewParamVals - OldParamVals|)$), and where you end up after updating the parameters in accord with that new case. If the hyper-parameters don't have to move very far in order to encompass the new case-x-ranking(s) then the ranking that would be created by the current parameters are very close. (Jeff want's to use something like $tanh(1/|NewParamVals - OldParamVals|)$. Regardless, neither this, nor the log will have a true zero, and so isn't a true distance metric, but it can come arbitrarily close to zero.) One way to think of this is as maximizing the joint likelihood of the latent variables in such a way that they are a compromise between latent values that best explain each of the rankings that are being compared. This metric will correlate with the desired distance, insofar as that compromise latent value is very far away from the priors, which is what you want.

The advantage of the Bayesian approach is that there are fewer parameters than our current algorithmic approach, and thus fewer things to get wrong. Of course, the disadvantage is that there are fewer parameters than our current algorithmic approach, and thus fewer things to can get right. For instance if you know, for instance, that specialists won't indicate the same treatment twice, or won't indicate a similar treatment to a one they've already indicated (as in the categorical strategy), you can put that into the likelihood and and it naturally folds into the computation, whereas trying to tweak the foot rule could be arbitrarily complex, and can create unintended consequences that ruin the consistency of the metric.