Prevalence of Simpson's Paradox in Nonparametric Statistical Analysis of Medical and Other Scientific Data: Theoretical and Computational Analysis

James Boudreau[1]
Justin Ehrlich[2]
Shane Sanders[3]

September 9, 2020

I. Introduction

Simpson's Aggregation Paradox, also known as the *Yule-Simpson Aggregation Paradox*, represents an anomaly in statistics whereby two qualitatively equivalent statistical test results—each arising from one of two distinct constitutent data sets—disappears when the same statistical test is applied to the pooled data. The paradox was first put forth by Yule (1903) and later developed by Simspon (1951). While first considered strictly for the domain of parametric testing, its presence in non-parametric statistical results has recently been studied (Haunsperger, 2003; Haunsperger and Saari ,1991; Bargagliotti, 2009). Of particular importance to the present study, Haunsperger and Saari (1991) find conditions for *Simpson reversals* in rank sum statistical testing, where the term *Simpson reversal* is used synonymously with the term *instances of Simpson's Aggregation Paradox* herein. In general, the paradox has been found to affect statistical results in many important scientific domains, including environmental and related ecological research (see, e.g., Pineiro et al., 2006 or Allison and Goldberg, 2002). In studying global temperature over time, Foster and Rahmstorf (2011) note that the scale of data (time scale of study) can influence the statistical results of a study, for example.

In one respect, the Simpson paradox can be viewed as a robustness check on a given statistical result. When the paradox occurs, it follows that a given result is at least partly a function of scale or sample size. As noted, the paradox has been shown to occur for the *Wilcoxon-Mann-Whitney* (*WMW*) *Rank Sum Test*. However, there exists no computational or empirical evidence as to the frequency with which instances of the paradox occur in the case of the *WMW Test* and little such evidence for non-parametric statistical tests overall. Are *Simpson reversals* pervasive or only a marginal concern for the *WMW Test*? Even previous research as to the incidence of the paradox for parametric statistical tests is scarce and provides somewhat contrasting conclusions.

There are two studies that directly estimate the incidence of *Simpson's Paradox* for parametric tests: one pertaining to contingency tables and the other pertaining to path models. Specifically, Pavlides and Perlman (2009) find that a Simpson Reversal occurs for one-sixtieth (1.67%) of all $2x2x2$ contingency tables. Kock (2015) estimates the likelihood of a *Simpson reversal* in path models as approximately 12.8%. For nonparametric statistical tesitng, Nagaraja and Sanders (2020) consider a case in which a data set is ordinally replicated and then pooled with the replicate data set. In such an environment, the authors prove that *Simpson reversals* cannot

[1]Economics, Finance, and Qualitative Analysis, Kennesaw State University, email: jboudre5@kennesaw.edu.
[2]Sport Analytics, Syracuse University, Syracuse, email: jaehrlic@syr.edu.
[3] Sport Analytics, Syracuse University, Syracuse, email: sdsander@syr.edu.

occur if the sign test for matched pairs is applied to the primitive and pooled data sets. They also show evidence of *Simpson reversals* for the *WMW Test*.

Despite the important theoretical contributions by Haunsperger and Saari (1991) and Nagaraja and Sanders (2020), there have been no computational studies that assess the incidence of *Simpson reversals* in the case of non-parametric tests. Though we know the *WMW Test* yields instances of the paradox, we cannot ascertain without computational support whether these instances are more frequent, as in the case of path models, or somewhat rare, as in the case of contingency tables.

The answer to the previous question has potentially important implications. The *WMW Test* is a leading non-parametric statistical test across the medical sciences. For example, this test is routinely used to assess drug efficacy in *FDA* clinical trials (see, e.g., Boudreau et al. 2018 for a discussion of *FDA* use of this test in the Statistical Review and Evaluations of products such as Novantrone, Cologuard, Memantine, Pitressin, Berinert, SPD485, Oxcarbazepine, Oxaliplatin, Novartis, Trileptal, Vascepa, and countless others). Results as to the incidence of *Simpson reversals* for the *WMW Test* can effectively assess the general robustness of *WMW Test* results to data scale changes.

As *Simpson reversals* cast ambiguity upon a given original result, the incidence of *Simpson reversal* for a test shares similarities with the concept of a hypothesis test *p-value*. In the same way that a *p-value* assesses the proportion of significance results that are, in fact, non-robust due to sample variation, incidence of *Simpson reversal* assesses the proportion of statistical test results that are non-robust due to data scale dependence. In this sense, the proportional incidence of *Simpson reversals* might be thought of as loosely analogous to a hypothesis test *p-value* (e.g., when considering the magnitude of the proportion).

Herein, we report the initial development of methods to consider all 2-group, n-element per group cases of rank sum scoring for $n \in \{2,3,4,5,6,7,8\}$. For each case up to $n = 7$, we enumerate every possible rank outcome sequence in that case. For each given sequence, we then ordinally replicate the sequence and consider all possible poolings of the sequence with its ordinal replicate. For each case, we then compute the relative frequencey with which a strict *Simpson reversal* occurs. We find that strict instances of the *Paradox* cannot occur for 2-group, k-element per group cases of rank sum scoring where $n \in \{1,2\}$ but that instances theoretically occur for at least as many as roughly 1.7 percent of sequence poolings in the 2-group, 5-element and 2-group, 7-element cases. Given the computational complexity of the problem—for the 2-group, 8-element case, there are 7.74 trillion possible poolings of two rank data sequences—we are not able to extend these theoretical lower-bound (sufficiency) results beyond the $n = 7$ case at present. However, we use our theoretical sufficiency condition (Theorem 1) to guide a simulation approach to characterize the 2-group, 8-element case, as well as nuances of cases with $n \leq 7$.

We conclude from our computational results that the incidence of *Simpson reversal* for small sample cases of rank sum scoring is (not) roughly similar to previous results on *2x2x2* contingency tables (path models). Moreover, the computed rate of *Simpson reversals* in this setting is lower than a standard, allowable Type I error rate (α-value) for a statistical test. Given the conceptual similarities between a test's *p-value* and its *Simpson reversal* rate as discussed previously, we might then characterize the incidence of *Simpson reversals* for considered cases of rank sum testing as "tolerable" from the perspective of statistical sensitivity. For certain initial data sequences, however, *reversals* are found to be much more prevalent, occurring as frequently as roughly once in five poolings for certain sample size cases. As such, the incidence of *Simpson reversals* should ideally be considered conditional upon both the test and data under consideration.

II. Rank Sum Scoring and Simpson's Aggregation Paradox: Definitions and a Theorem

Definitions

Let us formally define two-group rank sum scoring. Consider two groups, *A* and *B*. Each group is defined as a rank-ordered sequence of n individual elements, where n is some integer greater than 1 ($n \in Z^{+}$). For example, A is defined as $A = (a_1, a_2, a_3, \dots, a_n)$, where the element a_i represents the i^{th} ranked element in A. We define an event as an objective process of comparison that generates a complete rank-order sequence of individuals across more than one group (i.e., both within and between groups). An event might be defined as a competition or as a statstical test. Consider an event in which elements of *A* and *B* are compared. If *A* and *B* are each composed of n elements, for example, then the event generates a rank-ordered outcome sequence of $2n$ elements. One possible outcome sequence for the case in which $n = 3$ is $F_{AB} = (a_1, b_1, b_2, a_2, b_3, a_3)$. If a_i precedes b_j in the outcome sequence, we say $a_i \succ b_j$ (a_i ranks higher than b_j). For simplicity, we assume that rank-order equality between two elements is not possible, an outcome that would obtain given continuous measurement of underlying parameter values. For any $a_i \in A$ and $b_j \in B$, that is, we have that $a_i \succ b_j \oplus b_j \succ a_i$ is a tautology.

Formally, we represent the rank of an element $a_i \in A$ in the outcome sequence F_{AB} as $r(a_i|F_{AB})$. Let $x_i^{+}(F_{AB}) = \{x \in F_{AB}: x \succ a_i\}$ be the set of elements in F_{AB} that rank better than a_i. Then, $r(a_i|F_{AB}) = |x_i^{+}(F_{AB}) + 1|$. From elemental rankings, we generate a rank sum score for each group as follows. The respective scores for A and B based for the outcome sequence F_{AB} are $S(A|F_{AB}) = \sum_{a_j \in A} r(a_j|F_{AB})$ and $S(B|F_{AB}) = \sum_{b_j \in B} r(b_j|F_{AB})$, where it must be that $S(A|F_{AB}) + S(B|F_{AB}) = \frac{2n(2n+1)}{2}$. That is, the sum of ranks for a $2n$ element sequence simply equals the sum of integers from 1 to $2n$. We map from group scores to group rankings to obtain the following outcomes.

If $S(A|F_{AB}) < S(B|F_{AB})$, then $A \succ B \equiv$ If $S(A|F_{AB}) < S(B|F_{AB})$, then *A ranks higher than B* (1)
If $S(A|F_{AB}) = S(B|F_{AB})$, then $A \sim B \equiv$ If $S(A|F_{AB}) = S(B|F_{AB})$, then *A ranks equally with B* (2)
If $S(A|F_{AB}) > S(B|F_{AB})$, then $A \prec B \equiv$ If $S(A|F_{AB}) < S(B|F_{AB})$, then *A ranks lower than B* (3)

Replicated Data Aggregation

We consider an environment in which a data set yields a given aggregate or group rank-ordering result under rank sum scoring (e.g., $A \succ B$). We then ordinally replicate the data. By necessity, the ordinal replicate data will yield the same group rank result under rank sum scoring. As rank sum scoring is a non-parametric form of scoring, only the *order* of elements influences the group ranking. We then aggregate the original data set with its ordinal replicate as in Nagaraja and Sanders (2020) and consider whether (under what conditions) the pooled data yields a different group rank result under rank sum scoring than do its two constituent data sets. That is, we consider the conditions for strict *Simpson reversal*, whereby outcome 1 (3) is obtained for each constitutent data sequences but outcome 3 (1) is obtained for the pooled sequence. It is important to note that an ordinal-replicate data sequence can have starkly different *parametric* values than the original data sequence that it ordinally replicates. Ordinal replication simply implies the same *ordering* of elements across the two sequences.

Let F_{AB} represent the original data sequence, F'_{AB} its ordinal replicate, and FF'_{AB} the sequence whereby F_{AB} and F'_{AB} are pooled by comparing the underlying parametric value of each element. Formally, we define a *Simpson reversal* as follows.

Definition: *Simpson Reversal*: A strict *Simpson reversal* occurs if $[S(A|F_{AB}) - S(B|F_{AB})]\cdot[S(A|FF'_{AB}) - S(B|FF'_{AB})] < 0$. Equivalently, a strict *Simspon reversal* occurs if $[S(A|F'_{AB}) - S(B|F'_{AB})]\ \cdot [S(A|FF'_{AB}) - S(B|FF'_{AB})] < 0$. These conditions yield the group rank result that $(A \succ_F B\ \wedge A \succ_{F'} B)$ but $B \succ_{FF'} A$ (i.e., that A ranks strictly higher than B in F and F', but B ranks strictly higher than A for FF').

Theorem

We now derive a sufficient condition for the occurrence and absence of *Simpson reversal* in Rank Sum Scoring.

<u>**Theorem 1:**</u> **Sufficient Condition for *Simpson Reversal* in Rank Sum Scoring:** For any two groups, A and B, such that $A \succ B$ in pairwise comparison for a given outcome sequence, F_{AB} (i.e., $A \succ_{F_{AB}} B$), let ζ be the largest integer such that $b_{i+\zeta-1} \succ a_i$ in F (F'). A strict *Simpson reversal* occurs if $S(B|F_{AB}) - S(A|F_{AB}) < n\zeta$.

Proof: Consider the differential impact toward a reversal that F'_{AB} can have when pooled with F_{AB} if all n elements of F'_{AB} are pooled with F_{AB} such that they are placed between $b_{i+\zeta-1}$ and a_i of F_{AB}. In this case, the pooling effect of F'_{AB} upon F_{AB} is to raise the score of A by $2n\zeta$ more rank sum units than the score of B. If the $2n$ elements of F'_{AB} are pooled with F_{AB} at this position, then ζ more elements of A in F_{AB} than B in F_{AB} lose $2n$ rank positions (gain $2n$ additional rank sum points) to the elements of F'_{AB}.

As a countervailing effect, A has a lower score than B by $S(B|F_{AB}) - S(A|F_{AB})$ units in F_{AB} (by definition) and by $S(B|F_{AB}) - S(A|F_{AB})$ units in F'_{AB}, as $S(B|F'_{AB}) - S(A|F'_{AB}) = S(B|F_{AB}) - S(A|F_{AB})$ due to F_{AB} and F'_{AB} being ordinal replicates. Then, $S(A|FF'_{AB})$ relative to $S(B|FF'_{AB})$ depends upon the magnitude of the pooling effect in comparison to the magnitude of $[S(B|F_{AB}) - S(A|F_{AB})]$ and $[S(B|F'_{AB}) - S(A|F'_{AB})]$, where the latter two terms are equal to each other. For a sequence, F_{AB}, and its ordinal replicate, then, a *Simpson reversal* is certain to occur if $2\cdot[S(B|F_{AB}) - S(A|F_{AB})] < 2n\zeta$ or if $S(B|F_{AB}) - S(A|F_{AB}) < n\zeta$ ∎

Interestingly, this condition is equivalent to the condition for a violation of *Independence from Irrelevant Alternatives* (IIA) found in Boudreau et al. (2014). This equivalence is not coincidental. Rather, *Simpson reversals* share important properties with IIA violations. In each case, a pairwise group ranking is overturned by the inclusion of additional data, where the imposed data is not expected to overturn the original ranking. Like an *IIA violation*, a *Simpson reversal* requires the additional data to impose a sufficiently differential effect upon the respective rank sum scores of the two groups being compared. The conditions for that differential effect are the same for *IIA* violations and for *Simpson reversals*.

III. The Sample Space: A Combinatorial Description

For the $2 \; x \; n$ case, there are $\frac{(2n)!}{(n!)^2}$ initial sequences, F. We are arranging $2n$ elements—n elements from each of 2 groups—where we do not distinguish between respective objects of a given group. For each initial sequence, we then ask in how many ways F can be pooled with its ordinal replicate, F'. This is equivalent to a "stars and bars" combinatorial problem, in which we are placing $2n$ "stars" or elements from F' into $2n$ "bars" or potential pooling positions between the elements of F. Therefore, there are $\frac{(4n)!}{([2n]!)^2}$ poolings for each initial sequence. As such, the number of poolings for a given $(2 \; x \; n)$ case equals the product of the number of initial sequences and the number of poolings per initial sequence. Hence, the total number of poolings equals $\frac{(2n)!}{(n!)^2} \cdot \frac{(4n)!}{([2n]!)^2}$ for each case, $(2 \; x \; n)$. For example, for the $2 \; x \; 7$ case, there are $\frac{(2 \cdot 7)!}{(7!)^2} = 3{,}432$ initial sequences, F. Moreover, there are $\frac{(4 \cdot 7)!}{([2 \cdot 7]!)^2} = 40{,}116{,}600$ poolings per initial sequence. As such, there are approximately $3{,}432 \cdot 40{,}116{,}600 \approx 137.68$ billion possible poolings for the $2 \; x \; 7$ case. We provide the sample space for each $(2 \; x \; n)$ case through $n = 7$, in Table 1 below.

IV. Computational Results and Discussion

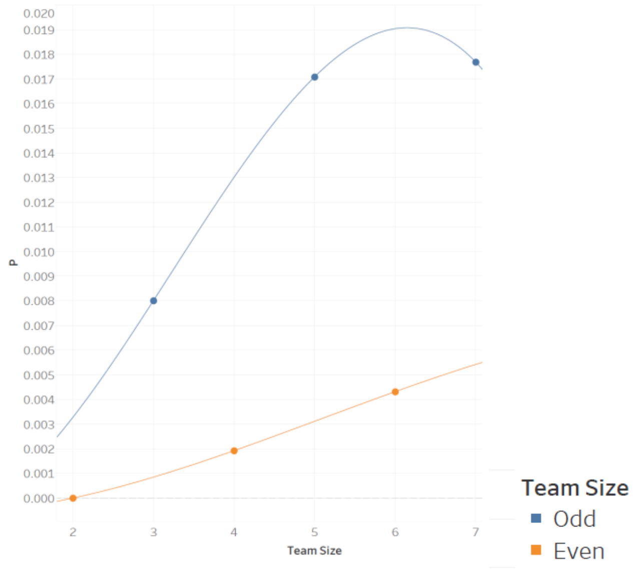
We wrote a computational algorithm in Java by which to search the sample space of each case where $0 < n(\in Z^+) < 7$. The algorithm programmatically generates all possible initial sequences, F_{AB} (F'_{AB}), for a case, then creates all possible pooled sequences, FF'_{AB}, for each initial sequence. For each initial sequence, rank sum scores for A and B are computed. This scoring task is then repeated for each pooling FF'_{AB} of F_{AB} and F'_{AB} and iteratively for each pooling of each initial sequences. Then, instances of $Simpson\,reversal$ are checked using the condition obtained in Theorem 1. This brute force, enumerative approach is extended later in the paper using a simulation approach. The algorithmic code is provided in Appendix 1 of the paper. Computational results are given in Table 1 as follows.

Table 1: Relative Frequency of Simpson Reversal by Case

|Groups|	|Data Points per Group|	|Initial Data Sequences|	|Poolings per Initial Sequence|	|Poolings overall|	Simpson Reversal Rel. Frequency
2	1	2	6	12	0/12 = **0%**
2	2	6	70	420	0/420 = **0%**
2	3	20	924	18,480	30/18,480 = **0.80%**
2	4	70	12,870	900,900	1,732/900,900 = **0.19%**
2	5	252	184,756	46,558,512	795,392/46,558,512 = **1.71%**
2	6	924	2,704,156	2,498,640,144	10,780,504/2,498,640,144 = **0.43%**
2	7	3,432	40,116,600	137,680,171,200	2,435,044,740/ 137,680,171,200 = **1.77%**

We observe that *Simpson reversals* are not possible for sufficiently small n (i.e., $n < 3$). In the context of Theorem 1, the largest possible ζ is not sufficiently large to motivate a strict *Simpson reversal*. For the $2x1$ and $2x2$ cases, a group that is strictly outranked in F_{AB} cannot have a positive ζ and therefore a strict *Simpson reversal* is not possible for these cases. We can also consider computed cases where $n > 2$. From even to odd cases, the results suggest a wavelike movement in the likelihood of a *Simpson reversal*. In general, there is a lower likelihood of strict *Simpson reversal* in even cases due to the possibility of ties for *n-even* cases of pairwise rank sum scoring (but not for *n-odd* cases). With some probability mass allowing for a pairwise tie in the *n-even* cases, strict *Simpson reversals* are less likely. This result also holds for other social choice violations (e.g., violations of *transitivity* and of *IIA;* see Boudreau *et al. 2014*). To evaluate the marginal effect of increases in n, as distinct from the effect of changes from even to odd case, one should compare the iterative trend between n and $n + 2$ rather than that between n and $n + 1$. We do this for the even and odd cases respectively in Figure 1.

Figure 1:



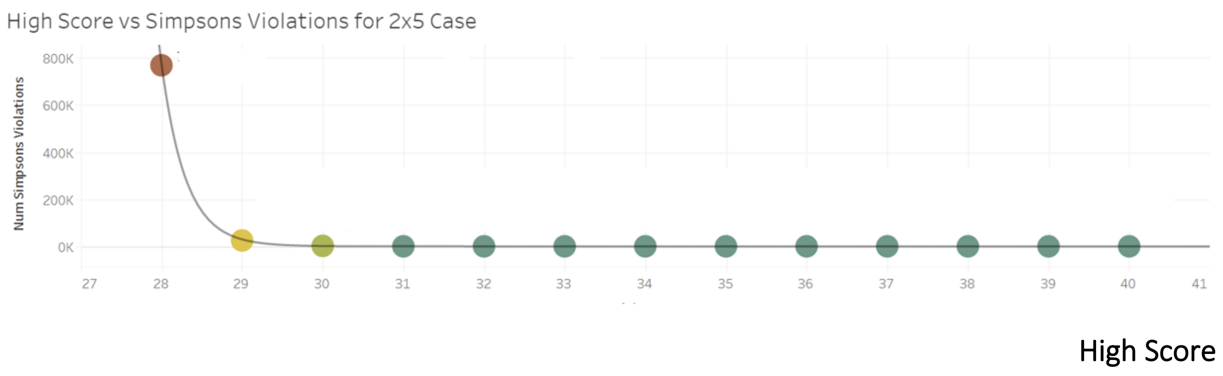
Over the set of cases computed, the relative frequency of *reversal* rises for both the even and odd set of cases. For the *2x8* case, we run a simulation to estimate whether this trend might continue.

Specifically, we randomly select and generate one-quarter of all possible initial sequences, F_{AB}, (without replacement) for this case and then replicate each selected initial sequence. For each selected initial sequence and its replicate, we then randomly select approximately 0.1% of all possible poolings or a little more than 600,000 poolings per sampled initial sequence. For each pooling, we check for *reversals* as in the main algorithm. Doing so, we estimate that 0.63% of all poolings result in reversal for the *2x8* case. In proportion terms, this represents a substantial increase from the *2x6* case. As such, this estimate suggests that our trend of rising relative frequency of reversal from n to $n + 2$ is maintained for the *2x8* case.

We find that strict instances of the *Paradox* cannot occur for 2-group, k-element per group cases of rank sum scoring where $k \in \{1,2\}$ but that instances occur for as many as roughly 1.7 percent of sequence poolings in the 2-group, 5-element and 2-group, 7-element cases. We conclude from our computational results that the incidence of *Simpson reversal* for small sample cases of rank sum scoring is (not) roughly similar to previous results on *2x2x2* contingency tables (path models). Moreover, the computed rate of *Simpson reversals* in this setting is generally lower than a standard, allowable Type I error rate (α-value) for a statistical test. Given conceptual similarities between a test's *p-value* and its *Simpson reversal* rate, as discussed previously, we might then characterize the incidence of *Simpson reversals* for considered cases of rank sum testing as "tolerable" from the perspective of statistical sensitivity.

Next, we consider how likelihood of *Simpson reversal* relates to rank sum score for A and B in F_{AB}. We do this sub-analysis for the *2x5* case and visualize the results in the heat map and scatter plot of Figure 2.

Figure 2: Heat Map and Scatter Plot Relating Rank Sum Scores to Likelihood of Reversal



For the *2x5* case, *reversals* are most likely when the rank sum score margin in F_{AB} is closest (i.e., where one group scores 27 and the other scores 28). A *reversal* is more likely if the original score margin is close due to the relative ease with which a *reversal* can be obtained in such a case. As the score margin increases, the relative frequency of *reversals* declines quickly. This observed relationship between match "closeness" and likelihood of violation mirrors earlier results for violations of *transitivity* and *IIA* under rank sum scoring (see Boudreau et al. 2019). We also find that *reversals* cannot occur if the rank sum score margin in F_{AB} is equal to 7 or more for the *2x5* case. If the score margin is 7 or more, then it must be that $\zeta \leq 1$. As such, we know that

$S(B|F_{AB}) - S(A|F_{AB}) > n\zeta$ for this range of score margins in the *2x5* case, and a *reversal* cannot occur.

While the overall likelihood of *reversal* is relatively low for small sample cases of rank sum scoring (e.g., relative to a standard α-value), there is evidence that certain types of sequences are problematic. For example, sequences that yield closer scores were shown to be more productive of *reversals*. As such, we compute the relative frequency of *reversal* for each initial sequence in each case and then identify the initial sequence for each case that yields the highest such relative frequency, as well as the relative frequency itself. In Figure 3, we plot the highest relative frequency of *reversal* at the initial sequence level for each computed case. These same results are represented in greater detail within Table 2.

Figure 3: Highest Initial Sequence Level Reversal Likelihood by Case

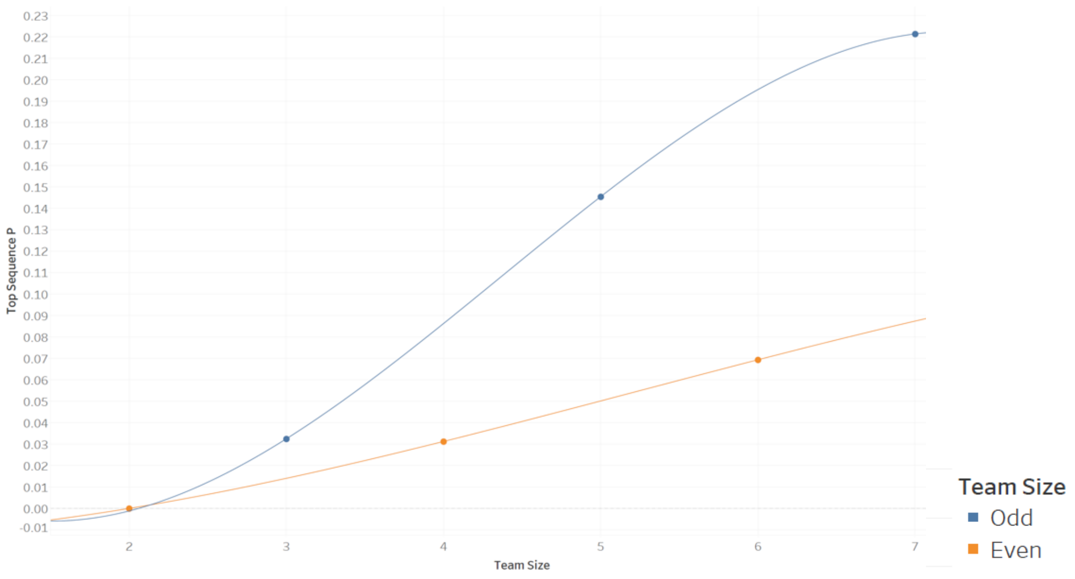


Table 2: Highest Initial Sequence Level Reversal Likelihood by Case

|Groups|	|Data Points per Group|	Highest Simpson Reversal Likelihood by Initial Sequence	Generating Sequence	S(A) - S(B)	ζ
2	1	0/6	NA	NA	NA
2	2	0/70	NA	NA	NA
2	3	30/924 = 3.25%	Abbaab	10 - 11	1
2	4	402/12,870= 3.12%	Abbbaaab	19 - 17	1
2	5	26,872/184,756 = 14.54%	Aabbbbaaab	27 - 28	2
2	6	187,520/2,704,156 = 6.93%	Aaabbbbbabaa	38 - 40	2
2	7	8881034/40116600 = 22.14%	Aaabbbbbbaaaab	52 - 53	3

Note that the maximum *reversal* likelihood generating sequence for each case is not unique. In each case, one could transpose the elements ‘a’ and the elements ‘b’ to obtain the same *reversal* likelihood. We find that the maximum reversal likelihood generating sequence also generates the closest margin of victory in each case (i.e., 1 rank sum unit for n-odd cases and 2 rank sum units for n-even cases). While the *overall* likelihood of *reversal* is consistently below 0.02 for computed cases, *reversals* are found to be much more prevalent for certain initial sequences. In the $2x7$ case, the maximum initial sequence conditional likelihood of *reversal* is approximately 0.22, for example. The results of Figure 3 suggest that it is important to consider not only the statistical test but also the particular data (sequence) of interest when assessing prevalence of *Simpson reversals.* As with the overall likelihood of *reversal* for computed cases, we find that the maximum likelihood of *reversal* at the initial sequence level of the data strictly increases from the n to $n + 2$ case for the range of computed cases.

V. Conclusion

In this paper we have begun an investigation into the likelihood of *Simpson reversals*. Here our sole theoretical result was a sufficiency condition, but in a future paper we plan to provide both sufficient and necessary conditions in order to provide more accurate bounds on the score differential between groups A and B that either guarantee the existence of a *reversal* or make one impossible. Such results will then allow us to streamline our computational methods even more in order to assess larger sized groups.

The importance of being able to handle large samples is something our preliminary results here indicate. Though this is a first approach, we have shown that group size has an impact on the likelihood of *reversals*: as n increases, *reversals* become more possible in general. The individual cases of sequences displaying higher likelihoods of *reversal* themselves also see higher heights as n increases. More generally, of course, empirical data samples usually have relatively large n. This paper is a first step toward analyzing such data, and our future papers will continue to build on it.

References

Allison, V. J., & Goldberg, D. E. (2002). Species-Level versus Community-Level Patterns of Mycorrhizal Dependence on Phosphorus: An Example of Simpson's Paradox. *Functional Ecology, 16*(3), 346-352.

Bargagliotti, A. E. (2009). Aggregation and decision making using ranked data. *Mathematical Social Sciences, 58*(3), 354-366.

Boudreau, J., Ehrlich, J., Raza, M., & Sanders, S. (2018). The likelihood of social choice violations in rank sum scoring: algorithms and evidence from NCAA cross country running. *Public Choice, 174*(3-4), 219-238.

Boudreau, J., Ehrlich, J., Sanders, S., & Winn, A. (2014). Social choice violations in rank sum scoring: A formalization of conditions and corrective probability computations. *Mathematical Social Sciences, 71*, 20-29.

Foster, G., & Rahmstorf, S. (2011). Global temperature evolution 1979-2010. *Environmental Research Letters, 6*(4), 1-8.

Haunsperger, D. B. (2003). Aggregated statistical rankings are arbitrary. *Social Choice and Welfare, 20*(2), 261-272.

Haunsperger, D. B., & Saari, D. G. (1996). Paradoxes in nonparametric tests. *Canadian Journal of Statistics, 24*(1), 95-104.

Kock, N. (2015). How likely is Simpson's Paradox in path models? *International Journal of e-Collaboration, 11*(1), 1-7.

Nagaraja, H., & Sanders, S. (2020). The aggregation paradox for statistical rankings and nonparametric tests. *PLOS ONE, 15*(3), e0228627.

Pavlides, M. G., & Perlman, M. D. (2009). How likely is Simpson's paradox? *The American Statistician, 63*(3), 226-233.

Pineiro, G., Oesterheld, M., Batista, W. B., & Paruelo, J. M. (2006). Opposite changes of whole-soil vs. pools C:N ratios: a case of Simpson's paradox with implications on nitrogen cycling. *Global Change Biology, 12*, 804-809.

Simpson, E. H. (1951). The interpretation of interaction in contingency tables. *Journal of the Royal Statistical Society. Series B (Methodological), 13*, 238-241.

Yule, G. U. (1903). Notes on the theory of association of attributes in statistics. *Biometrika, 2*(2), 121-134.

Appendix I: Computational Code

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;
import java.util.SortedSet;
import java.util.TreeSet;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Justin Ehrlich
 */
```

```
public class SimpsonsParadox {

    private static int numGroups = 0;
    private static int numDataPoints = 0;

    int numPossibleInd = 0;
    int numIndWeakOnlyAnomalyTypeI = 0;
    int numIndWeakOnlyAnomalyTypeII = 0;
    int numPossibleCycles = 0;
    int numAnomaly = 0;
    int firstDegreeTransativityViolations = 0;
    int numWeakAnomaly = 0;
    int secondDegreeTransativityViolations = 0;
    int thirdDegreeTransativityViolations = 0;


    long numWinnerChanged = 0;
    long numWinnerChangedPossible = 0;

    long topNumWinnerChanged = 0;
    long topNumWinnerChangedPossible = 0;
    String topInitialSequence = "";

    Map<Long, Long> numIndependenceViolationCategoryOccurences = new HashMap<Long
, Long>();
    Map<Long, Long> numIndependenceViolationCategoryCycles = new HashMap<Long, Lo
ng>();


    Map<Long, Long> numSimponsParadoxViolationHighScore = new HashMap<Long, Long>
();
    Map<Long, Long> numSimponsParadoxViolationPossibleHighScore = new HashMap<Lon
g, Long>();

    public SimpsonsParadox() {
    }



    private void createShuffledEvent(String dataPoints, String originalDataPoints
, int numGroups, int numBins, int currentBin, int dataPointsAdded, char originalW
inner){

        if(dataPointsAdded == originalDataPoints.length()){
            char winner = findDependentWinner(dataPoints,numGroups);
            numWinnerChangedPossible++;
            if(winner == ' ' || originalWinner == ' '){ //weak
                return;
            }
            if(winner != originalWinner){
                numWinnerChanged++;
            }
            return;
        }
        if(currentBin >= numBins){
            //only allow for the correct number of bins. starts at 0 so should no
t equal numBins
            return;
        }
        for(int subsetSize=0; subsetSize <= originalDataPoints.length()-
dataPointsAdded; subsetSize++){
            String preString = dataPoints.substring(0, dataPointsAdded+currentBin
); //endIndex is exclusive, startIndex is inclusive
            String postString = dataPoints.substring(dataPointsAdded+currentBin);
            createShuffledEvent(preString + originalDataPoints.substring(dataPoin
tsAdded, dataPointsAdded+subsetSize) +
                postString,originalDataPoints,numGroups, numBins, currentBin+1, d
ataPointsAdded+subsetSize, originalWinner);

        }
    }
    //return ' ' if winner is tied
    private char findDependentWinner(String dataPoints, int numTeam){
        char[] groups = new char[numTeam];
        for(int i=0; i<numTeam; i++){
            groups[i] = (char) ('a' + (char)i);
        }
        int[] groupsScores = new int[numTeam];


        int counter = 0;
        for (int i = 0; i < dataPoints.length(); i++) {
            counter++;
            for(int j=0; j<numTeam; j++){
                if(dataPoints.charAt(i) == groups[j]){
                    groupsScores[j] = groupsScores[j]+counter;
                }
            }
        }
        int min=groupsScores[0];
        int minIndex = 0;
        for (int i = 0; i < numTeam; i++){
            if(groupsScores[i] < min){
                min=groupsScores[i];
                minIndex = i;
            }
        }
        //detect tie
        for(int i = 0; i < numGroups; i++){
            if(i != minIndex){
                if(groupsScores[i] == groupsScores[minIndex]){
                    return(' ');
                }
            }

        }
        return(groups[minIndex]);
    }


    private int findDependentWinnerScore(String dataPoints, int numTeam){
        char[] groups = new char[numTeam];
        for(int i=0; i<numTeam; i++){
            groups[i] = (char) ('a' + (char)i);
        }
        int[] groupsScores = new int[numTeam];


        int xScore = 0;
        int yScore = 0;
        int counter = 0;
        for (int i = 0; i < dataPoints.length(); i++) {
            counter++;
            for(int j=0; j<numTeam; j++){
                if(dataPoints.charAt(i) == groups[j]){
                    groupsScores[j] = groupsScores[j]+counter;
                }
            }
        }
        int min=groupsScores[0];
        int minIndex = 0;
        for (int i = 0; i < numTeam; i++){
            if(groupsScores[i] < min){
                min=groupsScores[i];
                minIndex = i;
            }
        }
        //detect tie
        for(int i = 0; i < numGroups; i++){
            if(i != minIndex){
                if(groupsScores[i] == groupsScores[minIndex]){
                    return(' ');
                }
            }

        }
        return(groupsScores[minIndex]);
    }

    private void findSimpsonsParadox(String dataPoints, int numGroups){
        long prevNumWinnerChanged = numWinnerChanged;
        long prevNumWinnerChangedPossible = numWinnerChangedPossible;
        char winner = findDependentWinner(dataPoints,numGroups);
        int numBins = dataPoints.length() + 1;
        createShuffledEvent(dataPoints,dataPoints,numGroups, numBins, 0, 0,winner
);
        long deltaNumWinnerChanged = numWinnerChanged - prevNumWinnerChanged;
        long deltaNumWinnerChangedPossible = numWinnerChangedPossible - prevNumWi
nnerChangedPossible;

        if(deltaNumWinnerChanged > topNumWinnerChanged){
            topInitialSequence = dataPoints;
            topNumWinnerChanged = deltaNumWinnerChanged;
            topNumWinnerChangedPossible = deltaNumWinnerChangedPossible;
        }

        long topScore = findDependentWinnerScore(dataPoints,numGroups);

        if (numSimponsParadoxViolationHighScore.containsKey(topScore)){
            numSimponsParadoxViolationHighScore.put(topScore, numSimponsParadoxVi
olationHighScore.get(topScore)+deltaNumWinnerChanged);
        } else{
            numSimponsParadoxViolationHighScore.put(topScore, (long)deltaNumWinne
rChanged);
        }

        if (numSimponsParadoxViolationPossibleHighScore.containsKey(topScore)){
            numSimponsParadoxViolationPossibleHighScore.put(topScore, numSimponsP
aradoxViolationPossibleHighScore.get(topScore)+topNumWinnerChangedPossible);
        } else{
            numSimponsParadoxViolationPossibleHighScore.put(topScore, (long)topNu
mWinnerChangedPossible);
        }

    }
    public static int countOccurrences(String haystack, char needle) {
        int count = 0;
        for (int i = 0; i < haystack.length(); i++) {
            if (haystack.charAt(i) == needle) {
                count++;
            }
        }
        return count;
    }
    private void generate(String dataPoints, int numGroups, int numDataPoints) th
rows IOException {
        for (int i = 0; i < numGroups; i++) {
            //don't allow more data points per group than numDataPoints
            if (countOccurrences(dataPoints, (char) (i + 97)) > numDataPoints) {
                return;
            }
        }
        //if more dataPoints are needed
        if (dataPoints.length() < numGroups * numDataPoints) {
            for (int i = 0; i < numGroups; i++) {
                generate(dataPoints + (char) (i + 97), numGroups, numDataPoints);
            }
        } else {
            permutationsCompleted++;
            findSimpsonsParadox(dataPoints, numGroups);
        }
    }
    public static void main(String[] args) throws IOException {
        int numGroups = 2;
        for(numDataPoints = 1; numDataPoints <= 7; numDataPoints++){
            SimpsonsParadox pR = new SimpsonsParadox();
            pR.generate("",numGroups, numDataPoints);
        }
    }

}
```