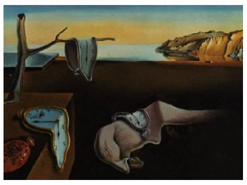

Journal of Statistical Software

October 2020, Volume 95, Code Snippet 1. doi: 10.18637/jss.v095.c01

Pseudo-Ranks: How to Calculate Them Efficiently in R

Martin Happ
University of Salzburg

Georg Zimmermann
University of Salzburg
Paracelsus Medical Private University

Edgar Brunner
University of Salzburg
University of Göttingen

Arne C. Bathke
University of Salzburg
University of Kentucky

Abstract

Many popular nonparametric inferential methods are based on ranks. Among the most commonly used and most famous tests are for example the Wilcoxon-Mann-Whitney test for two independent samples, and the Kruskal-Wallis test for multiple independent groups. However, recently, it has become clear that the use of ranks may lead to paradoxical results in case of more than two groups. Luckily, these problems can be avoided simply by using pseudo-ranks instead of ranks. These pseudo-ranks, however, suffer from being (a) at first less intuitive and not as straightforward in their interpretation, (b) computationally much more expensive to calculate. The computational cost has been prohibitive, for example, for large-scale simulative evaluations or application of resampling-based pseudo-rank procedures. In this paper, we provide different algorithms to calculate pseudo-ranks efficiently in order to solve problem (b) and thus render it possible to overcome the current limitations of procedures based on pseudo-ranks.

Keywords: nonparametric statistics, ranks, pseudo-ranks, R.

1. Introduction

There exist many rank-based inference methods, and they are used ubiquitously across the many subject matter areas where statistical inference is applied. Some of the best known examples include the Wilcoxon-Mann-Whitney test (Wilcoxon 1945; Mann and Whitney 1947) for inference regarding two independent samples and the Kruskal-Wallis test (Kruskal

Group sizes	Group	Weighted	Unweighted
20	1	0.635	0.727
10	2	0.388	0.500
5	3	0.185	0.273
5	1	0.815	0.727
10	2	0.612	0.500
20	3	0.365	0.273

Table 1: Weighted and unweighted relative effects for two different group allocations in case of normal distributions with $F_1 = N(1,1)$, $F_2 = N(0,1)$ and $F_3 = N(-1,1)$.

1952) for a comparison of multiple groups – each of the above publications having been cited thousands of times. For detecting ordered alternatives (patterned alternatives, trends), the Jonckheere-Terpstra test (Terpstra 1952; Jonckheere 1954) and the Hettmansperger-Norton test (Hettmansperger and Norton 1987; Brunner and Puri 2002) have become popular. Mann and Whitney (1947) used as an effect size the probability that an observation from the first group is less than an observation from the second group. This quantity is referred to as relative effect (Brunner and Puri 2001, 2002) with reference to Birnbaum and Klose (1957).

To extend this idea from two to multiple groups, one could simply consider all pairwise relative effects. However, these pairwise relative effects are not transitive, thus yielding potential paradoxical results. Specifically, for independent random variables X_i, $i = 1, 2, 3$, it may happen that each of the pairwise effects $p_{1,2} = \mathsf{P}(X_1 < X_2)$, $p_{2,3} = \mathsf{P}(X_2 < X_3)$, and $p_{3,1} = \mathsf{P}(X_3 < X_1)$ is less than $1/2$. This appears paradoxical as its interpretation is that X_1 tends to greater values than X_2, while X_2 tends to greater values than X_3, and finally X_3 to greater values than X_1. Concrete examples are provided, for instance, in Thangavelu and Brunner (2007) and Brunner, Konietschke, Bathke, and Pauly (2020).

A first step in solving this problem is to compare each group with one and the same reference group. For instance, one could choose a weighted mean of the cumulative distribution functions (CDFs) as the reference distribution. The effects obtained in this way are referred to as weighted relative effects, and many rank statistics are based on these effects. For example, in case of the Kruskal-Wallis test, the CDF of each group is compared with the weighted average of all CDFs involved in the trial where the weights are chosen as the proportion of group sizes divided by the total sample size. However, the weighted relative effects thus obtained depend on the ratios of group sizes. For example, let us consider the following example from Brunner, Konietschke, Pauly, and Puri (2017) given in Table 1. This example demonstrates that the weighted relative effects $p_i = \int W dF_i$, $i \in \{1, 2, 3\}$, even with known and fixed distribution functions, heavily depend on the ratio n_i/N where $W(x) = \frac{1}{N}\left(n_1F_1(x) + n_2F_2(x) + n_3F_3(x)\right)$ and $N = \sum_{j=1}^{3} n_j$. The distributions F_i, $i \in \{1, 2, 3\}$, in this example are normal distributions with variance 1 and expectations $\mu_1 = 1$, $\mu_2 = 0$, and $\mu_3 = -1$. The group sizes are either $n_1 = 20, n_2 = 10, n_3 = 5$ or $n_1 = 5, n_2 = 10, n_3 = 20$. In contrast, the unweighted relative effects

$$q_i = \int U dF_i, \tag{1}$$

$i \in \{1, 2, 3\}$, do not depend on the allocation rate to the groups. Here,

$$U(x) = \tfrac{1}{3}\left(F_1(x) + F_2(x) + F_3(x)\right)$$

denotes the unweighted mean of the distribution functions. Note that the quantities $p_i = \int W dF_i$ and q_i as defined in (1) in this example are not estimates, but the true effects. It is not desirable that these effects depend on the group sizes. Hence, a straightforward solution is to use unweighted relative effects, based on the unweighted average $U(x)$ of all CDFs. They lead in a natural manner to the so-called pseudo-ranks instead of ranks for the estimator of the group effects, see Section 2.

Other possible paradoxical outcomes when using weighted relative effects have been pointed out by Brunner (2017); Brunner *et al.* (2020) and Brunner, Bathke, and Konietschke (2019). These may be especially problematic for trend tests such as the Hettmansperger-Norton test (Hettmansperger and Norton 1987) as the trend can be drastically different if the ratios n_i/N are changed. Moreover, the trend may have opposite direction for the same set of distributions.

Although pseudo-ranks have already been considered by Kulle (1999) and Domhof (2001), they were first mentioned in the statistical literature by Brunner and Puri (2001) in the discussion at the end of their paper. Later, Thangavelu and Brunner (2007) and Konietschke, Hothorn, and Brunner (2012) derived general asymptotic results on pseudo-ranks. Also statistics based on pseudo-ranks have been published by Gao and Alvo (2005b,a); Gao, Alvo, Chen, and Li (2008); Konietschke *et al.* (2012), and Brunner *et al.* (2017). Nevertheless, they have not gained widespread popularity yet. This may be due to two major reasons: (1) Many users and statistics practitioners are not yet familiar with the paradoxical results that can arise using classical rank-based tests. (2) In standard statistical software, there is no efficient algorithm to calculate pseudo-ranks. Indeed, for example, no fast and efficient method to calculate pseudo-ranks has been implemented within the statistical software environment R (R Core Team 2020) so far. The methods available in the R packages **nparcomp** (Konietschke, Placzek, Schaarschmidt, and Hothorn 2015) and **rankFD** (Konietschke, Friedrich, Brunner, and Pauly 2020) rely on a direct calculation of the pseudo-ranks by using pairwise ranks (see Section 4.2). Such an approach can be very slow for a large number of groups.

As a remedy, and in order to open up the field of nonparametric statistics to a more widespread use of pseudo-ranks which will also solve the above-mentioned paradoxa that may arise when using simple rank-based tests, we provide a fast algorithm to calculate pseudo-ranks even for large data sets. In Section 5, we illustrate the application of this new algorithm in context of an artificial data example. Furthermore, we compare this new algorithm with the pairwise calculation of pseudo-ranks and the calculation based on count functions of all pairwise differences (see Section 6).

2. Defining ranks and pseudo-ranks

For ease of illustration, consider a one-way factorial model, assuming independent observations $X_{ik} \sim F_i$ from subjects $k \in \{1, 2, \ldots, n_i\}$ in groups $i \in \{1, 2, \ldots, a\}$. Here, $N = \sum_{i=1}^a n_i$ denotes the total sample size of all groups combined. Let

$$U := \frac{1}{a}\sum_{i=1}^{a} F_i \quad \text{and} \quad W := \frac{1}{N}\sum_{i=1}^{a} n_i F_i \tag{2}$$

denote the unweighted and the weighted mean distribution function, respectively. In the definitions above, we use the normalized versions of the distribution functions F_i (i.e., $F_i =$

$\frac{1}{2}[F_i^- + F_i^+]$), because the theoretical results can then be applied to continuous as well as ordinal variables in a straightforward manner, see for example Ruymgaart (1980); Akritas, Arnold, and Brunner (1997); Akritas and Brunner (1997) among others. We would like to mention that W is actually a special case of $\sum_{i=1}^a w_i F_i$, a weighted sum of the distribution functions $F_1, \dots, F_a$, where $w_i \geq 0$ and $\sum_{i=1}^a w_i = 1$. For example, stratified sampling schemes can thus be accounted for in the nonparametric model, by specifying the weights appropriately. This more general approach is discussed in detail in Brunner *et al.* (2020). Moreover, it should be noted that our model can easily be extended to multi-factorial designs, by splitting up the index i accordingly. Since in general, the functions U and W are unknown, we therefore consider their respective empirical versions

$$\widehat{U} := \frac{1}{a}\sum_{i=1}^{a} \widehat{F}_i \quad \text{and} \quad \widehat{W} := \frac{1}{N}\sum_{i=1}^{a} n_i \widehat{F}_i,$$

where $\widehat{F}_i = \frac{1}{2}\left(\widehat{F}_i^- + \widehat{F}_i^+\right)$, $i \in \{1, \dots, a\}$, are the normalized empirical distribution functions. Then, the mid-rank R_{ik} of an observation X_{ik} is defined by

$$R_{ik} = \frac{1}{2} + N\widehat{W}(X_{ik}) = \frac{1}{2} + \sum_{l=1}^{a}\sum_{m=1}^{n_l} c(X_{ik} - X_{lm})$$

and the mid pseudo-rank R_{ik}^{ψ} by

$$R_{ik}^{\psi} = \frac{1}{2} + N\widehat{U}(X_{ik}) = \frac{1}{2} + \frac{N}{a}\sum_{l=1}^{a} n_l^{-1} \sum_{m=1}^{n_l} c(X_{ik} - X_{lm}) \qquad (3)$$

for $i \in \{1, 2, \dots, a\}$ and $k \in \{1, 2, \dots, n_i\}$. Here, c is a function with $c(t) = 0, 1/2, 1$ depending on $t <, =, > 0$, respectively. If there are no ties (i.e., no equal values) in the data, we can simply sort the data from the smallest to the largest observation and assign rank 1 to the smallest, rank 2 to the second-smallest, and so on. This provides for a "natural" and easily interpretable way to calculate ranks. In case of ties, we have three options. We can assign to all observations with the same value the smallest rank, leading to the so-called min-ranks. These are sometimes used in competitions where two competitors with equal performance value are both assigned first place. It is also possible to assign the largest rank to them, thus leading to max-ranks. If we take the average of min- and max-ranks, we obtain the so-called mid-ranks which have been adopted in nonparametric statistics due to their favorable symmetry properties. In the remainder of this paper, we will also mostly use the latter and refer to the mid-ranks simply as ranks.

Note that both, ranks and pseudo-ranks, are invariant under strictly-monotone transformations and if $X_{ik} \leq X_{jl}$ then we also have $R_{ik} \leq R_{jl}$ and $R_{ik}^{\psi} \leq R_{jl}^{\psi}$. In case of equal group sizes (i.e., $n_1 = \cdots = n_a$), it is obvious that ranks and pseudo-ranks are identical.

Recently, it was pointed out by Thangavelu and Brunner (2007) and Brunner *et al.* (2020) that rank statistics may lead to paradoxical results. It was demonstrated in Table 1 in Section 1 that the true (theoretical) weighted relative effects depend on the group sizes. These relative effects are estimated by $\hat{p}_i = \int \widehat{W} d\widehat{F}_i$, $i \in \{1, \dots, a\}$, where $\widehat{W}(x) = \frac{1}{N}\sum_{i=1}^a n_i \widehat{F}_i$ which can be expressed in terms of the ranks of the observations. Similarly, estimators for the unweighted relative effects $\hat{q}_i$ can be written in terms of pseudo-ranks. Depending on the ratio $n_i / \sum_j n_j$, the order of the relative effects p_i may change. Therefore, it has been proposed to use pseudo-rank-based test statistics as a solution to avoid such paradoxical results. However, one problem of pseudo-ranks is that they are slightly more difficult to calculate, and existing algorithms have been computationally expensive. For example, a direct implementation of the definition in (3) would require N comparisons for each of the N observations, thus leading to $O(N^2)$ arithmetical operations. Therefore using the above definition in (3) to calculate pseudo-ranks is not feasible for large data sets, for tests using resampling techniques or for simulations. We will present three different algorithms. The second and third algorithm use the relation between ranks and pseudo-ranks. The first algorithm provides recursive formulas for pseudo-ranks and has been shown to be the fastest among those three, see Section 6. This algorithm is also implemented in the R package **pseudorank** (Happ, Zimmermann, Bathke, and Brunner 2020) available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=pseudorank`.

3. Efficient recursive calculation of pseudo-ranks

Existing efficient algorithms to compute ranks rely on very efficient sorting algorithms. For example, in R the “radix sort” algorithm is used for sorting the data. Following the sorting, rank 1 is assigned to the smallest observation, rank 2 to the second smallest, and so on. If some observations have the same value, the average rank is assigned to those. This can be calculated very fast with R. However, for pseudo-ranks, simply sorting the data is not enough as the increments from one pseudo-rank to the next are not 1 as for ranks, but they depend on the respective sizes of the groups to which the observations belong. Nevertheless, we can exploit a conceptually similar approach and propose a recursive formula for calculating pseudo-ranks.

Let us denote with $X_{(1)} \leq \cdots \leq X_{(N)}$ the order statistics and with $R^{\psi}_{(i)}$ the pseudo-rank of the order statistic $X_{(i)}$. The size of the sample i to which the observation $X_{(i)}$ belongs is denoted by $n_{(i)}$. For discrete random variables, the order statistics are not uniquely defined. But this does not matter, as only the order of the blocks with the same value is of importance.

Define $\boldsymbol{m} = (1/n_{(1)}, \dots, 1/n_{(N)})^\top$ to be the vector of the inverse group sizes, and let $\boldsymbol{t}^{(i)} = (1_{X_{(i)}=X_{(1)}}, \dots, 1_{X_{(i)}=X_{(N)}})^\top$ denote the vector indicating all observations with the same value as $X_{(i)}$. Now, the pseudo-ranks can be calculated recursively by

$$R^{\psi}_{(i)} = R^{\psi}_{(i-1)} + (1 - t_i^{(i-1)})\, \frac{N}{2a} \left(\boldsymbol{t}^{(i)} + \boldsymbol{t}^{(i-1)}\right)^\top \boldsymbol{m} \tag{4}$$

for $i = 2, \dots, N$ where $t_j^{(i)}$ refers to the jth component of the vector $\boldsymbol{t}^{(i)}$. The recursion start is given by

$$R^{\psi}_{(1)} = \frac{1}{2}\left(\frac{N}{a}(\boldsymbol{t}^{(1)})^\top \boldsymbol{m} + 1\right).$$

The derivation of this recursive representation is given in Appendix A. For this recursion formula still $O(N^2)$ arithmetic operations are necessary. The recursion in (4), however, can be written more efficiently distinguishing the two cases of no ties and of ties. This is considered in the following two sections. Note that this is just a simplification for programming. It is equally possible to rewrite (4) in terms of sums to avoid the vector products in (4). However, this may be more error-prone, since several cases have to be distinguished at the same time.

X_k	Group	R^{ψ}_{k}	$X_{(k)}$	Group	$R^{\psi}_{(k)}$
1	1	1.5	1	1	1.5
3	2	4.33	1.5	3	2.83
3.1	2	5.33	2	3	3.5
2	3	3.5	3	2	4.33
1.5	3	2.83	3.1	2	5.33
4	3	6.17	4	3	6.17

Table 2: Example for data without ties, where $n_i = i$ for $i = 1, 2, 3$, that is $N/(2a) = 1$.

3.1. The case of no ties

In this case, we obtain $(\boldsymbol{t}^{(i)})^\top \boldsymbol{m} = 1/n_{(i)}$ and $t_i^{(i-1)} = 0$ for $i = 2, \dots, N$. Therefore, the recursive formula simplifies to

$$R^{\psi}_{(i)} = R^{\psi}_{(i-1)} + \frac{N}{2a}\left(\frac{1}{n_{(i-1)}} + \frac{1}{n_{(i)}}\right), \tag{5}$$

$$R^{\psi}_{(1)} = \frac{1}{2}\left(\frac{N}{an_{(1)}} + 1\right). \tag{6}$$

Using this representation, we avoid calculating the vector products in Formula 4.

A small illustrative data example for this case is given in Table 2. Here, we have $N/a = 2$ and $n_i = i$ for $i = 1, 2, 3$. Then, for example, the pseudo-rank for the observation $X_6 = 4$ is calculated as $R^{\psi}_6 = 5.33 + 1/n_2 + 1/n_3 = 6.17$.

3.2. The case of ties

Let $T_k \subseteq \{1, 2, \dots, N\}$ denote the set of indices for the order statistics of all equal values for $X_{(k)}$ with

$$i = \min\{l \in T_k\} \quad \text{and} \quad j = \max\{l \in T_k\},$$

such that $X_{(k)} = X_{(l)}$ for all $l \in T_k$ and $X_{(k)} \neq X_{(l)}$ for $l \notin T_k$. Note that we can write $T_k = \{i, i+1, \dots, j-1, j\}$ because our sample and pseudo-ranks are ordered. To compute the pseudo-ranks in this case, we still use Formula 5 for all observations to obtain so-called "intermediate" pseudo-ranks $\tilde{R}^{\psi}_{(l)}$ which can be transformed easily into pseudo-ranks $R^{\psi}_{(l)}$ by

$$\begin{aligned} R^{\psi}_{(l)} &= \tilde{R}^{\psi}_{(l)} - \frac{N}{2a}\sum_{s=i}^{l}\left(\frac{1}{n_{(s)}} + \frac{1}{n_{(s-1)}}\right) + \frac{N}{2a}\left(\frac{1}{n_{(i-1)}} + \sum_{s=i}^{j}\frac{1}{n_{(s)}}\right) \\ &= \tilde{R}^{\psi}_{(i-1)} + \frac{N}{2a}\left(\frac{1}{n_{(i-1)}} + \sum_{s=i}^{j}\frac{1}{n_{(s)}}\right) \end{aligned} \tag{7}$$

for all $l \in T_k$ where we define $n_{(0)} = a/N$ and $\tilde{R}^{\psi}_{(0)} = 0$. By doing this recursive calculation in two steps, we only need $O(N)$ arithmetic operations instead of $O(N^2)$. Note that for $T_k = \{k\}$, Formula 7 simplifies to (5). Hence, $\tilde{R}^{\psi}_{(k)}$ equals $R^{\psi}_{(k)}$ in this case. This means,

X_k	Group	$X_{(k)}$	Group	$\tilde{R}^{\psi}_{(k)}$	$R^{\psi}_{(k)}$
1	1	1	1	1.5	1.5
3	2	2	2	3.00	3.33
3	3	2	3	3.83	3.33
2	2	3	2	4.67	5.00
2	3	3	3	5.50	5.00
4	3	4	3	6.17	6.17

Table 3: Example for data with ties, where $n_i = i$ for $i = 1, 2, 3$, that is $N/(2a) = 1$.

we only need to adjust for those observations $X_{(k)}$ which are tied with at least one other observation $X_{(l)}$ for $k \neq l$.

3.3. General algorithm (RECPR)

A general recursive algorithm for computing pseudo-ranks (RECPR algorithm) is obtained by summarizing the results from Sections 3.1 and 3.2.

RECPR algorithm

Step 1: First the sample is sorted while keeping the labels of the group specifications.

Step 2: Ignoring potential ties in the data, intermediate pseudo-ranks $\tilde{R}^{\psi}_{(k)}$ are computed using Equations 5 and 6 – see Section 3.1.

Step 3: Finally, the intermediate pseudo-ranks $\tilde{R}^{\psi}_{(k)}$ are replaced by the final pseudo-ranks $R^{\psi}_{(k)}$ using Equation 7.

It may be noted that for the RECPR algorithm, only $O(N)$ arithmetic operations (without sorting the data) are needed instead of $O(N^2)$ that are necessary when computing pseudo-ranks based on the count function in Equation 3. If we take the sorting into consideration then the RECPR algorithm has a time complexity of $O(N \log N)$ as the sort function from C++11 utilizes the introsort algorithm which has time complexity $O(N \log N)$ in the worst case, see for example Musser (1997).

This algorithm is demonstrated with the small example given in Table 3. Here, we have $X_{(2)} = X_{(3)} = 2$ and $X_{(4)} = X_{(5)} = 3$. All other observations are distinct from each other. For simplicity, we only state the pseudo-ranks for the sorted data. Note that the pseudo-ranks $R^{\psi}_{(2)}$ and $R^{\psi}_{(3)}$ remain the same when interchanging the order statistics $X_{(2)}$ and $X_{(3)}$ with each other as only the orders of the total blocks of ties matter. After sorting, we calculate the intermediate pseudo-ranks, that is, we just ignore the ties and simply use Formula 5. Then, in Step 3, we need to adjust for ties according to Formula 7. That is, we calculate

$$R^{\psi}_{(2)} = R^{\psi}_{(3)} = 1.5 + \tfrac{1}{n_1} + \tfrac{1}{n_2} + \tfrac{1}{n_3} = 3.33,$$
$$R^{\psi}_{(4)} = R^{\psi}_{(5)} = 3.83 + \tfrac{1}{n_2} + \tfrac{2}{n_3} = 5.00.$$

To obtain the set T_k for each observation $X_{(k)}$, the simplest solution is to check for equal values in a while-loop during Step 3 of the algorithm. Another possibility is to use mid-ranks to determine the end of a block of ties for an observation $X_{(k)}$, that is

$$j = 2\ R_{(i)} - i,$$

where $i = \min\{l \in T_k\}$ is the start of the block of tied values for $X_{(k)}$. If there are many ties in the data or the sample size is quite large then this algorithm will become slightly slower as potentially more intermediate pseudo-ranks have to be replaced in Step 3 of the algorithm. But overall, there are only $O(N)$ arithmetic calculations necessary for this algorithm.

Similarly, we obtain algorithms for minimum and maximum pseudo-ranks. For minimum pseudo-ranks, we simply replace the count function c in (3) by the function $c^-(x) = 0, 1$ for $x \leq 0$ and $x > 0$, thus leading to left-continuous empirical distribution functions. For maximum pseudo-ranks, we use the function $c^+(x) = 0, 1$ for $x < 0$ and $x \geq 0$ which results in right-continuous empirical distribution functions. Then maximum pseudo-ranks $R_{ik}^{\psi+}$ and minimum pseudo-ranks $R_{ik}^{\psi-}$ are defined by

$$\begin{aligned} R_{ik}^{\psi+} &= \frac{N}{a} \sum_{l=1}^{a} n_l^{-1} \sum_{m=1}^{n_l} c^+(X_{ik} - X_{lm}), \\ R_{ik}^{\psi-} &= 1 + \frac{N}{a} \sum_{l=1}^{a} n_l^{-1} \sum_{m=1}^{n_l} c^-(X_{ik} - X_{lm}). \end{aligned}$$

Clearly, mid pseudo-ranks are then the average of maximum and minimum pseudo-ranks. For more details, we refer to Appendix B.

The algorithm "recursive calculation" proposed above is implemented in the R package **pseudorank** as a S3 method. However, the recursive algorithm requires the data to be sorted. This can be done quite efficiently in R. The recursive calculation of the pseudo-ranks itself is implemented in C++ and integrated into the R environment using the R package **Rcpp** from Eddelbuettel and François (2011). We use C++ instead of R directly because the R language is not very suited for this type of calculation with for-loops, see, for example Morandat, Hill, Osvald, and Vitek (2012). The function to calculate pseudo-ranks is called `pseudorank`. It requires either two vectors, one denoting the data and one denoting the groups, or a formula object and a data frame as arguments, see, for example, the following code.

```
R> library("pseudorank")
R> df <- data.frame(data = c(1, 2, 2, 3, 4),
+    group = as.factor(c(1, 1, 2, 2, 3)))
R> pseudorank(df$data, df$group)

[1] 0.9166667 2.1666667 2.1666667 3.4166667 4.6666667

R> pseudorank(data ~ group, df)

[1] 0.9166667 2.1666667 2.1666667 3.4166667 4.6666667
```

The function `pseudorank` calculates by default "mid" pseudo-ranks (obtained by setting the argument `ties.method = "average"`). It is also possible to calculate "minimum" or "maximum" pseudo-ranks by using `ties.method = "min"` or `ties.method = "max"` respectively.

Averaging the minimum and maximum pseudo-ranks yields the mid pseudo-ranks, this is the same as with ranks.

If the data set contains missing values, then these can either be removed or kept and put at the beginning or end of the data vector. This is the same as for the `rank` function from base R, and our function `pseudorank` shall be considered an extension of this `rank` function. Therefore we decided to provide the same functionality. However, we would strongly recommend to use the standard argument `na.last = NA` to remove the missing values. If `NA` values are kept, then the pseudo-ranks for those `NA` values are not uniquely defined if there is more than one missing value, as the pseudo-ranks depend on the order in which they appear in the vector. In contrast, the function `rank` uses `na.last = TRUE` as its standard argument. Even for that base R function, we would advise to use `na.last = NA` unless there are special circumstances where it is necessary to keep missing values. Note that using that function, conventional ranks are also not uniquely defined for missing values. In particular, in the function `rank`, the argument `ties.method = "average"` is ignored for missing values, that is, they are implicitly assumed to be distinct. See the following R code on how missing values can be handled for pseudo-ranks. For illustration, consider the following artificial data with one missing value.

```
R> df <- data.frame(data = c(NA, 2, 2, 3, 4),
+    group = as.factor(c(1, 1, 2, 2, 3)))
```

We showcase all three variants of handling missing values. First, we put the missing values last.

```
R> pseudorank(data ~ group, data = df, na.last = TRUE)

[1] 5.083333 1.333333 1.333333 2.583333 3.833333
```

Another option is to put the missing values at the beginning.

```
R> pseudorank(data ~ group, data = df, na.last = FALSE)

[1] 0.9166667 2.1666667 2.1666667 3.4166667 4.6666667
```

However, our recommended variant is to remove missing values entirely from the data set.

```
R> pseudorank(data ~ group, data = df, na.last = NA)

[1] 1.500000 1.500000 2.833333 3.833333
```

To calculate minimum or maximum pseudo-ranks, the arguments `ties.method = "min"` and `ties.method = "max"`, respectively, can be used in the function `pseudorank`. For the usage, see the following R code.

```
R> df <- data.frame(data = c(1, 7, 1, 2, 3, 3, 5.5, 6, 7),
+    group = as.factor( c(1, 1, 1, 2, 2, 3, 3, 3, 3)))
R> pseudorank(df$data, df$group, ties.method = "max")

[1] 2.00 9.00 2.00 3.50 5.75 5.75 6.50 7.25 9.00
```

4. Further algorithms to calculate pseudo-ranks

4.1. Computation based on count functions

The definition of pseudo-ranks, see (3), relies on computing the count function for all pairwise differences of the sample. That is, we need to calculate N differences for each observation X_k, $k \in \{1, \ldots, N\}$. This implies that $O(N^2)$ arithmetic operations are necessary when using this method. The calculation of one pseudo-rank R_k^{ψ} can be programmed in a vectorized form with R. But we still need at least one for-loop to calculate all N pseudo-ranks.

For large samples, this algorithm can be improved by using parallelization. But for the simulation in Section 6 we did not parallelize the code as this would be counterproductive for small samples. Furthermore, even with parallelization, the algorithm stays highly inefficient as $O(N^2)$ arithmetic operations are necessary. In contrast, the RECPR algorithm is not suited for parallelization as this algorithm is recursively defined.

4.2. Computation based on pairwise ranks

Another way to calculate pseudo-ranks is by using so-called internal and pairwise ranks. This algorithm is used in the R package **rankFD**. Here, we denote with $R_{ik}^{(ir)}$ the rank of X_{ik} over all observations from groups i and r (pairwise ranks). Accordingly for $R_{ik}^{(i)}$, we do the ranking over all observations from group i (internal ranks). Then, the following representation of pseudo-ranks holds.

$$R_{ik}^{\psi} = \frac{1}{2} + \frac{N}{a}\left[\sum_{r \neq i} \frac{1}{n_r}\left(R_{ik}^{(ir)} - R_{ik}^{(i)}\right) + \frac{1}{n_i}\left(R_{ik}^{(i)} - \frac{1}{2}\right)\right]$$

This algorithm works well for a small number of groups a but gets worse very quickly as a increases. This can be seen in the simulation study in Section 6. In comparison, the performance of the recursive algorithm does not depend on the number of groups.

4.3. Computation based on the AB algorithm

In Section 2 we have already seen from Formula 2 that in case of equal sample sizes, ranks are equal to pseudo-ranks. Therefore we can calculate pseudo-ranks via ranks even though this may be inefficient sometimes. We can use this relation to state the following algorithm where we artificially balance the groups:

1. Calculate the least common multiple (LCM) of the sample sizes $n_1, \ldots, n_a$.
2. Artificially balance the groups by amplifying the data such that the new sample sizes are given by $n_1^* = \cdots = n_a^* = \mathrm{LCM}(n_1, \ldots, n_a)$ and $N^* = an_1^*$. That is, each observation X_{ik} appears $\lambda_i = n_i^*/n_i$ times in the amplified data set.
3. Calculate the ranks based on the amplified data. Note that the empirical CDFs of the different groups are the same as for the original data because all observations within one group are amplified by the same factor λ_i. Therefore, the pseudo-ranks of the original data are a linear function of the ranks based on the amplified data. Let R_{ik}^{A} denote the

rank of the observation X_{ik} based on the amplified data and R^{ψ}_{ik} the pseudo-rank based on the original observations. Then we have the following relationship:

$$R^{\psi}_{ik} = \frac{N}{N^*}\left(R^{A}_{ik} - \frac{1}{2}\right) + \frac{1}{2}.$$

4. Restrict the amplified data set to the "original" (non-amplified) observations.

This AB algorithm (artificially balancing algorithm) works well as long as the amplification factor $\lambda_i = n^*_i/n_i$ and the sample sizes are not too large, for example $\lambda_i \leq 2$. In particular, some aspects of the statistical software environment R are not very suited for large data sets. By amplifying our data, we would lose most of the advantages of this algorithm using a '`data.frame`' to store the data. Hence for large data sets, the data structure '`data.table`' from the R package with the same name should be used, see Dowle and Srinivasan (2020). Overall the AB algorithm is more useful to show the connection between ranks and pseudo-ranks, namely, that pseudo-ranks are merely affine transformations of ranks based on an amplified data set. However for real applications, we recommend using the RECPR algorithm from Section 3.

5. Application of pseudo-ranks

Many nonparametric hypothesis tests are based on the weighted relative effects. A classical example for nonparametric trend tests is the Hettmansperger-Norton test (Hettmansperger and Norton 1987). As previously discussed in Sections 1 and 2, ranks may lead to paradoxical results. Therefore, we have implemented a pseudo-rank-based analog to the Hettmansperger-Norton test in the package **pseudorank** for illustration. It is also possible to use ranks for calculating the test statistic by setting the argument `pseudoranks = FALSE`. For the application of pseudo-ranks, we consider an artificial data set included in the package **pseudorank** where a substance was administered in three different concentrations (1, 2 and 3). The data set is generated from a mixture of normal distributions in each group which is basically based on an example about tricky dice (Brunner 2017). The data is given in Table 4 along with the calculated pseudo-ranks and ranks. Other examples, where pseudo-ranks are used in context of real data examples, can be found in Brunner *et al.* (2019).

First we calculate the pseudo-rank and rank for each observation and compute the weighted and unweighted relative effects with the function `summaryBy` from the package **doBy** (Højsgaard and Halekoh 2020). Note that we are only interested in $\hat{q}_i = \int \widehat{U} d\widehat{F}_i$ and $\hat{p}_i = \int \widehat{W} d\widehat{F}_i$, hence we need to subtract 1/2 from the pseudo-ranks and divide the result by the total sample size $N = 54$.

```
R> library("pseudorank")
R> library("doBy")
R> dat[, "ranks"] <- (rank(dat[, "score"]) - 1/2)/N
R> dat[, "pseudoranks"] <- (pseudorank(score ~ conc, data = dat) - 1/2)/N
R> summaryBy(score + ranks + pseudoranks ~ conc, data = dat, FUN = mean)

  conc score.mean ranks.mean pseudoranks.mean
1    1   4.333333  0.4629630              0.5
2    2   4.330556  0.4907407              0.5
3    3   4.333333  0.5462963              0.5
```

Concentration	Score
1	0.5 3.8 4.1 5.6 6.2 5.8
2	1.3 1.9 1.5 6.7 7 6.9 1.9 1.8 1.1 7.2 7.5 6.6 2.1 2 1.4 6.9 7.1 7.5 1.1
	1.6 1.8 7.3 7 6.7 1.7 2.1 1.4 6.8 7.6 6.6 1.4 2 1.1 7.2 7.5 6.6
3	2.3 3.1 2.7 5 4.6 8.1 2.4 3 2.9 5.3 4.4 8.2
	Pseudo-ranks
1	2 23 26 35 41 38
2	5.25 10 7.25 44.50 47 46 10 9 4.25 48.50 50.25 43.25 12
	11 6.25 46 47.75 50.25 4.25 7.75 9 49.25 47 44.50 8.25 12
	6.25 45.25 51.25 43.25 6.25 11 4.25 48.50 50.25 43.25
3	13.25 20.75 16.25 31.25 29.75 52.25 14.75 19.25 17.75 32.75 28.25 53.75
	Ranks
1	1 26 27 32 34 33
2	5 14.5 9 38.5 43.5 41.5 14.5 12.5 3 46.5 50 36 18.5 16.5 7 41.5 45 50 3
	10 12.5 48 43.5 38.5 11 18.5 7 40 52 36 7 16.5 3 46.5 50 36
3	20 25 22 30 29 53 21 24 23 31 28 54

Table 4: Simulated data of three groups ($n_1 = 6, n_2 = 36, n_3 = 12$) with pseudo-ranks and ranks.

Group	Mean	$\widehat{p}_i$	$\widehat{q}_i$
1	4.33	0.46	0.5
2	4.33	0.49	0.5
3	4.33	0.55	0.5

Table 5: Unweighted and weighted relative effects for the data given in Table 4.

In Table 5, the weighted and unweighted relative effects as well as the means for each group of the data set are summarized. There is an increasing trend for the weighted relative effects. In contrast, the unweighted effects and the means are identical.

If we use this artificial data set to draw with replacement new observations and thus generate larger group sizes, we can apply the Hettmansperger-Norton test from the package **pseudorank** in order to test for an increasing trend. If we choose $n_1 = 60, n_2 = 360$ and $n_3 = 120$ then we obtain a significant result using ranks and a non-significant result with pseudo-ranks. Hence, the usage of ranks would lead to a completely different answer than by using a pseudo-rank based (or possibly even parametric) trend test. For different ratios of group sizes it is also possible to construct cases where we have a significant decreasing trend for the weighted relative effects but the unweighted relative effects and group means are still identical as they do not depend on the ratios of group sizes. Therefore, we recommend pseudo-ranks instead of ranks to avoid possible paradoxical results in case of unequal sample sizes.

```
R> hettmansperger_norton_test(score ~ conc, data = dat2, pseudoranks = FALSE,
+    alternative = "increasing")

Hettmansperger-Norton Trend Test

Call:
```

```
score ~ conc

Alternative:  increasing
Test Statistic:  2.280412
Distribution of Statistic:  Standard-Normal
unweighted relative Effects / Pseudo-ranks:  FALSE
p-Value:  0.01129162

R> hettmansperger_norton_test(score ~ conc, data = dat2, pseudoranks = TRUE,
+    alternative = "increasing")

Hettmansperger-Norton Trend Test

Call:
score ~ conc

Alternative:  increasing
Test Statistic:  0.1933954
Distribution of Statistic:  Standard-Normal
unweighted relative Effects / Pseudo-ranks:  TRUE
p-Value:  0.4233246
```

The function `hettmansperger_norton_test` also returns the vector $\hat{\boldsymbol{q}}$ or $\hat{\boldsymbol{p}}$ of estimated relative effects. Hence, it is not necessary to calculate the pseudo-ranks manually and to use `summaryBy` to compute the mean for each group.

```
R> hettmansperger_norton_test(score ~ conc, data = dat,
+    pseudoranks = TRUE)$pHat

[1] 0.5 0.5 0.5
```

6. Benchmark study

We compared the three algorithms (RECPR, pairwise, and AB) in this paper with the function `rank` from R and the direct calculation of pseudo-ranks, see the definition in Formula 3. We will refer to the direct calculation simply as "count" algorithm. The code for the pairwise calculation was taken from the package **rankFD**. For the RECPR algorithm we used the function `pseudorank` from the package **pseudorank** and used the data and group vector as arguments as in our case it was slightly faster than the version with '`formula`' objects. Note that we are comparing an S3 method with non-generic functions. But in simulations the overhead from the S3 method dispatch has proven to be negligible in our case.

Because of the similarities between ranks and pseudo-ranks, it should not be possible to calculate pseudo-ranks faster than ranks. Therefore the comparison with the `rank` function tells us somewhat how close our algorithms are to the optimum. For all simulations, we used R version 3.4.1 running on Windows 7 x64 (build 7601) with a 3.2 GHz CPU and 8 GB RAM. The tables and figures for the simulation results were created with the R packages **xtable** (Dahl, Scott, Roosen, Magnusson, and Swinton 2019) and **ggplot2** (Wickham 2016).

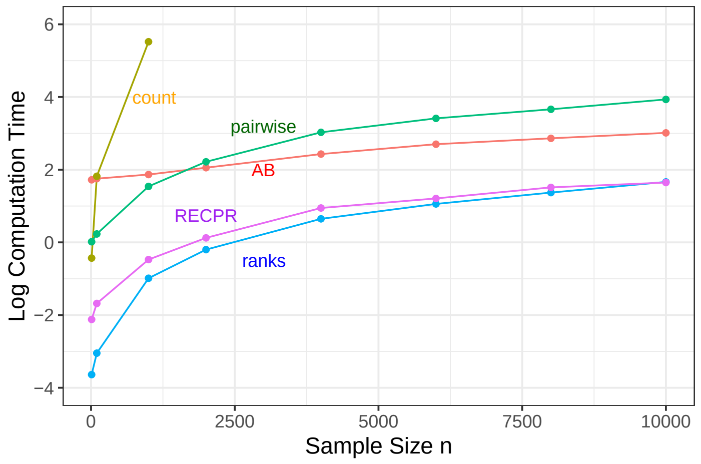

Figure 1: Simulation for three groups with ties in the data; logarithmized median computation times in milliseconds.

We repeated each method (ranks calculation, RECPR algorithm, AB algorithm, pairwise algorithm and count algorithm) 1000 times and measured the computation time with the function `microbenchmark` from the R package **microbenchmark** (Mersmann 2019). These times were noticeably right-skewed, we therefore only report the median computation time for each method.

We first considered a design with $a = 3$ groups with sample sizes $n_1 = n_2 = n$ and $n_3 = 2 \cdot n$ where $n = 10, 100, 1000, 2000, 4000, 6000, 8000, 10000$. This would be an optimal situation for the AB algorithm as only the first two groups need to be amplified by the factor two. We simulated normally distributed data and applied the `round` function in order to artificially create ties in the data as this is the worst case scenario for the recursive algorithm. This means that more "intermediate" pseudo-ranks have to be replaced by pseudo-ranks in a second for-loop. The results for this simulation are given in Table 6, and the logarithmized computation times are presented in Figure 1.

The AB algorithm performed quite well for the largest simulated sample size ($n = 10000$) in this scenario but this algorithm was still slower than the recursive method. The recursive algorithm was the fastest among the four pseudo-rank methods for all sample sizes. But there is still quite a bit of a difference between the rank and the recursive calculation for smaller sample sizes. This may be due to the fact that for pseudo-ranks we need to sort two vectors whereas for the calculation of ranks it is sufficient to only sort the vector containing the original observations. The pairwise algorithm did not perform well for large sample sizes even though there were only three groups in this setting. The calculation based on count functions took considerably longer than any of the other methods. However, this outcome was expected as this algorithm is quite inefficient, see Section 4.1.

In a second simulation we considered sample sizes $n_1 = \cdots = n_4 = n$ and $n_5 = 2 \cdot n$ for $n = 10, 100, \ldots, 10000$. The results are displayed in Table 7 and Figure 2. The recursive algorithm was only barely affected by adding more groups and thus increasing the overall sample size slightly. But the pairwise algorithm was considerably slower than in the previous simulation for three groups. The AB algorithm was slightly slower than before but is still faster for large sample sizes than the pairwise algorithm in this special situation where we

n	Ranks	RECPR	AB	Pairwise	Count
10	0.03	0.12	5.59	1.02	0.65
100	0.05	0.19	5.78	1.26	6.17
1000	0.37	0.62	6.46	4.66	249.66
2000	0.82	1.13	7.81	9.18	959.32
4000	1.91	2.58	11.35	20.66	3537.97
6000	2.88	3.35	14.89	30.32	8111.88
8000	3.93	4.54	17.51	38.92	13354.55
10000	5.28	5.19	20.35	51.07	21170.45

Table 6: Simulation for three groups with ties in the data; median computation times in milliseconds.

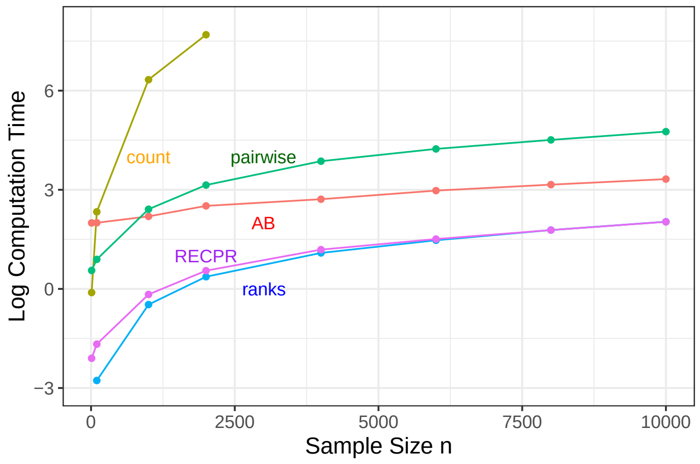

Figure 2: Simulation for five groups; with ties in the data; logarithmized median computation times in milliseconds.

only need to duplicate each observation for the first four groups. We also observed that for $n \geq 8000$, the recursive calculation was as fast as the rank calculation. The differences between both methods were negligible (≤ 0.1 ms). This was somewhat surprising as more calculations are necessary for pseudo-ranks. But this is probably due to the fact that we use a slightly different method for sorting the data. Namely, we use the C++ function `sort` from namespace **std** with a custom comparator to return the order of a vector. This approach has shown to be faster than the corresponding R function `order`. For $n = 10^4$, our RECPR algorithm took about 8.70 ms using the R function `order` and only 7.62 ms using C++. In comparison, the `rank` function took about 7.64 ms.

For the third and last simulation, we considered twelve groups where the sample sizes were $n_1 = \cdots = n_{11} = n$ and $n_{12} = 2 \cdot n$ with $n = 10, 100, \ldots, 1000$. As we can see in Table 8 or Figure 3, the AB algorithm performed considerably worse as data from 11 groups needed to be amplified. Clearly, the pairwise algorithm also took substantially longer as even more pairwise and internal ranks had to be calculated. The recursive calculation was not affected by increasing the number of groups.

n	Ranks	RECPR	AB	Pairwise	Count
10	0.03	0.12	7.37	1.75	0.90
100	0.06	0.19	7.40	2.44	10.31
1000	0.62	0.85	8.98	11.15	562.33
2000	1.44	1.74	12.33	23.17	2192.54
4000	2.97	3.28	15.10	47.72	8256.42
6000	4.36	4.52	19.58	69.21	18178.49
8000	5.94	5.93	23.47	90.89	32313.07
10000	7.64	7.62	27.75	116.74	50773.44

Table 7: Simulation for five groups; with ties in the data; median computation times in milliseconds.

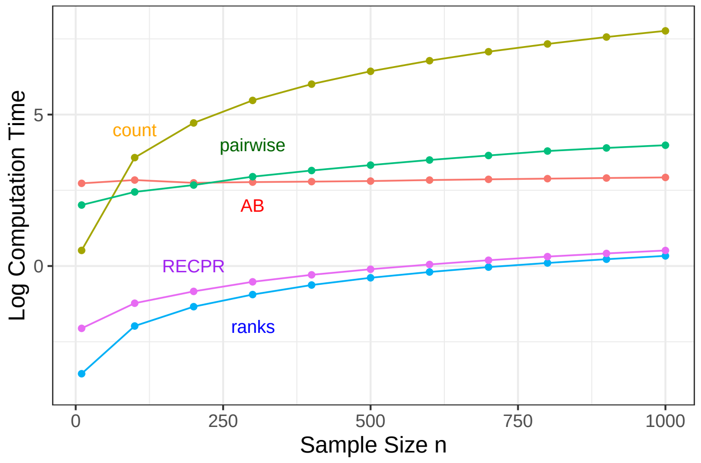

Figure 3: Simulation for 12 groups; with ties in the data; logarithmized median computation times in milliseconds.

n	Ranks	RECPR	AB	Pairwise	Count
10	0.03	0.13	15.33	7.49	1.67
100	0.14	0.29	17.09	11.53	35.83
200	0.26	0.43	15.59	14.48	112.65
300	0.39	0.59	15.97	19.10	236.98
400	0.53	0.75	16.23	23.41	405.72
500	0.68	0.90	16.53	27.98	620.00
600	0.82	1.05	17.09	33.14	879.38
700	0.96	1.21	17.50	38.45	1183.33
800	1.11	1.36	17.90	44.54	1528.63
900	1.25	1.51	18.28	49.34	1921.00
1000	1.40	1.67	18.63	53.96	2355.98

Table 8: Simulation for 12 groups; with ties in the data; median computation times in milliseconds.

7. Conclusion

Many rank-based inference methods have the disadvantage that even when using the same model with the same distribution functions, different allocation ratios may lead to completely different results, also in cases where the total sample size stays the same. This is caused by the (weighted) relative effects which are used for these rank statistics and has been pointed out recently, for example, by Brunner *et al.* (2020). These effects depend on the group sizes. This undesirable property of rank tests can be solved by using pseudo-ranks which correspond to unweighted relative effects. But as we have seen in the simulation results, the calculation of pseudo-ranks just by using their definition is not wise in terms of computational cost. Other algorithms such as the pairwise algorithm which are already used by some R packages can be quite slow if the number of groups is large. Hence, this is a problem if we want to compute statistical tests relying on some form of resampling or if we want to perform power simulations.

Therefore new algorithms are needed. In this paper we have presented three algorithms to calculate pseudo-ranks. The AB algorithm mainly demonstrates the relation between ranks and pseudo-ranks but is not very suitable for practical applications. The computation time for the pairwise algorithm heavily depends on the number of groups, as already discussed. The newly proposed recursive calculation (RECPR) was clearly the best among those four methods compared in our simulations. We provide for this recursive calculation C++ code implemented in an R package called **pseudorank** which is available on CRAN at `https://CRAN.R-project.org/package=pseudorank/` and on GitHub at `https://github.com/happma/pseudorank`.

Acknowledgments

The research was supported by the Austrian Science Fund (FWF): I 2697-N31 and by the WISS 2025 (Science and Innovation Strategy Salzburg 2025) project "IDALab Salzburg" (20102-F1901166-KZP, 20204-WISS/225/197-2019 and 20102-F1901166-KZP).

References

Akritas MG, Arnold SF, Brunner E (1997). "Nonparametric Hypotheses and Rank Statistics for Unbalanced Factorial Designs." *Journal of the American Statistical Association*, **92**(437), 258–265. `doi:10.1080/01621459.1997.10473623`.

Akritas MG, Brunner E (1997). "A Unified Approach to Rank Tests for Mixed Models." *Journal of Statistical Planning and Inference*, **61**(2), 249–277. `doi:10.1016/s0378-3758(96)00177-2`.

Birnbaum ZW, Klose OM (1957). "Bounds for the Variance of the Mann-Whitney Statistic." *The Annals of Mathematical Statistics*, **28**(4), 933–945. `doi:10.1214/aoms/1177706794`.

Brunner E (2017). "Ranks and Pseudoranks – Paradoxical Results of Rank Procedures in Case of Unequal Sample Sizes." 13th Workshop SMSA, Berlin, February, 2017.

Brunner E, Bathke AC, Konietschke F (2019). *Rank- and Pseudo-Rank Procedures for Independent Observations in Factorial Designs: Using R and SAS*. Springer-Verlag. doi:10.1007/978-3-030-02914-2.

Brunner E, Konietschke F, Bathke AC, Pauly M (2020). "Ranks and Pseudo-Ranks – Surprising Results of Certain Rank Tests in Unbalanced Designs." *International Statistical Review*. Forthcoming.

Brunner E, Konietschke F, Pauly M, Puri ML (2017). "Rank-Based Procedures in Factorial Designs: Hypotheses about Non-Parametric Treatment Effects." *Journal of the Royal Statistical Society B*, **79**(5), 1463–1485. doi:10.1111/rssb.12222.

Brunner E, Puri ML (2001). "Nonparametric Methods in Factorial Designs." *Statistical Papers*, **42**(1), 1–52. doi:10.1007/s003620000039.

Brunner E, Puri ML (2002). "A Class of Rank-Score Tests in Factorial Designs." *Journal of Statistical Planning and Inference*, **103**(1–2), 331–360. doi:10.1016/s0378-3758(01)00230-0.

Dahl DB, Scott D, Roosen C, Magnusson A, Swinton J (2019). ***xtable**: Export Tables to LaTeX or HTML*. R package version 1.8-4, URL https://CRAN.R-project.org/package=xtable.

Domhof S (2001). *Nichtparametrische Relative Effekte*. Ph.D. thesis, University of Göttingen. URL http://hdl.handle.net/11858/00-1735-0000-000D-F284-4.

Dowle M, Srinivasan A (2020). ***data.table**: Extension of '`data.frame`'*. R package version 1.13.0, URL https://CRAN.R-project.org/package=data.table.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Gao X, Alvo M (2005a). "A Nonparametric Test for Interaction in Two-Way Layouts." *Canadian Journal of Statistics*, **33**(4), 529–543. doi:10.1002/cjs.5550330405.

Gao X, Alvo M (2005b). "A Unified Nonparametric Approach for Unbalanced Factorial Designs." *Journal of the American Statistical Association*, **100**(471), 926–941. doi:10.1198/016214505000000042.

Gao X, Alvo M, Chen J, Li G (2008). "Nonparametric Multiple Comparison Procedures for Unbalanced One-Way Factorial Designs." *Journal of Statistical Planning and Inference*, **138**(8), 2574–2591. doi:10.1016/j.jspi.2007.10.015.

Happ M, Zimmermann G, Bathke AC, Brunner E (2020). ***pseudorank**: Pseudo-Ranks*. R package version 1.0.1, URL https://CRAN.R-project.org/package=pseudorank.

Hettmansperger TP, Norton RM (1987). "Tests for Patterned Alternatives in k-Sample Problems." *Journal of the American Statistical Association*, **82**(397), 292–299. doi:10.1080/01621459.1987.10478432.

Højsgaard S, Halekoh U (2020). ***doBy**: Groupwise Statistics, LSmeans, Linear Contrasts, Utilities*. R package version 4.6.6, URL https://CRAN.R-project.org/package=doBy.

Jonckheere AR (1954). "A Distribution-Free k-Sample Test Against Ordered Alternatives." *Biometrika*, **41**(1–2), 133–145. doi:10.1093/biomet/41.1-2.133.

Konietschke F, Friedrich S, Brunner E, Pauly M (2020). ***rankFD**: Rank-Based Tests for General Factorial Designs.* R package version 0.0.5, URL https://CRAN.R-project.org/package=rankFD.

Konietschke F, Hothorn LA, Brunner E (2012). "Rank-Based Multiple Test Procedures and Simultaneous Confidence Intervals." *Electronic Journal of Statistics*, **6**, 738–759. doi:10.1214/12-ejs691.

Konietschke F, Placzek M, Schaarschmidt F, Hothorn LA (2015). "**nparcomp**: An R Software Package for Nonparametric Multiple Comparisons and Simultaneous Confidence Intervals." *Journal of Statistical Software*, **64**(9), 1–17. doi:10.18637/jss.v064.i09.

Kruskal WH (1952). "A Nonparametric Test for the Several Sample Problem." *The Annals of Mathematical Statistics*, **23**(4), 525–540.

Kulle B (1999). *Nichtparametrisches Behrens-Fisher-Problem im Mehr-Stichprobenfall.* Diploma thesis, Institute of Math. Stochastics, University of Göttingen.

Mann HB, Whitney DR (1947). "On a Test of Whether One of Two Random Variables Is Stochastically Larger than the Other." *The Annals of Mathematical Statistics*, **18**(1), 50–60.

Mersmann O (2019). ***microbenchmark**: Accurate Timing Functions.* R package version 1.4-7, URL https://CRAN.R-project.org/package=microbenchmark.

Morandat F, Hill B, Osvald L, Vitek J (2012). "Evaluating the Design of the R Language." In J Noble (ed.), *ECOOP 2012 – Object-Oriented Programming*, pp. 104–131. Springer-Verlag, Berlin, Heidelberg.

Musser DR (1997). "Introspective Sorting and Selection Algorithms." *Software: Practice and Experience*, **27**(8), 983–993.

R Core Team (2020). *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Ruymgaart FH (1980). "A Unified Approach to the Asymptotic Distribution Theory of Certain Midrank Statistics." In *Statistique Non Parametrique Asymptotique*, pp. 1–18. Springer-Verlag. doi:10.1007/bfb0097422.

Terpstra TJ (1952). "The Asymptotic Normality and Consistency of Kendall's Test Against Trend, When Ties Are Present in One Ranking." *Indagations Mathematicae*, **14**, 327–333. doi:10.1016/s1385-7258(52)50043-x.

Thangavelu K, Brunner E (2007). "Wilcoxon-Mann-Whitney Test for Stratified Samples and Efron's Paradox Dice." *Journal of Statistical Planning and Inference*, **137**(3), 720–737. doi:10.1016/j.jspi.2006.06.005.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis.* 2nd edition. Springer-Verlag. doi:10.1007/978-3-319-24277-4. URL https://ggplot2.tidyverse.org/.

Wilcoxon F (1945). "Individual Comparisons by Ranking Methods." *Biometrics Bulletin*, **1**(6), 80–83. doi:10.2307/3001968.

A. Derivation of the recursive algorithm

In order to prove Formula 4, consider again the order statistics $X_{(1)}, \ldots, X_{(n)}$ of our sample. For the recursion start, it is clear that the summands of $\widehat{U}(X_{(1)})$ are count functions of the form $\frac{1}{n_{(k)}}\, c(X_{(1)} - X_{(k)})$, and these are equal to $1/2$ if and only if $k \in T_1$ where T_1 is the set of all indices of order statistics which satisfy $X_{(k)} = X_{(1)}$ for all $k \in T_1$. Hence, we obtain

$$\widehat{U}(X_{(1)}) = \frac{1}{2a}(\boldsymbol{t}^{(1)})^\top \boldsymbol{m},$$

where $\boldsymbol{t}^{(1)}$ and $\boldsymbol{m}$ are defined in Section 3. Then, this shows the representation for the recursion start

$$R^{\psi}_{(1)} = \frac{1}{2} + \frac{N}{2a}(\boldsymbol{t}^{(1)})^\top \boldsymbol{m}.$$

Now, let us consider $j > 1$ and $X_{(j)} \neq X_{(j-1)}$. Then, the difference $\widehat{U}(X_{(j)}) - \widehat{U}(X_{(j-1)})$ is a sum of non-negative count functions

$$\frac{1}{n_{(k)}}\, c(X_{(j)} - X_{(k)}),$$

where $k \in T_j$ or

$$\frac{1}{n_{(k)}}\left(c(X_{(j)} - X_{(k)}) - c(X_{(j-1)} - X_{(k)})\right)$$

for $k \in T_{j-1}$. In both cases, these simplify to $\frac{1}{2n_{(k)}}$. Thus, the increment from $R^{\psi}_{(j-1)}$ to $R^{\psi}_{(j)}$ is equal to

$$\frac{N}{2a}\left(\boldsymbol{t}^{(j)} + \boldsymbol{t}^{(j-1)}\right)^\top \boldsymbol{m}.$$

But this term is only added if $X_{(j)} \neq X_{(j-1)}$, or equivalently, if $(1 - t_j^{(j-1)}) = 1$. This concludes the proof for the recursive representation of pseudo-ranks.

The Formulas 5 and 7 follow directly from (4) by writing the vector products as sums and splitting up the formula into two parts, that is we calculate first the intermediate pseudo-ranks and then adjust for ties to obtain mid pseudo-ranks.

B. Minimum and maximum pseudo-ranks

Similarly to Formula 4, we can obtain formulas for minimum and maximum pseudo-ranks. For minimum pseudo-ranks $R^{\psi-}_{(i)}$, we only consider the left-continuous empirical distribution functions. This leads to

$$\begin{aligned} R^{\psi-}_{(i)} &= R^{\psi-}_{(i-1)} + (1 - t_i^{(i-1)})\ \frac{N}{a}(\boldsymbol{t}^{(i-1)})^\top \boldsymbol{m}, \\ R^{\psi-}_{(1)} &= 1 \end{aligned}$$

$X_{(k)}$	Group	$R_k^{\psi-}$	$R_k^{\psi+}$	R_k^-	R_k^+
1.00	3	1.00	0.75	1	1
2.00	3	1.75	1.50	2	2
3.00	3	2.50	2.25	3	3
4.00	3	3.25	3.00	4	4
5.00	2	4.00	4.00	5	5
6.00	2	5.00	9.00	6	9
6.00	2	5.00	9.00	6	9
6.00	1	5.00	9.00	6	9
6.00	1	5.00	9.00	6	9

Table 9: Example demonstrating the relation between minimum and maximum (pseudo)-ranks.

for $i \in \{2, \dots, N\}$. For maximum pseudo-ranks $R_{(i)}^{\psi+}$ (i.e., using the right-continuous empirical distribution functions), we obtain

$$R_{(i)}^{\psi+} = R_{(i-1)}^{\psi+} + (1 - t_i^{(i-1)})\ \frac{N}{a}(\boldsymbol{t}^{(i)})^\top \boldsymbol{m},$$
$$R_{(1)}^{\psi+} = \frac{N}{a}(\boldsymbol{t}^{(i)})^\top \boldsymbol{m}$$

for $i \in \{2, \dots, N\}$. The derivation of these formulas is similar to those of mid pseudo-ranks and is therefore omitted.

The names minimum and maximum pseudo-ranks may be a bit misleading. In general, the inequality $R_{(k)}^{\psi-} \leq R_{(k)}^{\psi+}$ is not true. However, it is correct for ranks, namely, $R_{(k)}^- \leq R_{(k)}^+$. Consider the following example in Table 9. In this situation, we have, for example, $R_{(1)}^{\psi-} = 1 > R_{(1)}^{\psi+} = \frac{3}{4}$ and $R_{(4)}^{\psi-} = \frac{13}{4} > R_{(4)}^{\psi+} = 3$, but $R_{(9)}^{\psi-} = 5 < R_{(9)}^{\psi+} = 9$. We still decided to use the same name as for ranks because in both cases the left-continuous or right-continuous empirical distribution functions are used. And for equal group sizes, minimum (maximum) pseudo-ranks and minimum (maximum) ranks are identical.

The following result follows directly by using the relation between ranks and pseudo-ranks form Section 4.3. The minimum and maximum pseudo-ranks satisfy $R_{ik}^{\psi-} > R_{ik}^{\psi+}$ if and only if

$$R_{ik}^{A+} - R_{ik}^{A-} < \tfrac{N^*}{N} - 1, \tag{8}$$

where N^* is the sample size of the amplified data set and R_{ik}^{A+}, R_{ik}^{A-} are the maximum and minimum ranks of X_{ik} based on the amplified data. Note that for an observation X_{ik} which is not tied with any other observation, we obtain $R_{ik}^{A+} - R_{ik}^{A-} = \lambda_i - 1$ as the amplified data set contains $\lambda_i = \mathrm{LCM}(n_1, \dots, n_a)/n_i$ copies of X_{ik}. Then the inequality (8) is true if and only if

$$0 < \sum_{j=1}^{a} n_j(\lambda_j - \lambda_i) = N^* - \lambda_i N \tag{9}$$

$$\iff 0 < a - \frac{1}{n_1}N. \tag{10}$$

In general, we have $R_{ik}^{A+} - R_{ik}^{A-} = \sum_{s=1}^{a} b_s \lambda_s - 1$ where b_s is the number of ties for the observation X_{ik} within group $s, = 1, \ldots, a$. Let $c_s = b_s/n_s$ denote the proportion of tied values for X_{ik} within the sth group and $\sum_{s=1}^{a} c_s = c$. Then (8) or equivalently $R_{ik}^{\psi-} > R_{ik}^{\psi+}$ is true, if and only if

$$0 < a - cN. \tag{11}$$

Note that (10) is a special case of (11) with $c = 1/n_i$ if X_{ik} is not tied with any other observation.

For $X_{(1)}$ to $X_{(4)}$ in Table 9, the amplification factor λ_3 for group 3 is the smallest one ($\lambda_1 = 6$, $\lambda_2 = 4$ and $\lambda_3 = 3$), hence (9) is positive. Therefore, we observe $R_{(k)}^{\psi-} > R_{(k)}^{\psi+}$ for $k = 1, \ldots, 4$. For $X_{(5)}$, the sum in (9) is zero, thus we obtain $R_{(5)}^{\psi-} \leq R_{(5)}^{\psi+}$. For $X_{(k)}$, $k = 6, \ldots, 9$, the condition is $a - cN = 3 - 5/3\ N < 0$, hence $R_{(k)}^{\psi-} \leq R_{(k)}^{\psi+}$.

Affiliation:

Arne C. Bathke, Martin Happ
Department of Mathematics
University of Salzburg
IDA Lab Salzburg
Hellbrunner Strasse 34
5020 Salzburg, Austria
E-mail: arne.bathke@sbg.ac.at, martin.happ@sbg.ac.at

Edgar Brunner
Institut für Medizinische Statistik
Universitätsmedizin Göttingen
Humboldtallee 32
37073 Göttingen, Germany
E-Mail: ebrunne1@gwdg.de

Georg Zimmermann
Paracelsus Medical University
IDA Lab Salzburg
Strubergasse 22
5020 Salzburg, Austria
E-mail: georg.zimmermann@pmu.ac.at

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2020, Volume 95, Code Snippet 1 *Submitted:* 2018-08-06
doi:10.18637/jss.v095.c01 *Accepted:* 2019-06-29