Exact calculations in the Mann-Whitney-Wilcoxon test

Pietro Battiston

This version: July 17, 2015

The *Mann-Whitney-Wilcoxon test*, or *Mann-Whitney U test*, tests non-parametrically the hypothesis that two sets come from the same distribution. Intuitively, this is done by ordering all the elements of the two sets and then studying the ranks. Broadly speaking, the two following steps are executed:

1. a *U statistics* is calculated from the two sets,
2. the resulting value is compared to the theoretical distribution of such U statistics, yielding a probability which can be interpreted as a p-value.

The first task can be accomplished with at least two different methods, both quite straightforward and computationally cheap.

This short note focuses on the execution of the second step. Mann and Whitney (1947) provide in their Equation (1) a recursive definition of the probability of a given U given sizes m and n of the two sets:

$$p(n, m, U) = \frac{n}{n+m} p(n-1, m, U) + \frac{m}{n+m} p(n, m-1, U)$$

which, knowing that

- $p(0, m, 0) = p(n, 0, 0) = 1$,
- $p(0, m, U) = p(n, 0, U) = 1$ whenever $U = 0$,
- $p(n, m, U)$ whenever $U < 0$,

makes it easy to calculate $p(n, m, U)$ for any n, m, U (and hence to derive the desired p-value as $\sum_{u=0}^{U} p(n, m, U)$). However, given the computational cost of this recursive calculation, statistical packages tend to provide an approximation, based on the fact that for m and n large enough, the distribution of $U - \frac{1}{2}(nm+1)$ *differs only a negligible amount from the normal distribution* (as Mann and Whitney, 1947 themselves state, considering the case $m = n = 8$ - they then proceed to showing that the limit distribution of U is normal with m and n both approaching infinity).

As of writing this note,

- the *R* command **wilcox.test()**[1] uses the exact calculation *"if the [total size of the two] samples contain less than 50 finite values"* (or if the argument **exact** is set to TRUE),
- the STATA command *ranksum*[2] always uses the approximation[3], but the add-on package *ranksumex* performs instead the exact calculation *"if the total size of the two samples is <= 25"*.[4],
- the function **stats.mannwhitneyu** from Python's scipy library always uses the approximation, and this is why the documentation states *"use only when the number of observation in each sample is > 20"*;[5] however a set of patches currently under review deprecates it, and introduces the new method **stats.mann_whitney_u**, which does the exact computation when any of the two samples has less than 10 elements (or if the argument "exact" is set to True).

Now, the MWW test is often used with *very* small samples, and in particular has the nice feature (i.e. compared to the "Wilcoxon signed rank test") that the two sets need not to be matched. So one of the two can be significantly smaller than the other. For instance you can ask "if I have one observation with value 0 and 50 observations with values > 0, what is the likelihood of the first observation coming from the same distribution as the others 50?" (and get a non-parametric answer). The problem is that while 50 is a pretty large number for our purposes, 1 (the size of the first set in this example) is definitely not very close to infinity. How reasonable is the normal approximation in this case? Is its accuracy really related to the *sum* of m and n? Concerning the meaning of the "exact" parameter in R and Python, what is the best implementation of the "auto" default value? In order to answer these questions, I ran a simple analysis on the difference between result of the exact algorithm and the approximated one.

The results can be seen in the following figures, for different m, n, both with $U = \frac{q(q+1)}{2}$, where $q = min(3, floor(m/2), floor(n/2))$ ("small U"), and with $U = max\left(\frac{n(n+1)}{2}, mn\right)$ ("large U").

Notice that the "Python" plots do not refer to the implementation soon to be merged in scipy, but rather to my naive implementation of the algorithm described by Mann and Whitney (1947).[6]

Indeed, the error decreases with increasing m, n. However it does not decrease as a function of $m+n$; rather, it has a non-trivial behavior, and for some

[1] `https://stat.ethz.ch/R-manual/R-devel/library/stats/html/wilcox.test.html`

[2] `http://www.stata.com/help.cgi?ranksum`

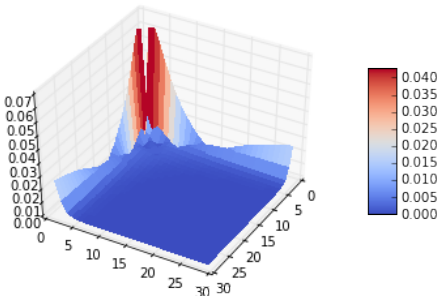
[3] As reported also in `http://www.senresearch.org/stata-and-r-ranksum-test-p-values-differ.html`

[4] `http://www.stata-journal.com/software/sj13-2/st0297/ranksumex.sthlp`

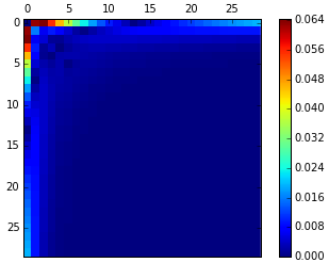
[5] `http://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.stats.mannwhitneyu.html`

[6] The IPython notebook used for the analysis and my Python implementation it uses (based on the source of the R implementation) can be found at `http://pietrobattiston.it/economics/mww`.

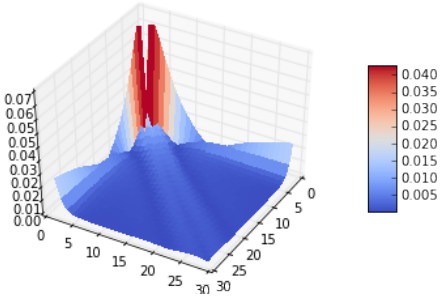
(a) Plot of error, small U



(b) Heat map of error, small U



(c) Plot of error, large U



(d) Heat map of error, large U

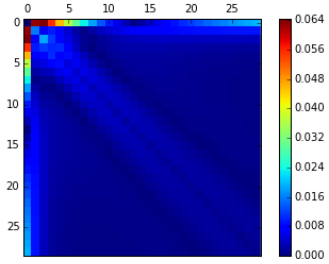
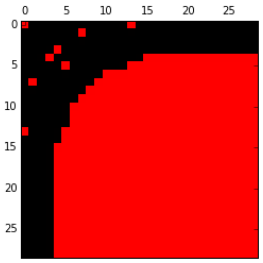
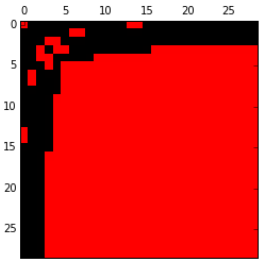


Figure 1: Error of the normal approximation, for varying (m, n).

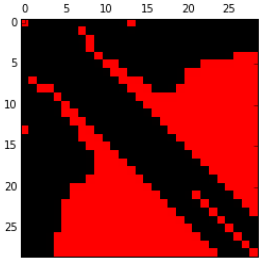
(a) Pairs (m, n) for which the error is above 0.001 - small U



(b) Pairs (m, n) for which the error is above 0.002 - small U



(c) Pairs (m, n) for which the error is above 0.001 - large U



(d) Pairs (m, n) for which the error is above 0.002 - large U

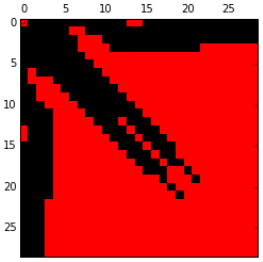
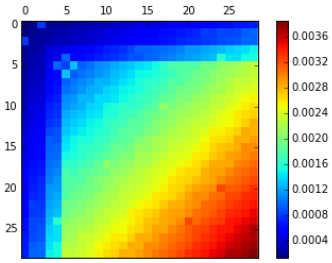
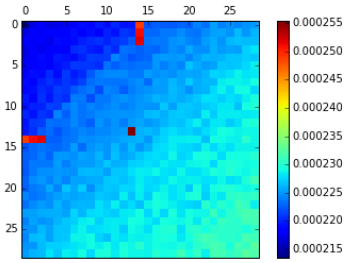


Figure 2: (m, n) pairs for which the error is larger than a given threshold.

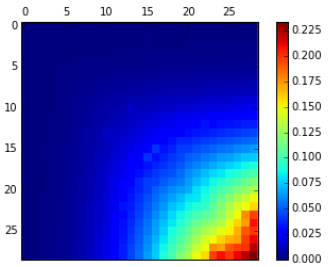
(a) Python reimplementation - small U



(b) R - small U



(c) Python reimplementation - large U



(d) R - large U

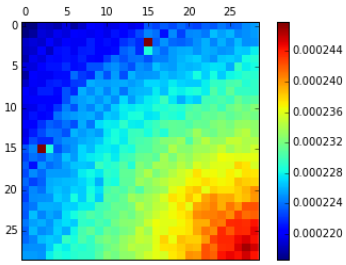


Figure 3: Computation times for varying of (m, n).

small values of m it can actually be *increasing* in n. Figure 2 in particular highlights the fact that while the error of the approximation tends to 0 for m close to n, this does not happen - or happens more slowly - for m or n close to 0.

Resorting to the approximated significance for large values of m or n but small values of the other could be justified if the computational cost of the exacted calculations was exploding in their sum. But as can be seen in Figure 3, this does not happen: the computational cost seems to be rather related to the *minimum* between m and n, (except when U is small - but this is exactly when the computation is overall cheaper). This is coherent with the theoretical complexity of the algorithms used in the simulations, which is $O(n^2m^2)$; Nagarajan and Keich, 2009 discuss a method with lower asymptotic cost, an implementation of which is apparently contained in the *ranksumex* package (Harris et al., 2013).

So in conclusion the safe choice for a statistical package seems to resort to the approximation of the statistics for a given U not if $m + n$ is large enough, but rather if m and n both are large enough - what "large enough" means is then a decision which can be taken looking at the specific computational cost. The new method **mann_whitney_u** in scipy hence seems to make the right choice (although notice that differently from the implementations analyzed in the present note, its calculation of the exact p-value is not based on the algorithm by Mann and Whitney, 1947, but on a different one which could have a computational cost behaving drastically differently for varying m, n and U).

References

Harris, T., J. W. Hardin, et al. (2013). Exact wilcoxon signed-rank and wilcoxon mann–whitney ranksum tests. *Stata Journal 13*(2), 337–343.

Mann, H. B. and D. R. Whitney (1947). On a test of whether one of two random variables is stochastically larger than the other. *The annals of mathematical statistics*, 50–60.

Nagarajan, N. and U. Keich (2009). Reliability and efficiency of algorithms for computing the significance of the mann–whitney test. *Computational Statistics 24*(4), 605–622.