

# Applications of Generating Functions in Nonparametric Tests

**Peter Weiß**

The combination of *Mathematica*'s high algebraic capacity and method of generating functions is becoming an extremely efficient tool in probability theory and statistics. After an introductory example and a short overview on nonparametric methods, we show how generating functions of discrete statistics can be handled using *Mathematica*. Next, we solve two combinatorial problems, which are essential in calculating the generating function of nonparametric test statistics. Finally, these methods are used to produce accurate and extensive tables and plots of the distributions of some of the most popular nonparametric test statistics. Furthermore, we use graphs to illustrate approximations of the distributions of these test statistics by standard distributions.

## ■ An Introductory Example

We start our investigation with an example, which is used in the *Statistica* [1] manual to explain the Kruskal–Wallis analysis of variance with tied observations.

$n = 15$  small children who were randomly assigned to one of  $k = 3$  experimental groups (treatments) of equal size,  $n_1 = n_2 = n_3 = 5$ . Each child was shown a series of pairs of stimuli. Their task was to choose one of those stimuli, and, if it was the “correct” one, they received a reward. The relevant dimensions that the children had to detect to make correct choices were form (group 1), color (group 2), and size (group 3). The dependent variable was the number of trials the children required to detect the dimension that was being rewarded. The following table shows the data set.

Group 1	9	10	11	8	9
Group 2	10	8	15	10	11
Group 3	12	10	12	17	15

We have to test if the distributions of the groups' performances (the numbers of trials required to detect the relevant dimensions) differ significantly from each

other. Because there is no reason to assume that the performances are normally distributed, we can not apply the usual analysis of variance. Following [2], we have to use the Kruskal–Wallis analysis of variance with tied observations (the values 8, 9, 10, 11, 12, and 15 occur several times and are called ties).

Therefore, we determine the different performances, their frequencies, and their midranks

Performance	Frequency	Midrank	Performance	Frequency	Midrank
8	2	1.5	12	2	11.5
9	2	3.5	15	2	13.5
10	4	6.5	17	1	15
11	2	9.5			

and calculate the sum ( $R_1 = 24.5$ ,  $R_2 = 37.5$ ,  $R_3 = 58$ ) of the midranks. We have to reject the null hypothesis (the performances of the three groups are equally distributed) and accept the alternative hypothesis (the performances of these three groups are not equally distributed) if the Kruskal–Wallis statistic with tied observations

$$K = \frac{1}{\Delta} \left[ \frac{12}{n(n+1)} \sum_{i=1}^k \frac{1}{n_i} R_i^2 - 3(n+1) \right]$$

with

$$\Delta = 1 - \frac{1}{n^3 - n} \sum_{j=1}^l (d_j^3 - d_j)$$

is greater than or equal to a certain value,  $\kappa_{1-\alpha}$ . In this formula,  $d_j$  denotes the frequency of the  $j$ -th performance.  $\kappa_{1-\alpha}$  (the so-called  $1-\alpha$  quantile of the Kruskal–Wallis statistic with tied observations) is the greatest real number  $x$  with

$$\Pr(\{K < x\}) < 1 - \alpha.$$

The aim of this article is to show how to calculate the distribution and especially the quantiles of such nonparametric test statistics using *Mathematica*. Applying these methods to this introductory example, we find  $\kappa_{0.95} = 5.65651$ . Thus, because of

$$K = 5.85062 > 5.65651 = \kappa_{0.95},$$

we conclude that the distributions of the performances of the different groups differ significantly (with a level of significance of  $\alpha = 5\%$ ) from each other.

## ■ Nonparametric Tests

Nonparametric methods were developed to be used when the researcher knows nothing about the distribution of the variable of interest (hence the name *nonpara-*

*metric*). Basically, there is at least one nonparametric equivalent for each parametric type of test. When you want to compare the distribution of two independent samples, usually you would use the Student  $t$ -test; one of the most important alternatives for this test is the Wilcoxon rank sum test. If you have more than two independent samples to compare, usually you would use analysis of variance; the nonparametric equivalent to this method is the Kruskal–Wallis analysis of variance.

Nonparametric tests are robust (they need no assumptions about the distribution of the background population), efficient (in early days it was believed that a heavy price in loss of efficiency would have to be paid for robustness, but [2, 3] and several other authors showed clearly that the efficiency is comparable with classical tests using the assumption of normality), and easy to handle using computers.

Nevertheless, they are not widely accepted. One possible reason is that the application of a nonparametric test requires the use of (and the confidence in) a table (see, for example, [4] or [5]) of the distribution of its test statistics.

Generating such tables requires heavy algebraic manipulations and is therefore mostly beyond the scope of introductory textbooks. Published tables are sometimes inaccurate or not extensive enough (especially in the case of ties). Moreover, Mitic [6] pointed out that the entries of some published tables differ, depending on their source. Finally, if the sample size is large, it is possible to approximate most of these distributions by standard distributions, but little is known about the quality of these approximations for small sample sizes.

Thus, I believe, *Mathematica* users are interested in procedures that allow them to generate accurate and extensive tables of the distributions of nonparametric test statistics and plot the distribution functions of these statistics. With these procedures, *Mathematica* users are independent of published tables and can investigate the quality of the approximation by standard distributions.

## ■ Generating Functions

Roughly speaking, a generating function is a polynomial in one or more variables (in expanded form), whose exponents are real numbers and whose coefficients are the numbers we are seeking. Generating functions are widely used in probability theory ([7], [8], or [9]) and combinatorics [10]. In this article we use the word “polynomial” in a nonstandard sense. Usually polynomials have integer exponents only. Because *Mathematica* works well with this kind of “generalized polynomials,” we use them instead.

For instance, if *Stat* is a discrete statistic with possible values  $x_1, x_2, \dots, x_n$ , its generating function is defined as

$$G[\text{Stat}] = \sum_{i=1}^n \Pr(\{\text{Stat} = x_i\}) t^{x_i},$$

where the argument  $t$  plays only an auxiliary role. Thus, if  $Stat$  is a binomial distributed with parameters  $n$  and  $p$ , we have

$$G[Stat] = \sum_{i=0}^n \Pr(\{Stat = i\}) t^i = \sum_{i=0}^n \binom{n}{i} p^i (1-p)^{n-i} t^i = (pt + (1-p))^n.$$

The generating function  $G[Stat]$  of a discrete statistic  $Stat$  contains all the information about its distribution. Given the generating function, it is thus possible to calculate (as well as plot or tabulate)

- the probability density function, PDFG[Stat, x] := Pr({Stat = x});
- the cumulated distribution function, CDFG[Stat, x] := Pr({Stat < x});
- the alpha-quantile, QTLG[Stat,  $\alpha$ ] := max {x | DFG[Stat, x] <  $\alpha$ };
- the  $n$ -th moment, MOMG[Stat, n]; and
- the variance, VARG[Stat]

of  $Stat$  (the letter “G” in the name of these functions indicates that we base our calculations on the generating function  $G[Stat]$  of  $Stat$ ). The following easy and relatively fast procedures are useful in this context:

```
In[25]:= PDFG[Stat_, x_] := Coefficient[G[Stat], t, x];
```

```
In[26]:= CDFG[Stat_, x_] :=
Module[{glist, g, ex, exinf, coef, cum, i = 0},
  glist = Apply[List, G[Stat]];
  g = Length[glist];
  ex = Table[Exponent[glist[[i]], t], {i, 1, g}];
  exinf = Append[ex, Infinity];
  coef = glist /. t -> 1;
  cum = FoldList[Plus, 0, coef];
  Scan[If[#1 >= x, Return[cum[[i + 1]]], i++] &, exinf];
```

```
In[27]:= PlotCDFG[Stat_, opt___] :=
Module[{glist, g, ex, exmod, coef, cum, tbl},
  glist = Apply[List, G[Stat]];
  g = Length[glist];
  ex = Table[Exponent[glist[[i]], t], {i, 1, g}];
  exmod = Prepend[Append[ex, Last[ex] + 1], First[ex] - 1];
  coef = glist /. t -> 1;
  cum = FoldList[Plus, 0, coef];
  tbl = Table[{exmod[[i + j]], cum[[j]]}, {j, 1, g + 1}, {i, 0, 1}];
  Show[Graphics[Line[Flatten[tbl, 1]]], opt];
```

```

In[28]:= TableCDFG[Stat_, xmin_, xmax_] :=
  Module[{glist, g, ex, exinf, coef, cum, imin, imax, i = 0, j = 0},
    glist = Apply[List, G[Stat]];
    g = Length[glist];
    ex = Table[Exponent[glist[[i]], t], {i, 1, g}];
    exinf = Append[ex, Infinity];
    coef = glist /. t → 1;
    cum = FoldList[Plus, 0, coef];
    imin = Scan[If[#1 >= xmin, Return[i + 1], i++] &, ex];
    imax = Scan[If[#1 >= xmax, Return[j], j++] &, exinf];
    TableForm[Table[N[{ex[[i]], cum[[i]]}], {i, imin, imax}],
      TableHeadings → {None, {"x", "CDF[x]"}},
      TableSpacing → {0.5, 5}];

In[29]:= QTLG[Stat_, α_] :=
  Module[{glist, g, ex, coef, cum, i = 0},
    glist = Apply[List, G[Stat]];
    g = Length[glist];
    ex = Table[Exponent[glist[[i]], t], {i, 1, g}];
    coef = glist /. t → 1;
    cum = FoldList[Plus, 0, coef];
    Scan[If[#1 >= α, Return[ex[[i]], i++] &, cum]];

In[30]:= pol[Stat_, 1] := D[G[Stat], t];
  pol[Stat_, n_] := D[pol[Stat, n - 1] t, t];
  MOMG[Stat_, n_] := pol[Stat, n] /. t → 1;
  VARG[Stat_] := MOMG[Stat, 2] - MOMG[Stat, 1]^2;

```

Now we give an example of an application of generating functions in combinatorics. For instance, if we divide  $n$  numbered balls randomly among  $k$  urns and denote by  $A\{n_1, n_2, \dots, n_k\}$  the number of cases that  $n_1$  balls come into urn  $U_1, \dots, n_k$  balls come into urn  $U_k$ , then the generating function of these numbers is defined as

$$\begin{aligned}
 a[n, k] &= \sum_{\substack{n_1, \dots, n_k=1 \\ n_1 + \dots + n_k = n}}^n A\{n_1, n_2, \dots, n_k\} x[1]^{n_1} \dots x[k]^{n_k} = \\
 &= \sum_{\substack{n_1, \dots, n_k=1 \\ n_1 + \dots + n_k = n}}^n \frac{n!}{n_1! \dots n_k!} x[1]^{n_1} \dots x[k]^{n_k} = \left( \sum_{i=1}^k x[i] \right)^n,
 \end{aligned}$$

where the arguments  $x[i]$  again play only an auxiliary role.

## ■ Two Combinatorial Problems

The crucial point in nonparametric test theory is that all possible arrangements of the ranks of the observed values are equally likely. To calculate, for example, the distribution density  $\Pr(\{Stat = x\})$  of a statistic  $Stat$ , which is based on ranks, it is therefore only necessary to obtain the number of cases fulfilling the condi-

tion  $Stat = x$ . The two following combinatorial problems give essential ideas in doing this.

### □ Problem 1

Suppose we have  $n$  balls, which are numbered  $1, 2, \dots, n$ . In how many different ways

$$P[\{n_1, \dots, n_k\}, \{r_1, \dots, r_k\}]$$

is it possible to divide these  $n$  balls among  $k$  urns, such that

- the  $i$ -th urn  $U_i$  contains  $n_i$  balls and
- the sum of the numbers of these  $n_i$  balls in urn  $U_i$  is  $r_i$

with  $n = n_1 + \dots + n_k$  and  $r = r_1 + \dots + r_k = \frac{1}{2} n(n+1)$ ?

For small  $n$  and  $k$  we can calculate this number by counting the relevant partitions. For example, there are the following 15 partitions of  $n = 6$  balls into  $k = 2$  urns with  $n_1 = 4$  balls in urn  $U_1$  and  $n_2 = 2$  balls in urn  $U_2$ :

$U_1$	$r_1$	$U_2$	$r_2$	$U_1$	$r_1$	$U_2$	$r_2$	$U_1$	$r_1$	$U_2$	$r_2$
{1, 2, 3, 4}	10	{5, 6}	11	{1, 2, 3, 6}	12	{4, 5}	9	{1, 3, 5, 6}	15	{2, 4}	6
{1, 2, 3, 5}	11	{4, 6}	10	{1, 2, 4, 6}	13	{3, 5}	8	{2, 3, 5, 6}	16	{1, 4}	5
{1, 2, 4, 5}	12	{3, 6}	9	{1, 3, 4, 6}	14	{2, 5}	7	{1, 4, 5, 6}	16	{2, 3}	5
{1, 3, 4, 5}	13	{2, 6}	8	{2, 3, 4, 6}	15	{1, 5}	6	{2, 4, 5, 6}	17	{1, 3}	4
{2, 3, 4, 5}	14	{1, 6}	7	{1, 2, 5, 6}	14	{3, 4}	7	{3, 4, 5, 6}	18	{1, 2}	3

Thus,  $P[\{4, 2\}, \{12, 9\}] = 2$  and  $P[\{4, 2\}, \{14, 7\}] = 3$ . If  $n$  and  $k$  are not as small as in this example, this method fails because there are too many partitions of  $n$  balls into  $k$  urns with  $n_i$  balls in urn  $U_i$ .

### □ Problem 2

Suppose we have  $n$  balls, which are marked by real numbers in the following way:  $d_1$  of these balls have mark  $m_1$ ,  $d_2$  have mark  $m_2$ , ..., and  $d_l$  have mark  $m_l$  (with  $n = d_1 + \dots + d_l$ ). In how many different ways

$$Q[\{d_1, \dots, d_l\}, \{m_1, \dots, m_l\}, \{n_1, \dots, n_k\}, \{r_1, \dots, r_k\}]$$

is it possible to divide these  $n$  balls among  $k$  urns, such that

- the  $i$ -th urn  $U_i$  contains  $n_i$  balls and
- the sum of the marks of these  $n_i$  balls is  $r_i$

with  $n = n_1 + \dots + n_k = d_1 + \dots + d_l$  and  $r = r_1 + \dots + r_k = d_1 m_1 + \dots + d_l m_l$ ?

This problem is obviously a generalization of our first problem, since for  $l = n$  we have

$$Q[\{1, \dots, 1\}, \{1, \dots, n\}, \{n_1, \dots, n_k\}, \{r_1, \dots, r_k\}] = P[\{n_1, \dots, n_k\}, \{r_1, \dots, r_k\}].$$

For small  $n$  and  $k$  it is again easy to calculate  $Q[\text{dlist}, \text{mlist}, \text{nlist}, \text{rlist}]$  by counting the relevant partitions. If we mark  $d_1 = 3$  of the  $n = 6$  balls with the mark  $m_1 = 1$  (let us assume that these are the balls with the numbers 1, 2, 3) and the remaining  $d_2 = 3$  balls with the mark  $m_2 = 2$ , and if we put again  $n_1 = 4$  of these balls in urn  $U_1$  and  $n_2 = 2$  of these balls in urn  $U_2$ , we obtain (the same) 15 different partitions of our marked balls. (In the following table we show only their marks.)

$U_1$	$r_1$	$U_2$	$r_2$	$U_1$	$r_1$	$U_2$	$r_2$	$U_1$	$r_1$	$U_2$	$r_2$
{1, 1, 1, 2}	5	{2, 2}	4	{1, 1, 1, 2}	5	{2, 2}	4	{1, 1, 2, 2}	6	{1, 2}	3
{1, 1, 1, 2}	5	{2, 2}	4	{1, 1, 2, 2}	6	{1, 2}	3	{1, 1, 2, 2}	6	{1, 2}	3
{1, 1, 2, 2}	6	{1, 2}	3	{1, 1, 2, 2}	6	{1, 2}	3	{1, 2, 2, 2}	7	{1, 1}	2
{1, 1, 2, 2}	6	{1, 2}	3	{1, 1, 2, 2}	6	{1, 2}	3	{1, 2, 2, 2}	7	{1, 1}	2
{1, 1, 2, 2}	6	{1, 2}	3	{1, 1, 2, 2}	6	{1, 2}	3	{1, 2, 2, 2}	7	{1, 1}	2

Thus  $Q[\{3, 3\}, \{1, 2\}, \{4, 2\}, \{6, 3\}] = 9$  and  $Q[\{3, 3\}, \{1, 2\}, \{4, 2\}, \{7, 2\}] = 3$ . If  $n$  and  $k$  are not as small as in this example, this method fails again.

□ **A Simple Method**

The aim of this section is to find the generating functions for the numbers  $P[\text{nlist}, \text{rlist}]$  and  $Q[\text{dlist}, \text{mlist}, \text{nlist}, \text{rlist}]$  with  $\text{nlist} := \{n_1, \dots, n_k\}$ ,  $\text{rlist} := \{r_1, \dots, r_k\}$ ,  $\text{dlist} := \{d_1, \dots, d_l\}$ , and  $\text{mlist} := \{m_1, \dots, m_l\}$ .

With some experience it is easy to see that our generating functions must have two sorts of variables: for each urn  $U_i$  one variable  $x[i]$  governing the number of balls in this urn and one variable  $y[i]$  governing the sum of the numbers (marks) of the balls in this urn. With this remark in mind, the generating functions for the numbers  $P[\text{nlist}, \text{rlist}]$  and  $Q[\text{dlist}, \text{mlist}, \text{nlist}, \text{rlist}]$  are obviously the polynomials

$$p[n, k] = \prod_{j=1}^n \left( \sum_{i=1}^k x[i] y[i]^j \right),$$

$$q[\text{dlist}, \text{mlist}, k] = \prod_{j=1}^l \left( \sum_{i=1}^k x[i] y[i]^{m_j} \right)^{d_j}.$$

To be more precise, the numbers  $P[nlist, rlist]$  and  $Q[dlist, mlist, nlist, rlist]$  are the coefficients of

$$\prod_{i=1}^k x[i]^{n_i} y[i]^{r_i}$$

of the polynomials  $p[n, k]$  and  $q[dlist, mlist, k]$ .

In *Mathematica*, we define these two polynomials  $p[n, k]$  and  $q[dlist, mlist, k]$  recursively:

```
In[34]:= xlist[k_] := Table[x[i], {i, 1, k}];
        ylist[k_] := Table[y[i], {i, 1, k}];

In[36]:= p[1, k_] := xlist[k].ylist[k];
        p[n_, k_] := p[n, k] = p[n-1, k] xlist[k].ylist[k]^n

In[38]:= q[{d_}, {m_}, k_] := (xlist[k].ylist[k]^m)^d;
        q[dlist_, mlist_, k_] := q[dlist, mlist, k] =
        q[{First[dlist]}, {First[mlist]}, k] q[Rest[dlist], Rest[mlist], k]

In[40]:= Expand[p[3, 2]]

Out[40]= x[1]^3 y[1]^6 + x[1]^2 x[2] y[1]^5 y[2] + x[1]^2 x[2] y[1]^4 y[2]^2 +
        x[1]^2 x[2] y[1]^3 y[2]^3 + x[1] x[2]^2 y[1]^3 y[2]^3 +
        x[1] x[2]^2 y[1]^2 y[2]^4 + x[1] x[2]^2 y[1] y[2]^5 + x[2]^3 y[2]^6

In[41]:= Expand[q[{2, 2}, {1, 2}, 2]]

Out[41]= x[1]^4 y[1]^6 + 2 x[1]^3 x[2] y[1]^5 y[2] +
        2 x[1]^3 x[2] y[1]^4 y[2]^2 + x[1]^2 x[2]^2 y[1]^4 y[2]^2 +
        4 x[1]^2 x[2]^2 y[1]^3 y[2]^3 + x[1]^2 x[2]^2 y[1]^2 y[2]^4 +
        2 x[1] x[2]^3 y[1]^2 y[2]^4 + 2 x[1] x[2]^3 y[1] y[2]^5 + x[2]^4 y[2]^6
```

Now we get the numbers  $P[nlist, rlist]$  and  $Q[dlist, mlist, nlist, rlist]$  by selecting the coefficients of the polynomials  $p[n, k]$  and  $q[dlist, mlist, k]$ :

```
P[nlist_, rlist_] :=
Module[{n, r, k},
  n = Apply[Plus, nlist];
  r = Apply[Plus, rlist];
  k = Length[nlist];
  If[r == n (n+1) / 2,
    Fold[Coefficient, Expand[p[n, k]],
      Union[xlist[k]^nlist, ylist[k]^rlist]],
    Print["r # n(n+1)/2"]]]
```

```

Q[dlist_, mlist_, nlist_, rlist_] :=
Module[{d, n, r, k},
  d = Apply[Plus, dlist];
  n = Apply[Plus, nlist];
  r = Apply[Plus, rlist];
  k = Length[nlist];
  If[n == d && r == dlist.mlist,
    Fold[Coefficient, Expand[q[dlist, mlist, k]],
      Union[xlist[k]^nlist, ylist[k]^rlist]],
    Print["n ≠ d1+...+d1 or r ≠ d1 m1+...+d1 m1"]]

```

Using these functions, we solve the combinatorial problems posed at the beginning of this section and a problem, which can not be solved by hand.

```
In[44]:= P[{4, 2}, {14, 7}]
```

```
Out[44]= 3
```

```
In[45]:= Q[{3, 3}, {1, 2}, {4, 2}, {6, 3}]
```

```
Out[45]= 9
```

```

Timing[
Q[{3, 4, 5, 4}, {1, 5, 2, 3}, {5, 5, 4, 2}, {15, 12, 11, 7}]]
{24.81 Second, 109440}

```

## □ A More Sophisticated Method

Unfortunately this easy method wastes time and memory (for instance, the last calculation needs more than 100 MB RAM). This disadvantage is because the number of terms of the polynomials  $p[n, k]$  and  $q[dlist, mlist, k]$  are of order  $k^n$ , which is a huge number even if  $n$  and  $k$  are not very large. We overcome this difficulty by fixing  $nlist = \{n_1, \dots, n_k\}$ . In this case the number of terms of the generating functions  $p[nlist]$  and  $q[dlist, mlist, nlist]$  of  $P[nlist, rlist]$  and  $Q[dlist, mlist, nlist, rlist]$  are only of order  $\text{Multinomial}[n_1, \dots, n_k]$ . This is also a large number, but an essentially smaller one than  $k^n$ .

It is obvious that these new generating functions  $p[nlist]$  and  $q[dlist, mlist, nlist]$  are the coefficients of

$$\prod_{i=1}^k x[i]^{n_i}$$

of our old polynomials  $p[n, k]$  and  $q[dlist, mlist, k]$ . Of course we do not use this fact, since it would not save any memory. What we do is calculate these new generating functions  $p[nlist]$  and  $q[dlist, mlist, nlist]$  recursively using the obvious formulas

$$p[\{n_1, \dots, n_k\}] = \sum_{i=1}^k y[i]^n p[\{n_1, \dots, n_i - 1, \dots, n_k\}]$$

$$q[\{d_1, \dots, d_l\}, \{m_1, \dots, m_l\}, \{n_1, \dots, n_k\}] =$$

$$\sum_{i=1}^k y[i]^{m_i} q[\{d_1, \dots, d_i - 1\}, \{m_1, \dots, m_l\}, \{n_1, \dots, n_i - 1, \dots, n_k\}]$$

and some self-evident boundary conditions, if some of the  $n_i$ 's are zero or  $d_l$  is zero.

```

In[47]:= p[{n1_}] := y[1]^(n1 (n1 + 1) / 2);
p[nlist_] := p[nlist] =
Module[{n, k, id, nlistnew, pos, sub},
  n = Apply[Plus, nlist];
  k = Length[nlist];
  id = IdentityMatrix[k];
  nlistnew = Delete[nlist, Position[nlist, 0]];
  pos = Flatten[Position[nlist, j_ /; j > 0]];
  sub = Table[y[i] -> y[pos[[i]]], {i, 1, Length[pos]}];
  If[FreeQ[nlist, 0],
    Expand[Sum[y[i]^n p[nlist - id[[i]]], {i, 1, k}]],
    p[nlistnew] /. sub]]

q[{d_}, {m_}, nlist_] := Apply[Multinomial, nlist]
  Apply[Times, ylist[Length[nlist]]^(m nlist)];
q[dlist_, mlist_, nlist_] := q[dlist, mlist, nlist] =
Module[{d, n, k, id, dlistnew, nlistnew, pos, sub},
  d = Apply[Plus, dlist];
  n = Apply[Plus, nlist];
  k = Length[nlist];
  id = IdentityMatrix[k];
  dlistnew = Append[Drop[dlist, -1], Last[dlist] - 1];
  nlistnew = Delete[nlist, Position[nlist, 0]];
  pos = Flatten[Position[nlist, j_ /; j > 0]];
  sub = Table[y[i] -> y[pos[[i]]], {i, 1, Length[pos]}];
  Which[d != n, Print["n != d1+...+dl"]; Abort[],
    ! (FreeQ[nlist, 0]), q[dlist, mlist, nlistnew] /. sub,
    ! (FreeQ[dlist, 0]), q[Drop[dlist, -1], Drop[mlist, -1], nlist],
    FreeQ[nlist, 0] && FreeQ[dlist, 0],
    Expand[Sum[y[i]^Last[mlist]
      q[dlistnew, mlist, nlist - id[[i]]], {i, 1, k}]]]]

In[51]:= p[{4, 2}]
Out[51]= y[1]^18 y[2]^3 + y[1]^17 y[2]^4 + 2 y[1]^16 y[2]^5 + 2 y[1]^15 y[2]^6 + 3 y[1]^14 y[2]^7 +
  2 y[1]^13 y[2]^8 + 2 y[1]^12 y[2]^9 + y[1]^11 y[2]^10 + y[1]^10 y[2]^11

```

```
In[52]:= q[{2, 1, 1}, {1, 3, 2}, {1, 2, 1}]
```

```
Out[52]= 2 y[1]^3 y[2]^3 y[3] + 2 y[1]^2 y[2]^4 y[3] +
          2 y[1] y[2]^5 y[3] + y[1]^3 y[2]^2 y[3]^2 +
          2 y[1] y[2]^4 y[3]^2 + y[1]^2 y[2]^2 y[3]^3 + 2 y[1] y[2]^3 y[3]^3
```

Now we get  $P[\text{nlist}, \text{rlist}]$  and  $Q[\text{dlist}, \text{mlist}, \text{nlist}, \text{rlist}]$  by selecting the coefficients of

$$\prod_{i=1}^k y[i]^{r_i}$$

of the polynomials  $p[\text{nlist}]$  and  $q[\text{dlist}, \text{mlist}, \text{nlist}]$ :

```
P[nlist_, rlist_] :=
Module[{n, r, k},
  n = Apply[Plus, nlist];
  r = Apply[Plus, rlist];
  k = Length[nlist];
  If[r == n (n + 1) / 2,
    Fold[Coefficient, p[nlist], ylist[k]^rlist],
    Print["r ≠ n(n+1)/2"]]

Q[dlist_, mlist_, nlist_, rlist_] :=
Module[{d, n, r, k},
  d = Apply[Plus, dlist];
  n = Apply[Plus, nlist];
  r = Apply[Plus, rlist];
  k = Length[nlist];
  If[d == n && r == dlist.mlist,
    Fold[Coefficient, q[dlist, mlist, nlist], ylist[k]^rlist],
    Print["n ≠ d1+...+d1 or r ≠ d1 m1+...+d1 m1"]]]
```

The following examples show the efficiency of this method:

```
In[55]:= P[{3, 4, 5}, {15, 25, 38}]
```

```
Out[55]= 91
```

```
In[56]:= Q[{2, 6, 4}, {1, 2, 4}, {3, 4, 5}, {6, 10, 14}]
```

```
Out[56]= 440
```

```
Timing[
Q[{3, 4, 5, 4}, {1, 5, 2, 3}, {5, 5, 4, 2}, {15, 12, 11, 7}]]
{3.5 Second, 109440}
```

## ■ The Wilcoxon Rank Sum Test

Suppose two independent samples  $X_1, \dots, X_{n_1}$  and  $Y_1, \dots, Y_{n_2}$  of sizes  $n_1$  and  $n_2$  are drawn from two continuous (not necessarily normal) populations. We wish to test the null hypothesis  $H_0$  (these two populations are identically distributed)

against the alternative hypothesis  $H_1$  (these two populations are not identically distributed).

The Wilcoxon rank sum test (which is closely related to the Mann–Whitney U test) is one of the best known and easiest to use tests in this situation (see, for example, [2]). It rejects the null hypothesis  $H_0$  if the sum of the ranks of the  $X_i$ 's in the combined ordered arrangement of the two samples is either too large or too small. This statistic is called Wilcoxon rank sum, which we denote by  $\text{WilcoxonRS}[n_1, n_2]$ .

## □ No Ties

Let us first assume that there are no ties, that is, all  $n = n_1 + n_2$  observations are different from each other. Calculating the generating function  $G[\text{WilcoxonRS}[n_1, n_2]]$  of the null distribution of the Wilcoxon rank sum, it is important to mention that for all

$$0 \leq k_1 \leq \frac{1}{2} n(n+1) \text{ and } k_2 = \frac{1}{2} n(n+1) - k_1,$$

we have

$$\Pr(\{\text{WilcoxonRS}[n_1, n_2] = k_1\}) = \frac{P[\{n_1, n_2\}, \{k_1, k_2\}]}{\text{Binomial}[n, n_1]}.$$

Thus we get

```
In[58]:= G[WilcoxonRS[n1_, n2_]] :=
  Apart[(p[{n1, n2}] /. {y[1] -> t, y[2] -> 1}) / Binomial[n1 + n2, n1]]
```

```
In[59]:= G[WilcoxonRS[4, 3]]
```

$$\text{Out[59]} = \frac{t^{10}}{35} + \frac{t^{11}}{35} + \frac{2t^{12}}{35} + \frac{3t^{13}}{35} + \frac{4t^{14}}{35} + \frac{4t^{15}}{35} + \frac{t^{16}}{7} + \frac{4t^{17}}{35} + \frac{4t^{18}}{35} + \frac{3t^{19}}{35} + \frac{2t^{20}}{35} + \frac{t^{21}}{35} + \frac{t^{22}}{35}$$

Together with our procedures of the third section, we can use this generating function to calculate the probability density function; to calculate, tabulate, and plot the cumulated distribution function; and to calculate alpha-quantiles of the null distribution of the Wilcoxon rank sum. For example,

```
In[60]:= CDFG[WilcoxonRS[4, 3], 13]
```

$$\text{Out[60]} = \frac{4}{35}$$

```
In[61]:= QTLG[WilcoxonRS[5, 5], 0.90]
```

$$\text{Out[61]} = 34$$

For large values of  $n_1$  and  $n_2$ , it is well known (see, for example, [2]) that the null distribution of the test statistic  $\text{WilcoxonRS}[n_1, n_2]$  can be approximated by an appropriate normal distribution. We can now investigate the quality of this

approximation in the case of relatively small values of  $n_1$  and  $n_2$  graphically (Figure 1).

```
In[62]:= Needs["Statistics`NormalDistribution`"]
In[63]:= n1 = 6; n2 = 3;
mu = MOMG[WilcoxonRS[n1, n2], 1];
sigma = Sqrt[VARG[WilcoxonRS[n1, n2]]];
plot1 = PlotCDFG[WilcoxonRS[n1, n2], DisplayFunction -> Identity];
plot2 = Plot[CDF[NormalDistribution[mu, sigma], x],
  {x, mu - Floor[3 sigma], mu + Ceiling[3 sigma]},
  DisplayFunction -> Identity];
fig1 = Show[{plot1, plot2}, PlotRange -> All,
  DisplayFunction -> $DisplayFunction, Axes -> True]
```

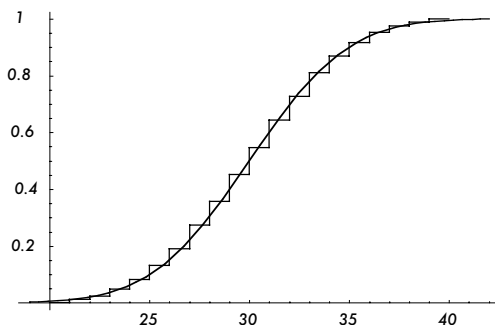


Figure 1.

Furthermore, it is possible to generate accurate tables of alpha-quantiles of the null distribution of this test statistic.

```
In[69]:= n1max = 12; n2max = 12; alpha = 0.90;
TableForm[
  Table[QTLG[WilcoxonRS[i, j], alpha], {i, 2, n1max}, {j, 1, n2max}],
  TableHeadings -> {Prepend[Table[i, {i, 3, n1max}], "n1=2"],
  Prepend[Table[j, {j, 2, n2max}], "n2=1"]},
  TableAlignments -> Center,
  TableSpacing -> {0.5, 1}]
```

Out[70]//TableForm=

	n2=1	2	3	4	5	6	7	8	9	10	11	12
n1=2	5	7	8	10	11	13	15	16	18	19	21	22
3	9	11	13	16	18	20	22	24	27	29	31	33
4	14	17	20	22	25	28	31	34	36	39	42	45
5	20	23	27	30	34	37	41	44	47	51	54	57
6	27	31	35	39	43	47	51	55	59	63	67	71
7	35	40	44	49	54	58	63	67	72	76	81	85
8	44	49	54	60	65	70	75	80	85	91	96	101
9	53	60	66	71	77	83	89	94	100	106	112	117
10	64	71	78	84	91	97	103	110	116	122	128	135
11	76	84	91	98	105	112	119	126	133	139	146	153
12	89	97	105	113	120	128	135	143	150	158	165	172

## □ Ties

If ties occur, we assign to tied observations the same midrank, which is the average of the ranks of these observations. Suppose we have the seven observations 1.3, 1.7, 1.7, 1.7, 1.9, 1.9, and 2.2, then the midranks of these observations are 1, 3, 3, 3, 5.5, 5.5, and 7.

If  $d_1$  observations have midrank  $m_1$  and ...  $d_j$  observations have midrank  $m_j$  and if these midranks are ordered increasingly, we can calculate these midranks using the formula

$$m_j = \sum_{s=1}^{j-1} d_s + \frac{d_j + 1}{2}.$$

Thus, we define

```
In[71]:= m[dlist_] := FoldList[Plus, 0, Drop[dlist, -1]] + (dlist + 1) / 2
```

Calculating the generating function  $G[\text{WilcoxonRSTies}[\text{dlist}, n_1, n_2]]$  of the null distribution of the Wilcoxon rank sum (which is now the sum of the midranks of the  $X_i$ 's in the combined ordered arrangement of the two samples), it is important to mention that for all

$$0 \leq k_1 \leq \frac{1}{2} n(n+1) \text{ and } k_2 = \frac{1}{2} n(n+1) - k_1$$

we have

$$\Pr(\{\text{WilcoxonRSTies}[\text{dlist}, n_1, n_2] = k_1\}) = \frac{Q[\text{dlist}, \{n_1, n_2\}, \{k_1, k_2\}]}{\text{Binomial}[n, n_1]}.$$

Thus, we get

```
G[WilcoxonRSTies[dlist_, n1_, n2_]] :=
  If[Apply[Plus, dlist] == n1 + n2,
    Apart[(q[dlist, m[dlist], {n1, n2}] /. {y[1] -> t, y[2] -> 1}) /
      Binomial[n1 + n2, n1]],
    Print["n1+n2 ≠ d1+...+dl"]; Abort[]]
```

```
In[73]:= G[WilcoxonRSTies[{2, 3, 4}, 5, 4]]
```

$$\text{Out[73]} = \frac{t^{15}}{126} + \frac{2 t^{37/2}}{21} + \frac{4 t^{21}}{63} + \frac{t^{22}}{7} + \frac{2 t^{49/2}}{7} + \frac{2 t^{51/2}}{63} + \frac{t^{27}}{21} + \frac{4 t^{28}}{21} + \frac{2 t^{61/2}}{21} + \frac{t^{63/2}}{63} + \frac{t^{34}}{42}$$

Again we can use this generating function to calculate the probability density function; to calculate, tabulate, and plot the cumulated distribution function; and to calculate alpha-quantiles of the null distribution of the Wilcoxon rank sum. For example,

```
In[74]:= TableCDFG[WilcoxonRSTies[{{3, 3, 4}, 5, 5], 30, 40]
```

```
Out[74]//TableForm=
```

x	CDF [x]
32.	0.761905
32.5	0.785714
35.5	0.928571
36.	0.97619
39.	0.988095

In [2] it is shown that for large values of  $n_1$  and  $n_2$  the null distribution of the test statistics  $\text{WilcoxonRSTies}[\text{dlist}, n_1, n_2]$  can be approximated by an appropriate normal distribution. Again we investigate the quality of this approximation in the case of relatively small values of  $n_1$  and  $n_2$  graphically (Figure 2).

```
In[75]:= dlist = {1, 2, 3, 2, 1}; n1 = 5; n2 = 4;
mu = MOMG[WilcoxonRSTies[dlist, n1, n2], 1];
sigma = Sqrt[VARG[WilcoxonRSTies[dlist, n1, n2]]];
plot1 = PlotCDFG[
  WilcoxonRSTies[dlist, n1, n2], DisplayFunction -> Identity];
plot2 = Plot[CDF[NormalDistribution[mu, sigma], x],
  {x, mu - Floor[3 sigma], mu + Ceiling[3 sigma]},
  DisplayFunction -> Identity];
fig2 = Show[{plot1, plot2}, PlotRange -> All,
  DisplayFunction -> $DisplayFunction, Axes -> True]
```

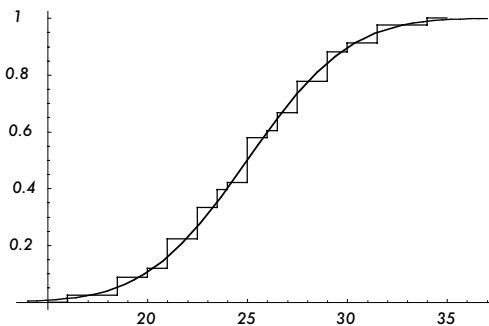


Figure 2.

## ■ The Kruskal–Wallis Analysis of Variance

The Kruskal–Wallis analysis of variance is a generalization of the Wilcoxon rank sum test. These two tests are related in the same way as the well-known analysis of variance and the two-sample Student  $t$ -test.

Suppose that  $k$  independent samples  $X_{11}, \dots, X_{1n_1}; \dots; X_{k1}, \dots, X_{kn_k}$  of sizes  $n_1, \dots, n_k$  are drawn from  $k$  continuous (not necessarily normal) populations. We want to test the null hypothesis  $H_0$  (these populations are identically distributed) against the alternative hypothesis  $H_1$  (these populations are not identically distributed).

## □ No Ties

Let us assume first that there are no ties. In that case, we calculate the sum  $R_1, \dots, R_k$  of the ranks of the  $X_1$ 's,  $\dots$ ,  $X_k$ 's in the combined ordered arrangement of these  $k$  samples and reject the null hypotheses  $H_0$ , if the Kruskal–Wallis statistics

$$\text{KruskalWallis}[\text{nlist}] = \frac{12}{n(n+1)} \sum_{i=1}^k \frac{1}{n_i} R_i^2 - 3(n+1)$$

with  $n = n_1 + \dots + n_k$  is too large.

Calculating the generating function  $G[\text{KruskalWallis}[\text{nlist}]]$  of the null distribution of the Kruskal–Wallis test statistics, it is important to mention that

$$\Pr(\{R_1 = r_1\} \cap \dots \cap \{R_k = r_k\}) = \frac{P[\text{nlist}, \text{rlist}]}{\text{Multinomial}[n_1, \dots, n_k]}.$$

Thus, we get the generating function  $G[\text{KruskalWallis}[\text{nlist}]]$  by applying the substitution

$$s1 = \left\{ y[i]^r \rightarrow t^{\wedge} \left( \frac{r^2}{n_i} \right) \mid i = 1, \dots, k \right\}$$

to

$$\frac{p[\text{nlist}]}{\text{Multinomial}[n_1, \dots, n_k]}$$

and the substitution

$$s2 = t^x \rightarrow t^{\wedge} \left( \frac{12x}{n(n+1)} - 3(n+1) \right)$$

to the result of the first substitution.

```
In[81]:= G[KruskalWallis[nlist_]] :=
Module[{n, k, a, b, sub1, sub2},
n = Apply[Plus, nlist];
k = Length[nlist];
a = 12 / (n (n + 1));
b = 3 (n + 1);
s1 = Table[y[i]^r_ -> t^(r^2/nlist[[i]]), {i, 1, k}];
s2 = t^x_ -> t^(a x - b);
Apart[(p[nlist] /. s1 /. s2) / Apply[Multinomial, nlist]]]
```

```
In[82]:= G[KruskalWallis[{2, 3, 2}]]
```

$$\text{Out[82]} = \frac{1}{35} + \frac{8t^{5/28}}{105} + \frac{4t^{3/14}}{105} + \frac{2\sqrt{t}}{35} + \frac{2t^{17/28}}{35} + \frac{2t^{5/7}}{35} + \frac{t^{6/7}}{35} + \frac{4t^{33/28}}{105} + \frac{2t^{19/14}}{35} + \frac{4t^{41/28}}{105} + \frac{2t^{45/28}}{35} + \frac{t^{27/14}}{35} + \frac{2t^2}{105} + \frac{4t^{31/14}}{105} + \frac{2t^{17/7}}{105} + \frac{4t^{69/28}}{105} + \frac{4t^{11/4}}{105} + \frac{2t^{20/7}}{105} + \frac{2t^{89/28}}{105} + \frac{t^{24/7}}{105} + \frac{2t^{101/28}}{105} + \frac{4t^{15/4}}{105} + \frac{8t^{55/14}}{105} + \frac{4t^{125/28}}{105} + \frac{2t^{9/2}}{105} + \frac{2t^{33/7}}{105} + \frac{t^{75/14}}{35}$$

Therefore, we have, for example,

```
In[83]:= TableCDFG[KruskalWallis[{2, 3, 3}], 4.3, 6.5]
```

```
Out[83]//TableForm=
```

x	CDF[x]
4.55556	0.9
4.69444	0.907143
5.	0.925
5.13889	0.939286
5.36111	0.967857
5.55556	0.975
6.25	0.989286

It is well known (we refer again to Lehmann [2]) that for large values of  $n_1, \dots, n_k$  it is possible to approximate the null distribution of the test statistics  $\text{KruskalWallis}[nlist]$  by the chi-square distribution with  $k - 1$  degrees of freedom. We are now in a position to study the quality of this approximation in the case of relatively small values of  $n_1, \dots, n_k$ . It is important to remark that this approximation is not very good in the most important region between the 0.95 and the 0.99 quantile (Figure 3).

```
In[84]:= nlist = {2, 3, 4};
k = Length[nlist];
plot1 = PlotCDFG[KruskalWallis[nlist], DisplayFunction -> Identity];
plot2 = Plot[CDF[ChiSquareDistribution[k - 1], x],
{x, 0, 8}, DisplayFunction -> Identity];
fig3 = Show[{plot1, plot2}, PlotRange -> All,
DisplayFunction -> $DisplayFunction, Axes -> True]
```

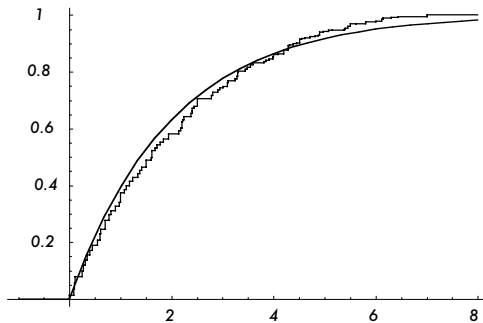


Figure 3.

## □ Ties

If ties occur, we assign to tied observations the same midrank and calculate the sum  $R_1, \dots, R_k$  of the midranks of the  $X_{1\cdot}$ 's,  $\dots$ ,  $X_{k\cdot}$ 's in the combined ordered arrangement of these  $k$  samples. We reject the null hypotheses  $H_0$ , if the modified Kruskal–Wallis statistics

$$\text{KruskalWallisTies}[\text{dlist}, \text{nlist}] = \frac{1}{\Delta} \left[ \frac{12}{n(n+1)} \sum_{i=1}^k \frac{1}{n_i} R_i^2 - 3(n+1) \right]$$

with

$$\Delta = 1 - \frac{1}{n^3 - n} \sum_{j=1}^l (d_j^3 - d_j)$$

and  $n = n_1 + \dots + n_k$  is too large.

Calculating the generating function  $G[\text{KruskalWallisTies}[\text{dlist}, \text{nlist}]]$  of the null distribution of this test statistics, it is again essential to mention that

$$\Pr(\{R_1 = r_1\} \cap \dots \cap \{R_k = r_k\}) = \frac{Q[\text{dlist}, m[\text{dlist}], \text{nlist}, \text{rlist}]}{\text{Multinomial}[n_1, \dots, n_k]}.$$

Thus, we get this generating function  $G[\text{KruskalWallisTies}[\text{dlist}, \text{nlist}]]$  by applying the substitution

$$s1 = \left\{ y[i]^r \rightarrow t^{\wedge} \left( \frac{r^2}{n_i} \right) \mid i = 1, \dots, k \right\}$$

to

$$\frac{q[\text{dlist}, m[\text{dlist}], \text{nlist}]}{\text{Multinomial}[n_1, \dots, n_k]}$$

and the substitution

$$s2 = t^x \rightarrow t^{\wedge} \left( \frac{1}{\Delta} \left[ \frac{12x}{n(n+1)} - 3(n+1) \right] \right)$$

to the result of the first substitution.

```
G[KruskalWallisTies[dlist_, nlist_]] :=
Module[{d, n, k, a, b, Delta, s1, s2},
d = Apply[Plus, dlist];
n = Apply[Plus, nlist];
k = Length[nlist];
a = 12 / (n (n + 1));
b = 3 (n + 1);
Delta = 1 - Apply[Plus, (dlist^3 - dlist)] / (n^3 - n);
s1 = Table[y[i]^r_ -> t^(r^2/nlist[[i]]), {i, 1, k}];
s2 = t^x_ -> t^((a x - b) / Delta);
If[d == n, Apart[
(q[dlist, m[dlist], nlist] /. s1 /. s2) / Apply[Multinomial, nlist]],
Print["d1+...+dl ≠ n1+...+nk"]; Abort[]]
```

```
In[90]:= G[KruskalWallisTies[{2, 2, 3}, {3, 2, 2}]]
```

$$\text{Out[90]} = \frac{2t^{7/50}}{35} + \frac{2t^{33/100}}{35} + \frac{8t^{19/50}}{35} + \frac{2t^{63/50}}{35} + \frac{4t^{153/100}}{35} + \frac{4t^{209/100}}{35} + \frac{t^{69/25}}{35} + \frac{2t^{293/100}}{35} + \frac{4t^{77/25}}{35} + \frac{2t^{17/4}}{35} + \frac{2t^{461/100}}{35} + \frac{t^5}{35} + \frac{2t^{126/25}}{105} + \frac{t^6}{105}$$

Again we have, for example,

```
In[91]:= TableCDFG[KruskalWallisTies[{2, 2, 3}, {3, 2, 2}], 1, 6]
```

```
Out[91]//TableForm=
```

x	CDF[x]
1.26	0.342857
1.53	0.4
2.09	0.514286
2.76	0.628571
2.93	0.657143
3.08	0.714286
4.25	0.828571
4.61	0.885714
5.	0.942857
5.04	0.971429

Lehmann [2] mentions that for large values of  $n_1, \dots, n_k$  it is possible to approximate the null distribution of the test statistics `KruskalWallisTies[dlist, nlist]` by the chi-square distribution with  $k - 1$  degrees of freedom. By experimenting we can investigate the quality of this approximation graphically in the case of small values of  $n_1, \dots, n_k$ . It is again important to remark that this approximation is not very good in the most important region between the 0.95 and the 0.99 quantile (Figure 4).

```
In[92]:= dlist = {3, 2, 1, 3, 2};
nlist = {4, 3, 4};
k = Length[nlist];
plot1 = PlotCDFG[
  KruskalWallisTies[dlist, nlist], DisplayFunction -> Identity];
plot2 = Plot[CDF[ChiSquareDistribution[k - 1], x],
  {x, 0, 10}, DisplayFunction -> Identity];
fig4 = Show[{plot1, plot2}, PlotRange -> All,
  DisplayFunction -> $DisplayFunction, Axes -> True]
```

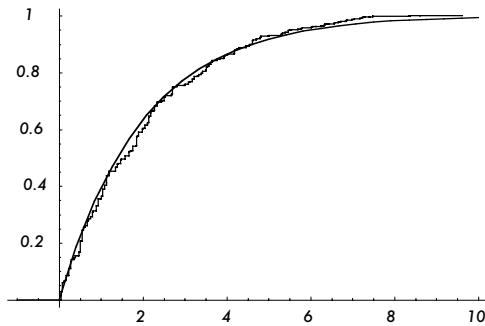
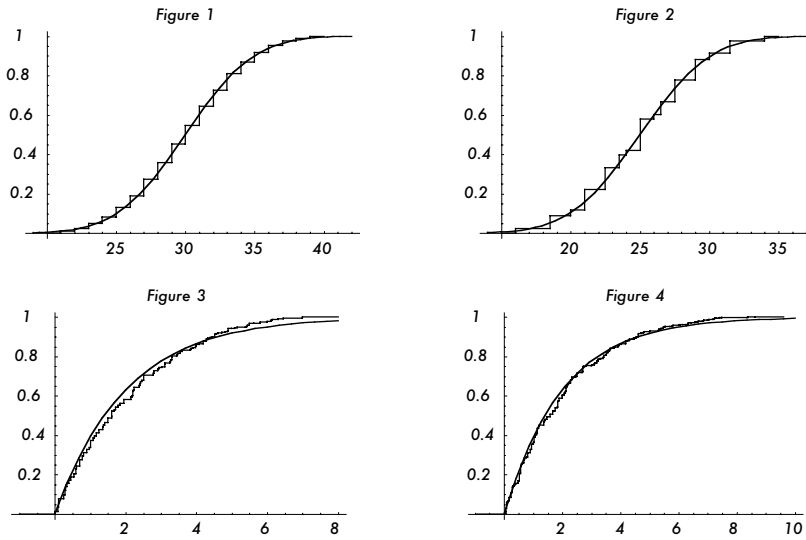


Figure 4.

The following picture shows the approximation of the null distribution of the Wilcoxon rank sum statistics with (Figure 2) and without (Figure 1) ties by an appropriate normal distribution and the approximation of the null distribution of the Kruskal–Wallis statistics with (Figure 4) and without (Figure 3) ties by an appropriate chi-square distribution.



## ■ Back to our Introductory Example

We are now able to calculate the exact value of the 0.95 quantile  $\kappa_{0.95}$  of our test statistics.

```
In[103]:= dlist = {2, 2, 4, 2, 2, 2, 1};
          nlist = {5, 5, 5};
          QTLG[KruskalWallisTies[dlist, nlist], 0.95] // N
```

```
Out[105]= 5.65651
```

Because the value of our test statistics is 5.85062, we will reject the null hypothesis. Approximating the null distribution of our test statistics by the chi-square distribution with 2 degrees of freedom—the 0.95 quantile of this distribution is 5.99145—we would not be able to reject the null hypothesis. This example shows that it is important to work with the exact values of quantiles and not with their approximations.

## ■ Conclusion

We have shown how the method of generating functions can be used to investigate the null distribution of the well-known Wilcoxon rank sum statistics and the

Kruskal–Wallis statistics (with and without ties). The proposed method is not confined to these test statistics; in contrary, the reader should be able now to apply this method to investigate nearly every test statistic based on ranks.

## ■ Acknowledgments

I would like to thank the referee for valuable suggestions.

## ■ References

- [1] *Statistica*, St. Paul, MN: Assessment Systems Corporation, (May 2005) [www.assess.com/Software/StatMain.htm](http://www.assess.com/Software/StatMain.htm).
- [2] E. L. Lehmann, *Nonparametrics: Statistical Methods Based on Ranks*, New York: McGraw-Hill, 1975.
- [3] H. Chernoff and I. R. Savage, "Asymptotic Normality and Efficiency of Certain Nonparametric Test Statistics," *Annals of Mathematical Statistics*, **29**, 1958 pp. 972–994.
- [4] G. E. Noether, *Introduction to Statistics: The Nonparametric Way*, New York: Springer-Verlag, 1991.
- [5] S. Kokoska and C. Nevison, *Statistical Tables and Formulae*, New York: Springer-Verlag, 1989.
- [6] P. Mitic, "Critical Values for the Wilcoxon Signed Rank Statistic," *The Mathematica Journal*, **6**(3), 1996 pp. 73–77.
- [7] W. B. Davenport, *Probability and Random Processes*, New York: McGraw-Hill, 1970.
- [8] V. K. Rohatgi, *An Introduction to Probability Theory and Mathematical Statistics*, New York: John Wiley & Sons, 1976.
- [9] N. L. Johnson and S. Kotz, *Urn Models and Their Application*, New York: John Wiley & Sons, 1977.
- [10] M. Hall Jr., *Combinatorial Theory*, New York: John Wiley & Sons, 1986.

## About the Author

Peter Weiß studied mathematics at the University of Innsbruck. Since 1973 he has been a professor for probability and mathematical statistics at the Johannes Kepler University in Linz. His current areas of interest are point-processes, random walk on random structures, and statistical data analysis for industrial applications. In 1990 he founded the new curriculum "Mechatronics," which is a combination of mechanical engineering, electrical engineering, and computer science.

### Peter Weiß

*Institute of Mathematics  
Johannes Kepler University  
Linz, Austria  
peter.weiss@jk.uni-linz.ac.at*