

5-1-2005

# JMASM17: An Algorithm And Code For Computing Exact Critical Values For Friedman's Nonparametric ANOVA


Sikha Bagui

*The University of West Florida, Pensacola, [bagui@uwf.edu](mailto:bagui@uwf.edu)*

Sbuhash Bagui

*The University of West Florida, Pensacola, [sbagui@uwf.edu](mailto:sbagui@uwf.edu)*

Follow this and additional works at: <http://digitalcommons.wayne.edu/jmasm>

 Part of the [Applied Statistics Commons](#), [Social and Behavioral Sciences Commons](#), and the [Statistical Theory Commons](#)

## Recommended Citation

Bagui, Sikha and Bagui, Sbuhash (2005) "JMASM17: An Algorithm And Code For Computing Exact Critical Values For Friedman's Nonparametric ANOVA," *Journal of Modern Applied Statistical Methods*: Vol. 4 : Iss. 1 , Article 28.

DOI: 10.22237/jmasm/1114907280

Available at: <http://digitalcommons.wayne.edu/jmasm/vol4/iss1/28>

This Algorithms and Code is brought to you for free and open access by the Open Access Journals at DigitalCommons@WayneState. It has been accepted for inclusion in Journal of Modern Applied Statistical Methods by an authorized editor of DigitalCommons@WayneState.

## JMASM17: An Algorithm And Code For Computing Exact Critical Values For Friedman's Nonparametric ANOVA

Sikha Bagui Subhash Bagui  
The University of West Florida, Pensacola

---

Provided in this article is an algorithm and code for computing exact critical values (or percentiles) for Friedman's nonparametric rank test for  $k$  related treatment populations using Visual Basic (VB.NET). This program has the ability to calculate critical values for any number of treatment populations ( $k$ ) and block sizes ( $b$ ) at any significance level ( $\alpha$ ). We developed an exact critical value table for  $k = 2(1)5$  and  $b = 2(1)15$ . This table will be useful to practitioners since it is not available in standard nonparametric statistics texts. The program can also be used to compute any other critical values.

Key words: Friedman's test, randomized block designs (RBD), ANOVA, Visual Basic

---

### Introduction

While experimenting (or dealing) with randomized block designs (RBDs) (or one-way repeated measures designs), if the normality of treatment populations or the assumptions of equal variances are not met, or the data are in ranks, it is recommended that Friedman's rank-based nonparametric test be used as an alternative to the conventional  $F$  test for the RBD (or one-way repeated measures analysis of variance) for  $k$  related treatment populations. This test was developed by Friedman (1937), and was designed to test the null hypothesis that all the  $k$  treatment populations are identical versus the alternative that at least two of the treatment populations differ in location. This test is based on a statistic that is a rank analogue of SST (total sum of squares) for the RBD and

is computed in the following manner. After the data from a RBD are obtained, the observed values in each block  $b$  are ranked from 1 (the smallest in the block) to  $k$  (the largest in the block). Let  $R_i$  denote the sum of the ranks of the values corresponding to treatment population  $i$ ,  $i = 1, 2, \dots, k$ . Then the Friedman's test statistic is given by

$$F_r = \frac{12}{bk(k+1)} \sum_{i=1}^k R_i^2 - 3b(k+1).$$

If the null hypothesis is true, it is expected that the rankings be randomly distributed within each block. If that is the case, the sum of the rankings in each treatment population will be approximately equal, and the resulting value of  $F_r$  will be small.

If the alternative hypothesis is true, the expectation is that this will lead to differences among the  $R_i$  values and obtain correspondingly large values of  $F_r$ . Thus, the null hypothesis is rejected in favor of the alternative hypothesis for large values of  $F_r$ . Exact null sampling distribution of  $F_r$  is not known. But, as with the Kruskal-Wallis (1952) statistic, the null distribution of the Friedman's  $F_r$  can be approximated by a chi-square ( $\chi^2$ ) distribution

---

Sikha Bagui is an Assistant Professor in the Department of Computer Science. Her areas of research are database and database design, data mining, pattern recognition, and statistical computing. Email: bagui@uwf.edu. Subhash Bagui is a Professor in the Department of Mathematics and Statistics. His areas of research are statistical classification and pattern recognition, bio-statistics, construction of designs, tolerance regions, statistical computing and reliability. Email: sbagui@uwf.edu.

with  $(k-1)$  degrees of freedom as long as  $b$  is large. Empirical evidence indicates that the approximation is adequate if the number of blocks  $b$  and /or the number of treatment populations  $k$  exceeds 5.

Again, in small sample situations, the chi-square approximation will not be adequate. Common statistics books with a chapter on nonparametric statistics do not provide exact critical values for Friedman's  $F_r$  test. Conover (1999) did not provide exact critical values for the Friedman's  $F_r$  test, but Hollander and Wolf (1973) and Lehmann (1998) provided a partial exact critical values table for Friedman's  $F_r$  test. Most commonly used statistical software such as MINITAB and SPSS provide only the asymptotic P-value for Friedman's  $F_r$  statistic. In view of this, in this article, we provide a VB.NET program that computes the exact critical values of Friedman's  $F_r$  statistic for any number of blocks  $b$  and any number of treatment populations  $k$ , at any significance level ( $\alpha$ ).

Also provided is an exact critical values table for the Friedman's  $F_r$  test for various combinations of (small) block sizes  $b$  and (small) treatment population sizes  $k$ . Headrick (2003) wrote an article for generating exact critical values for the Kruskal-Wallis nonparametric test using Fortran 77. We used his idea to generate exact critical values for Friedman's  $F_r$  test using VB.NET. VB.NET is user friendly and more accessible. Our VB.NET program works well with reasonable values of  $b$  and  $k$ .

### Methodology

In order to generate the critical values of Friedman's  $F_r$  statistic, we need to have the null distribution for  $F_r$ . In Friedman's test, the null hypothesis is that the effect of all  $k$  treatment populations are identical. Thus it is reasonable to use such types of null distributions for the  $F_r$  statistic that are derived under the assumption

that all observations for treatment populations are from the same population.

Therefore, to find the null distribution of the  $F_r$  statistic, first, generate  $b$  uniform pseudo-random numbers from the interval  $(0,1)$  for each of the  $k$  treatment populations. Assume that the probability of a tie is zero. Then random variates within each block are ranked from 1 to  $k$ . The program then calculates rank sums of each treatment population,  $R_i$ , and computes the value of the  $F_r$  statistic

$$F_r = \frac{12}{bk(k+1)} \sum_{i=1}^k R_i^2 - 3b(k+1).$$

This process is replicated a sufficient number of times until the null distribution of the  $F_r$  statistic is modeled adequately. Then the program returns a critical value that is associated with a percentile fraction of 0.90, 0.95, 0.975, or 0.99 (or equivalently a significance level alpha of 0.10, 0.05, 0.025, or 0.01). In some cases returned values may be true for a range of P-values.

With adequate number of runs, this VB.NET program yields the same values reported by Lehmann (1998) in Table M. In Table 1 below, we provide critical values for the  $F_r$  test for  $b = 2(1)15$ ,  $k = 2(1)5$  and  $\alpha = 0.1, 0.05, 0.025, 0.01$ . The notation  $F_{1-\alpha}$  in the Table 1 means  $(1-\alpha)100\%$  percentile of the  $F_r$  statistic which is equivalent to  $\alpha$  level critical value of the  $F_r$  statistic. This table will be useful to the practitioners since it is not available in standard statistics texts with a chapter on nonparametric statistics. The critical values in Table 1 are generated using 1 million replications in each case.

Table 1. Critical values for Friedman's  $F_r$  test.

| Rows ( $b$ ) | Columns ( $k$ ) | $F_{0.90}$ | $F_{0.95}$ | $F_{0.975}$ | $F_{0.99}$ |
|--------------|-----------------|------------|------------|-------------|------------|
| 2            | 2               | 2.0000     | 2.0000     | 2.0000      | 2.0000     |
| 3            | 2               | 3.0000     | 3.0000     | 3.0000      | 3.0000     |
| 4            | 2               | 4.0000     | 4.0000     | 4.0000      | 4.0000     |
| 5            | 2               | 1.800      | 5.0000     | 5.0000      | 5.0000     |
| 6            | 2               | 2.6667     | 2.6667     | 2.6667      | 2.6667     |
| 7            | 2               | 3.5714     | 3.5714     | 3.5714      | 7.0000     |
| 8            | 2               | 2.0000     | 4.5000     | 4.5000      | 4.5000     |
| 9            | 2               | 2.7778     | 2.7778     | 5.4444      | 5.4444     |
| 10           | 2               | 3.6000     | 3.6000     | 3.6000      | 6.4000     |
| 11           | 2               | 2.2727     | 4.4545     | 4.4545      | 7.3636     |
| 12           | 2               | 3.000      | 3.0000     | 5.3333      | 5.3333     |
| 13           | 2               | 1.9231     | 3.7692     | 3.7692      | 6.2308     |
| 14           | 2               | 2.5714     | 4.5714     | 4.5714      | 7.1429     |
| 15           | 2               | 3.2667     | 3.2667     | 5.4000      | 5.4000     |
| 2            | 3               | 4.0000     | 4.0000     | 4.0000      | 4.0000     |
| 3            | 3               | 4.6667     | 4.6667     | 6.0000      | 6.0000     |
| 4            | 3               | 4.5000     | 6.0000     | 6.5000      | 6.5000     |
| 5            | 3               | 4.8000     | 5.2000     | 6.4000      | 7.6000     |
| 6            | 3               | 4.3333     | 6.3333     | 7.0000      | 8.3333     |
| 7            | 3               | 4.5714     | 6.0000     | 7.1429      | 8.0000     |
| 8            | 3               | 4.7500     | 5.2500     | 7.0000      | 7.7500     |
| 9            | 3               | 4.6667     | 6.0000     | 6.8889      | 8.6667     |
| 10           | 3               | 4.2000     | 5.6000     | 7.4000      | 8.6000     |
| 11           | 3               | 4.9091     | 5.6364     | 7.0909      | 8.9091     |
| 12           | 3               | 4.6667     | 6.1667     | 7.1667      | 8.6667     |
| 13           | 3               | 4.5841     | 5.9469     | 7.2920      | 9.0796     |
| 14           | 3               | 4.4286     | 5.5714     | 7.0000      | 9.0000     |
| 15           | 3               | 4.8000     | 5.7333     | 6.9333      | 8.5333     |
| 2            | 4               | 5.4000     | 5.4000     | 6.0000      | 6.0000     |
| 3            | 4               | 5.8000     | 7.0000     | 7.4000      | 8.2000     |
| 4            | 4               | 6.0000     | 7.5000     | 8.1000      | 9.3000     |
| 5            | 4               | 6.1200     | 7.3200     | 8.2800      | 9.7200     |
| 6            | 4               | 6.2000     | 7.4000     | 8.6000      | 10.0000    |
| 7            | 4               | 6.2571     | 7.6286     | 8.6571      | 10.3714    |
| 8            | 4               | 6.1500     | 7.5000     | 8.8500      | 10.3500    |
| 9            | 4               | 6.0667     | 7.5333     | 8.7333      | 10.4667    |
| 10           | 4               | 6.2400     | 7.5600     | 8.8800      | 10.6800    |
| 11           | 4               | 6.1636     | 7.5818     | 8.8909      | 10.6364    |
| 12           | 4               | 6.1000     | 7.6000     | 9.0000      | 10.7000    |
| 13           | 4               | 6.0462     | 7.6154     | 9.0000      | 10.7539    |
| 14           | 4               | 6.2571     | 7.6286     | 9.0000      | 10.8857    |
| 15           | 4               | 6.2000     | 7.5600     | 9.0800      | 10.7600    |
| 2            | 5               | 6.8000     | 7.2000     | 7.6000      | 7.6000     |
| 3            | 5               | 7.2000     | 8.2667     | 9.3333      | 9.8667     |
| 4            | 5               | 7.4000     | 8.6000     | 9.6000      | 11.0000    |
| 5            | 5               | 7.5200     | 8.8000     | 10.0800     | 11.5200    |
| 6            | 5               | 7.6000     | 8.9333     | 10.2667     | 11.8667    |
| 7            | 5               | 7.6571     | 9.0286     | 10.5143     | 12.0043    |
| 8            | 5               | 7.7000     | 9.2000     | 10.6400     | 12.2000    |
| 9            | 5               | 7.6444     | 9.1556     | 10.5778     | 12.3556    |
| 10           | 5               | 7.6800     | 9.2000     | 10.6400     | 12.4000    |
| 11           | 5               | 7.7091     | 9.2363     | 10.7636     | 12.5818    |
| 12           | 5               | 7.6667     | 9.2667     | 10.7333     | 12.5333    |
| 13           | 5               | 7.6923     | 9.2923     | 10.7692     | 12.7385    |
| 14           | 5               | 7.7143     | 9.3143     | 10.8000     | 12.7429    |
| 15           | 5               | 7.6800     | 9.3333     | 10.8267     | 12.7467    |

## Conclusion

In case of large values of  $b$  and  $k$ , the program needs a large number of replications in order to adequately model the null distribution for Friedman's  $F_r$  statistic. The replication numbers should be in increasing order such as 10,000, 50,000, 100,000, 500,000, and 1,000,000 etc. and the process stopped once two consecutive values are almost the same. If there are  $b$  blocks and  $k$  treatment populations, then at least  $(k!)^b$  are necessary for a near fit for the  $F_r$  statistic.

For a good fit of  $F_r$ , one needs many more replications than  $(k!)^b$ . The VB.NET program is given in the Appendix. This program allows the user to provide values for replication numbers, block sizes, treatment population numbers, and the percentile fractions. Based on this, the program will return a critical value. Also, remember that distribution of Friedman's  $F_r$  statistic is discrete, so it not possible to achieve the exact level of significance. Thus the critical values obtained here correspond to approximate level of significance 0.01, 0.025, 0.05, and 0.10.

## References

- Conover, W. J. (1999). *Practical Nonparametric Statistics*. (3<sup>rd</sup> ed.). New York: Wiley.
- Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32, 675-701.
- Headrick, T. C. (2003). An algorithm for generating exact critical values for the Kruskal-Wallis One-way ANOVA. *Journal of Modern Applied Statistical Methods*, 2, 268-271.
- Hollander, M., & Wolfe, D. A. (1973). *Nonparametric Statistical Methods*. New York: John Wiley.
- Kruskal, W. H., & Wallis, W. A. (1952). Use of ranks in one-criterion analysis of variance. *Journal of American Statistical Association*, 47, 583-621.
- Lehmann, E. L. (1998). *Nonparametrics: Statistical Methods Based on Ranks*. New Jersey: Prentice Hall.
- Minitab. (2000). *Minitab for Windows, release 13.3*, Minitab, Inc., State College, PA.
- SPSS. (2002). *SPSS for Windows, version 11.0*, SPSS, Inc., Chicago, IL.

## Appendix:

```
Imports System.Windows.Forms
```

```
Public Class Form1
```

```
    Inherits System.Windows.Forms.Form
```

```
    Dim sum = 0, squared = 0, square_sum = 0, m = 0, n = 0, i = 0, j = 0, k = 0,
```

```
    l = 0, p = 0, q = 0, r As Integer
```

```
    Dim count = 0, v = 0, z As Integer
```

```
    Dim num As Single
```

```
    Dim percentile As Single
```

```
    Dim f As Single
```

```
    Dim file1 As System.IO.StreamWriter
```

```

Dim array1(,) As Single = New Single(,) {}
Dim array6(,) As Single = New Single(,) {}
Dim array3() As Single = New Single() {}
Dim array4() As Single = New Single() {}
Dim array5() As Integer = New Integer() {}
Dim array7() As Integer = New Integer() {}
Dim array8() As Single = New Single() {}
Dim array9() As Single = New Single() {}

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    'Calling the random number generator
    Randomize()
End Sub

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
    m = Val(TextBox1.Text) 'm is the number of rows(blocks)
    n = Val(TextBox2.Text) 'n is the number of columns
    z = Val(TextBox3.Text) 'z is the number of runs
    percentile = Val(TextBox4.Text) 'percentile value

    'Defining the arrays
    Dim array1(m, n) As Single
    Dim array6(m, n) As Single
    Dim array3(n) As Single
    Dim array4(n) As Single
    Dim array5(n) As Integer
    Dim array7(n) As Integer
    Dim array8(z) As Single
    Dim array9(z) As Single

    Dim row, col As Integer
    Dim output As String

    For p = 1 To z
        output = " "
        'creating initial m x n random array
        For row = 1 To m
            For col = 1 To n
                array1(row, col) = Rnd()
                output &= array1(row, col) & " "
            Next
            output &= vbCrLf
        Next

        j = 1
        k = 1
        For r = 1 To array1.GetUpperBound(0)

            'pulling out one row
            For col = 1 To n 'array1.GetUpperBound(0)
                num = array1(j, col)
                array3(col) = num
                output &= array3(col) & " "
            Next

```

```

j = j + 1

'copying one row into new array
For row = 1 To array3.GetUpperBound(0)
    array4(row) = array3(row)
Next

'sorting one row
Array.Sort(array3) 'Array3 - is sorted array

'ranking row
For row = 1 To array4.GetUpperBound(0)
    For i = 1 To array3.GetUpperBound(0)
        If array4(row) = array3(i) Then
            array5(row) = i
        End If
    Next
Next

'putting row back into two dimensional array
For row = 1 To array5.GetUpperBound(0)
    output &= array5(row) & " "
    array6(k, row) = array5(row)
Next
output &= vbCrLf
k = k + 1
Next
output = " "

'displaying two dimensional array
For row = 1 To array6.GetUpperBound(0)
    For col = 1 To n 'array6.GetUpperBound(0)
        output &= array6(row, col) & " "
    Next
    output &= vbCrLf
Next

'summing columns in two dimensional array
l = 1
sum = 0
square_sum = 0
For col = 1 To n 'array6.GetUpperBound(0)
    For row = 1 To array6.GetUpperBound(0)
        sum += array6(row, l)
    Next
    output = sum
    square_sum = sum * sum
    array7(l) = square_sum
    output &= vbCrLf
    output &= array7(l) & " "
    l = l + 1
    sum = 0
    square_sum = 0

```

```

Next
    f = 0
    squared = 0
= 1 To array7.GetUpperBound(0)
    squared += array7(row)
Next
    output = squared
    output = " "
    f = Convert.ToSingle(12 / (m * n * (n + 1)) * squared - 3 * m *
    (n + 1))
    array8(p) = f
    output &= array8(p) & " "
    f = 0
    squared = 0
Next
output = " "

For row = 1 To array8.GetUpperBound(0)
    output &= array8(row) & " "
Next

For row = 1 To array8.GetUpperBound(0)
    array9(row) = array8(row)
    output &= array9(row) & " "
Next

Array.Sort(array9) 'Array9 - sorted F values

For row = 1 To array9.GetUpperBound(0)
    output &= array9(row) & " "
Next

output = " "

count = 0
For row = 1 To array9.GetUpperBound(0)
    count += 1
Next

output = count

v = percentile * count

output = " "

output = array9(v)

MessageBox.Show(output, "95% percentile value")

End Sub

End Class

```