# scmamp: Statistical Comparison of Multiple Algorithms in Multiple Problems

*by Borja Calvo and Guzmán Santafé*

**Abstract** Comparing the results obtained by two or more algorithms in a set of problems is a central task in areas such as machine learning or optimization. Drawing conclusions from these comparisons may require the use of statistical tools such as hypothesis testing. There are some interesting papers that cover this topic. In this manuscript we present **scmamp**, an R package aimed at being a tool that simplifies the whole process of analyzing the results obtained when comparing algorithms, from loading the data to the production of plots and tables.

Comparing the performance of different algorithms is an essential step in many research and practical computational works. When new algorithms are proposed, they have to be compared with the state of the art. Similarly, when an algorithm is used for a particular problem, its performance with different sets of parameters has to be compared, in order to tune them for the best results.

When the differences are very clear (e.g., when an algorithm is the best in all the problems used in the comparison), the direct comparison of the results may be enough. However, this is an unusual situation and, thus, in most situations a direct comparison may be misleading and not enough to draw sound conclusions; in those cases, the statistical assessment of the results is advisable.

The statistical comparison of algorithms in the context of machine learning has been covered in several papers. In particular, the tools implemented in this package are those presented in Demšar (2006); García and Herrera (2008); García et al. (2010). Another good review that covers, among other aspects, the statistical assessment of the results in the context of supervised classification can be found in Santafé et al. (2015).

## Existing tools

Some of the methods presented in the referred papers are well known procedures that are included in classical statistics tools. As an example, *p-value* correction methods such as Holm (Holm, 1979) or omnibus tests such as Friedman's (Friedman, 1937) are implemented in R's base package. However, other methods are neither so well known nor trivial to implement. Worth highlighting is Bergmann and Hommel's procedure (Bergmann and Hommel, 1988) to correct the *p-values* when all the algorithms are compared pair-wise (see García and Herrera, 2008, page 2681).

There are tools that implement some of the methods included in this package. The first one is KEEL (Alcalá-Fdez et al., 2008), a Java toolbox that includes a module for the statistical assessment of the results obtained in a given experiment. However, although it can be used independently from the rest of the toolbox, its GUI offers only a limited combination of methods and any other analysis requires programming within Java.

As an alternative to the KEEL GUI, STATService 2.0 (Parejo et al., 2012) provides a web service to perform statistical analysis of multiple algorithms using KEEL's code. Along the same lines we have STAC (Rodríguez-Fdez et al., 2015), a Python web service that allows running different types of parametric and non-parametric tests using a simple interface and its own implementation of the methods.

The goal of **scmamp** is to provide a simple pipeline that allows any researcher to load their complete set of results, analyze them and produce the material needed for publication (tables and plots).

Under some circumstances we may be interested in analyzing the results in different groups of problems. An example of such a situation are the results presented in Blum et al. (2015), where the behavior of a set of algorithms was tested in problems of different size and complexity[1]. In order to deal with these kinds of problems, conversely to other existing tools, the package offers the possibility of subsetting the results, which can be handy when the problems can be subdivided into different groups (e.g., based on their size).

Another advantage of the **scmamp** package over other existing implementations is that the functions that perform the analyses accept additional user-defined test and correction functions, increasing

[1] As we will show later, part of these results are available in **scmamp** as an example of the type of results matrix used by the package.

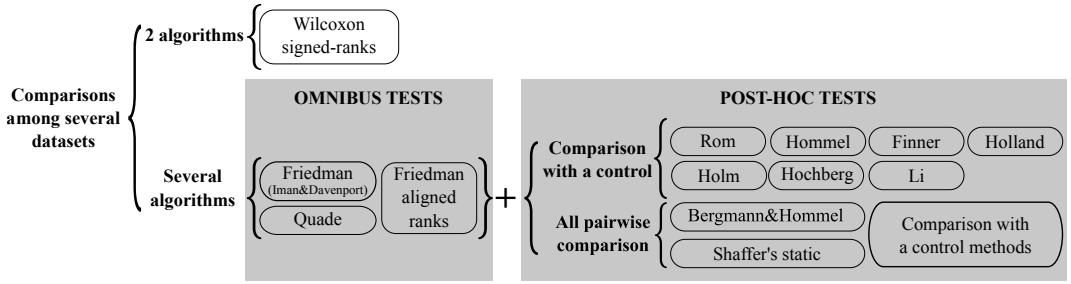


**Figure 1:** Recommended statistical test for different scenarios.

the flexibility of the analysis. Moreover, all the correction methods included in the **stats** package through the `p.adjust` function can be directly used in the **scmamp** package.

Finally, we mention that although KEEL and STATService generate tables to be directly used in publications, they do not generate plots. In our package we have included two functions to graphically represent the results of the comparison. Moreover, performing the analysis in R allows the user to easily create his/her own plots.

## Brief overview of the statistical tests and general recommendations

Several publications (Demšar, 2006; García and Herrera, 2008; García et al., 2010; Santafé et al., 2015) have stated a basic classification of general machine learning scenarios and the associated statistical tests which are appropriate for each situation. This package is mainly focused in the comparison of multiple algorithms in multiple datasets. However, due to the flexibility of the implemented functions, it can also be adapted to other situations. Figure 1 summarizes the most common situations when evaluating machine learning algorithms.

Comparisons of two algorithms among a set of different problems are common in the literature in order to decide between two competitors. The estimated scores for each algorithm on each problem are independent. However, as they may be obtained from different application domains (or problems with different characteristics), it is highly debatable whether they can be averaged over in order to obtain a meaningful overall estimation of the algorithm's performance. Consequently, non-parametric methods such as Wilcoxon signed-rank are usually recommended (Demšar, 2006).

On the other hand, in order to compare multiple algorithms in multiple problems, the general recommended methodology is as follows. First, we apply an omnibus test to detect if at least one of the algorithms performs differently than the others. Second, if we find a significant difference, then we apply a pair-wise test with the corresponding *post-hoc* correction for multiple comparisons. The Friedman test with Iman and Davemport extension is probably the most popular omnibus test, and it is usually a good choice when comparing more than five different algorithms. By contrast, when comparing five or fewer different algorithms, Friedman aligned ranks and the Quade test are more powerful alternatives (García et al., 2010).

Regarding *post-hoc* tests, the choice depends on the pair-wise comparisons: comparison with a control or all pair-wise comparisons. When comparing all the algorithms with a control, Bonferroni is the most simple but the least powerful procedure and, thus, it is not recomended in practice. By contrast, Hommel and Rom are the two most powerful procedures but they are also more complex than other methods. Alternatively, Finner is a simple procedure and it is the next with the highest power (see García et al., 2010). Nevertheless, except for Bonferroni, in practice there are not very big differences in the power of the *post-hoc* tests. Therefore, in general, Finner's method is a good choice due to its simplicity and power.

Similarly, when an all pair-wise comparison is conducted, those procedures to perform comparisons with a control can be used. In this case, the Nemenyi's test is the most simple but less powerful alternative and it is not usually recommended in practice. Alternatively, specific methods for all pair-wise comparison have a higher power. The one with highest power is Bergmann and Hommel, but it is a complex and computationally expensive method. The **scmamp** package optimizes this method for up to nine algorithms by having some pre-computed operations. Therefore, comparing more than nine algorithms with the Bergmann and Hommel procedure is unfeasible in practice. In those situations, Shaffer's static method or even other more simple procedures such as Finer and Holm are recommended.

## Brief examples

In this section we will illustrate the use of the package in three different situations. For a more detailed discussion on the use of the different functions the reader is referred to the package's vignettes. These may be accessed with the command `browseVignettes('scmamp')`.

The first example of use comes from García and Herrera (2008). Actually, we will use the set of results presented in that paper, which are included in the package in the variable `data.gh.2008`. These results collect the performance of a number of supervised classification algorithms in a set of 30 datasets. The goal of the study is comparing the different algorithms and determining which outperforms which. For more details, the reader is referred to García and Herrera (2008).

```
> library('scmamp')
> head(data.gh.2008)

            C4.5 k-NN(k=1) NaiveBayes Kernel   CN2
Abalone*   0.219     0.202      0.249  0.165 0.261
Adult*     0.803     0.750      0.813  0.692 0.798
Australian 0.859     0.814      0.845  0.542 0.816
Autos      0.809     0.774      0.673  0.275 0.785
Balance    0.768     0.790      0.727  0.872 0.706
Breast     0.759     0.654      0.734  0.703 0.714
```

The goal is analyzing all the pair-wise comparisons. Therefore, the first hypothesis to test is whether all the algorithms perform equally or, in contrast, some of them have a significantly different behavior. Then, all the differences are tested for every pair of algorithms and the resulting *p-values* are corrected. There are different ways to report the results, depending on the *post-hoc* analysis. A very intuitive tool is Demsar's critical difference plots. Although easy to interpret, these plots are based on the Nemenyi test, which is a very conservative one. There are other alternatives that can be used with more powerful methods (such as Bergmann and Hommel's correction). In this example we will use the `drawAlgorithmGraph` function which creates a graph based on the *p-values* corrected by any method.

First, we check the differences using the Iman and Davenport omnibus test.

```
> imanDavenportTest(data.gh.2008)

Iman Davenport's correction of Friedman's rank sum test

data:  data.gh.2008
Corrected Friedman's chi-squared = 14.3087, df1 = 4, df2 = 116,
p-value = 1.593e-09
```

The *p-value* shown above denotes that there is at least one algorithm that performs differently than the rest and, therefore, we can proceed with the *post-hoc* analysis of the results. There are several alternatives to perform this analysis, but we will focus on two. The first alternative is the Nemenyi test. Although, this test is not a recommended choice in practice since it is very conservative and has a low power, it is shown in this example because its associated plot is quite illustrative.

```
> nm <- nemenyiTest(data.gh.2008)
> nm

Nemenyi test

data:  data.gh.2008
Critical difference = 1.1277, k = 5, df = 145

> nm$diff.matrix

          C4.5  k-NN(k=1) NaiveBayes    Kernel        CN2
[1,]  0.000000 -1.1500000 -0.1000000 -2.233333 -1.0166667
[2,] -1.150000  0.0000000  1.0500000 -1.083333  0.1333333
[3,] -0.100000  1.0500000  0.0000000 -2.133333 -0.9166667
[4,] -2.233333 -1.0833333 -2.1333333  0.000000  1.2166667
[5,] -1.016667  0.1333333 -0.9166667  1.216667  0.0000000
```

This procedure determines the *critical difference*. Any two algorithms whose performance difference is greater that the critical difference are regarded as significantly different. As can be seen in the code

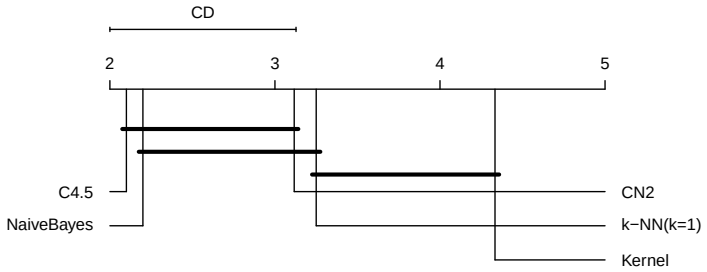


**Figure 2:** Example of critical difference plot.

above, the differences between every pair of algorithms is stored in the `diff.matrix` element of the result. The Nemenyi test has the advantage of having an associated plot to represent the results of the comparison. This plot can be obtained as follows.

```
> plotCD(results.matrix = data.gh.2008, alpha = 0.05)
```

The result can be seen in Figure 2. In this plot each algorithm is placed on an axis according to its average ranking. Then, those algorithms that show no significant differences are grouped together using a horizontal line. The plot also shows the size of the critical difference required for considering two algorithm as significantly different.

The second alternative shown in this example is the Friedman *post-hoc* test with Bergmann and Hommel's correction. Both steps can be carried out in a single line of code.[2]

```
> test.res <- postHocTest(data = data.gh.2008, test = 'friedman', correct = 'bergmann')
> test.res

$summary
       C4.5 k-NN(k=1) NaiveBayes    Kernel       CN2
[1,] 0.7797    0.6791     0.7565 0.5693667 0.7285333

$raw.pval
                   C4.5   k-NN(k=1)   NaiveBayes       Kernel         CN2
C4.5                 NA 0.004848763 8.064959e-01 4.486991e-08 0.012763008
k-NN(k=1)  4.848763e-03          NA 1.011233e-02 7.963489e-03 0.743971478
NaiveBayes 8.064959e-01 0.010112334           NA 1.736118e-07 0.024744672
Kernel     4.486991e-08 0.007963489 1.736118e-07           NA 0.002880485
CN2        1.276301e-02 0.743971478 2.474467e-02 2.880485e-03          NA

$corrected.pval
                   C4.5  k-NN(k=1)   NaiveBayes       Kernel        CN2
C4.5                 NA 0.02909258 1.000000e+00 4.486991e-07 0.03828902
k-NN(k=1)  2.909258e-02         NA 3.185396e-02 3.185396e-02 1.00000000
NaiveBayes 1.000000e+00 0.03185396           NA 1.041671e-06 0.03828902
Kernel     4.486991e-07 0.03185396 1.041671e-06           NA 0.01152194
CN2        3.828902e-02 1.00000000 3.828902e-02 1.152194e-02         NA
```

The above code runs the *post-hoc* test, computing the raw *p-value* for each pair of algorithms. These *p-values* are also corrected for multiple testing using Bergman and Hommel's correction. Additionally, a summary with the average values of each algorithm over all the dataset is obtained.

The package offers two ways of presenting the results obtained in the analysis. On the one hand, we can create a LaTeX table using the function `writeTabular`. This function includes parameters to control several aspects of the table. For more details on its use the reader is referred to the vignette covering the data loading and manipulation. The table generated can be seen in Table 1.

```
> # LaTeX formated: Significances highlighted in bold
> bold <- test.res$corrected.pval < 0.05
> bold[is.na(bold)] <- FALSE
> writeTabular(table = test.res$corrected.pval, format = 'f', bold = bold,
```

[2]These steps can be carried out independently using the functions implemented in the package. For more information the reader is referred to the package documentation.

| | C4.5 | k-NN(k=1) | NaiveBayes | Kernel | CN2 |
|---|---|---|---|---|---|
| C4.5 | n/a | **0.029** | 1.000 | **0.000** | **0.038** |
| k-NN(k=1) | **0.029** | n/a | **0.032** | **0.032** | 1.000 |
| NaiveBayes | 1.000 | **0.032** | n/a | **0.000** | **0.038** |
| Kernel | **0.000** | **0.032** | **0.000** | n/a | **0.012** |
| CN2 | **0.038** | 1.000 | **0.038** | **0.012** | n/a |

**Table 1:** Example of table generated by the package.

```
+              hrule = 0, vrule = 0)

\begin{tabular}{|l|llll|}
\hline
 & C4.5 & k-NN(k=1) & NaiveBayes & Kernel & CN2 \\
\hline
C4.5 & n/a & {\bf 0.029} & 1.000 & {\bf 0.000} & {\bf 0.038} \\
k-NN(k=1) & {\bf 0.029} & n/a & {\bf 0.032} & {\bf 0.032} & 1.000 \\
NaiveBayes & 1.000 & {\bf 0.032} & n/a & {\bf 0.000} & {\bf 0.038} \\
Kernel & {\bf 0.000} & {\bf 0.032} & {\bf 0.000} & n/a & {\bf 0.012} \\
CN2 & {\bf 0.038} & 1.000 & {\bf 0.038} & {\bf 0.012} & n/a \\
\hline
\end{tabular}
```

On the other hand, the results can be shown in a graph that represents the algorithms that show no significant differences as connected nodes.

```
> average.ranking <- colMeans(rankMatrix(data.gh.2008))
> drawAlgorithmGraph(pvalue.matrix = test.res$corrected.pval,
+                    mean.value = average.ranking)
```

In the graph, shown in Figure 3, we can see that, according to the test, there are no significant differences within the pairs C4.5/NaiveBayes and kNN/CN2. Compared with the critical difference plot, this method is able to detect more differences (for example, the Nemenyi test does not detect significant differences between the kNN and kernel algorithms).

For the second example we will use a dataset where the problems can be subdivided into groups. The dataset shows the results obtained by eight different algorithms used to find large independent sets in graphs. The performance of the algorithms is evaluated in a number of randomly generated graphs of different size and density. Thus the results matrix we will use contains in each row the result obtained by each algorithms for a given problem. This dataset is part of the results in Blum et al. (2015), and it is also included in the package.

```
> head(data.blum.2015)

  Size Radius FruitFly Shukla Ikeda Turau Rand1 Rand2 FrogCOL FrogMIS
1 1000  0.049      223    213   214   214   214   212     246     226
2 1000  0.049      224    207   209   216   205   211     241     219
3 1000  0.049      219    206   215   214   209   213     243     221
4 1000  0.049      227    208   218   218   215   219     251     230
5 1000  0.049      231    218   210   212   211   217     243     239
6 1000  0.049      230    214   214   208   211   206     246     229
```

The first two columns indicate the size and the density of the random graph while the last eight columns contain the results obtained by each algorithm. Although we could directly use the data loaded in the package, we use this example to briefly show how data can be loaded from one or more files.

Depending on how the experimentation is conducted, we may end up with a number of files, each containing part of the full result. Moreover, it can happen that the information we need is not only in the file content, but also in its name. The package includes a set of functions whose goal is simplifying this task of combining results. Here we will show how the data can be loaded from a set of example files distributed with the package. For further information about how data can be loaded the reader is referred to the corresponding package vignette.

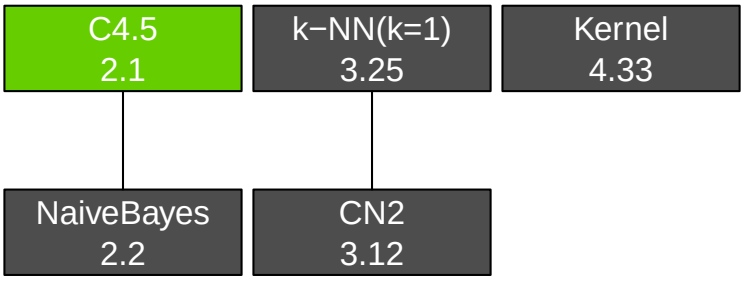


**Figure 3:** Example of algorithm graph.

```
> dir <- paste(system.file('loading_tests', package = 'scmamp'),
+              'experiment_files', sep ='/')

[1] "rgg_size_1000_r_0.049_FrogCOL.out"
[2] "rgg_size_1000_r_0.049_FrogMIS.out"
[3] "rgg_size_1000_r_0.049_FruitFly.out"
[4] "rgg_size_1000_r_0.049_Ikeda.out"
[5] "rgg_size_1000_r_0.049_Rand1.out"

> pattern <- 'rgg_size_([0-9]*)_r_(0.[0-9]*)_([a-z,A-Z,1,2]*).out'
> var.names <- c('Size', 'Radius', 'Algorithm')
> dataset <- readExperimentDir(directory = dir, names = var.names,
+                              fname.pattern = pattern, alg.var.name = 'Algorithm',
+                              value.col = 'Evaluation', col.names = 'Evaluation')
> head(dataset)

  Size Radius FrogCOL FrogMIS FruitFly Ikeda Rand1 Rand2 Shukla Turau
1 1000  0.049     246     226      223   214   214   212    213   214
2 1000  0.049     241     219      224   209   205   211    207   216
3 1000  0.049     243     221      219   215   209   213    206   214
4 1000  0.049     251     230      227   218   215   219    208   218
5 1000  0.049     243     239      231   210   211   217    218   212
6 1000  0.049     246     229      230   214   211   206    214   208
```

In order to extract information from the file names, the name structure has to be defined as a regular expression. In the above code we can see that there are three variables that are extracted from the file names: `Size`, `Radius` and `Algorithm`. Note that the latter does not appear in the final dataset, as it has been used to generate the different columns in the table.

In this dataset there are 30 problems for each combination of size and radius. For each problem the table contains the results obtained with eight algorithms. In this case, we want to compare all the algorithms with the reference one, FrogCOL. Thus, we will compare, using an Iman and Davenport test, the average performance of the algorithms for each combination of size and radius. First, we compute the mean values using the `summarizeData` function. Then, we run the test.

```
> dataset.means <- summarizeData(dataset, group.by = c('Radius', 'Size'),
+                                fun = mean, na.rm = TRUE)
> imanDavenportTest(data.gh.2008)

Iman Davenport's correction of Friedman's rank sum test

data:  data.gh.2008
Corrected Friedman's chi-squared = 14.3087, df1 = 4, df2 = 116,
p-value = 1.593e-09
```

The small *p-value* obtained in the test indicates that there is strong statistical evidence to state that at least one algorithm performs differently than the rest. Thus, a *post-hoc* test (Friedman + Finner's correction) can be conducted to detect differences by pairs.

```
> res <- postHocTest(data = dataset.means, algorithms = 3:10, test = 'friedman'
+                    correct = 'finner' , control = 'FrogCOL')
```

| FrogCOL | FrogMIS | FruitFly | Ikeda | Rand1 | Rand2 | Shukla | Turau |
|---|---|---|---|---|---|---|---|
| **132.6** | 125.5 | 105.9* | 116.1* | 116.3* | 116.6* | 117.1* | 116.2* |

**Table 2:** Rendering of a LaTeX table generated with the `writeTabular` function.

The table with the results can be generated as follows (the result can be seen in Table 2):

```
> best.res <- res$summary == max(res$summary)
> stat.diff <- res$corrected.pval < 0.05
> stat.diff[is.na(stat.diff)] <- FALSE
> writeTabular(table = res$summary, format = 'f', bold = best.res, mark = stat.diff,
+              digits = 1)
```

Finally, for the third example we will use the same dataset from Blum et al. (2015). In this example we want to compare the algorithms separately for every value of *Radius* given a fixed *Size*. The functions in **scmamp** can be also used for these kinds of comparisons. In this case we will use the Wilcoxon test with the *p-values* corrected using Finner's method and, then, a LaTeX table will be generated. In this table the best results will be highlighted in bold font and those without significant differences will be identified with a superscript.

First, filter the data.

```
> sub.dataset <- filterData(data = dataset, condition = 'Size==1000'
+                           remove.cols = 'Size')
```

Now, run the comparison.

```
> res <- postHocTest(data = sub.dataset, group.by = 'Radius', test = 'wilcoxon'
+                    correct = 'finner', control = 'FrogCOL')
> res$corrected.pval[1:5,1:6]

  Radius FrogCOL     FrogMIS     FruitFly       Ikeda       Rand1
1  0.049      NA 6.07021e-05 0.0000607021 6.07021e-05 6.07021e-05
2  0.058      NA 6.07021e-05 0.0000607021 6.07021e-05 6.07021e-05
3  0.067      NA 6.07021e-05 0.0000607021 6.07021e-05 6.07021e-05
4  0.076      NA 6.07021e-05 0.0071920496 6.07021e-05 6.07021e-05
5  0.085      NA 6.07021e-05 0.0005028761 6.07021e-05 6.07021e-05
```

Finally, generate the table. The boolean matrices needed for highlighting the results can be created manually or using the `booleanMatrix` function included in the package (the result is rendered in Table 3).

```
> tab <- res$summary
> best.res <- booleanMatrix(data = tab[, -1], find = 'max', by = 'row')
> best.res <- cbind(FALSE, best.res)
> no.diff <- booleanMatrix(data = res$corrected.pval[, -1], find = 'gt' , th = 0.05)
> no.diff <- cbind(FALSE, no.diff)
> no.diff[is.na(no.diff)] <- FALSE
> digits <- c(3, rep(1, 8))
> writeTabular(table = tab, format = 'f', bold = best.res, mark = no.diff,
+              hrule = 0, vrule = 1, print.row.names = FALSE, digits = digits)
```

## Conclusions

The **scmamp** package has been designed with the goal of simplifying the statistical analysis of the results obtained in comparisons of algorithms in multiple problems. With a few lines of code, the users can load and analyse the data and format the result for publication. This document is a brief introduction to the package. For further details on the use of the functions the reader is referred to the documentation and, particularly, to the vignettes of the package.

| Radius | FrogCOL | FrogMIS | FruitFly | Ikeda | Rand1 | Rand2 | Shukla | Turau |
|---|---|---|---|---|---|---|---|---|
| 0.049 | **247.7** | 227.6 | 226.3 | 213.2 | 212.7 | 214.5 | 212.4 | 211.9 |
| 0.058 | **189.9** | 176.9 | 174.9 | 162.0 | 161.5 | 163.5 | 162.9 | 162.7 |
| 0.067 | **151.9** | 140.6 | 142.2 | 130.4 | 129.8 | 129.6 | 130.8 | 129.9 |
| 0.076 | **122.9** | 114.1 | 117.8 | 104.6 | 105.3 | 104.9 | 105.3 | 105.3 |
| 0.085 | **102.4** | 94.3 | 99.4 | 85.9 | 86.7 | 87.1 | 87.4 | 86.7 |
| 0.094 | 85.5 | 79.3 | **85.8** | 72.9 | 72.6 | 72.3 | 74.2 | 72.9 |
| 0.103 | 74.2 | 68.1 | **75.6*** | 62.8 | 63.2 | 62.3 | 63.2 | 62.1 |
| 0.112 | 64.4 | 58.2 | **66.9** | 54.1 | 54.2 | 54.4 | 54.5 | 54.8 |
| 0.121 | 56.5 | 51.3 | **59.5** | 47.4 | 47.0 | 47.6 | 48.0 | 47.2 |
| 0.134 | **47.5** | 43.1 | 24.3 | 40.1 | 39.9 | 40.1 | 40.8 | 40.1 |

**Table 3:** Another rendering of a LaTeX table generated with the `writeTabular` function.

## Bibliography

J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesus, S. Ventura, J. M. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas, J. C. Fernández, and F. Herrera. KEEL: A software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2008. [p248]

B. Bergmann and G. Hommel. Improvements of general multiple test procedures for redundant systems of hypotheses. In *Multiple Hypotheses Testing*, volume 70, pages 100–115. Springer Berlin Heidelberg, 1988. [p248]

C. Blum, B. Calvo, and M. J. Blesa. FrogCOL and FrogMIS: New decentralized algorithms for finding large independent sets in graphs. *Swarm Intelligence*, 9:205–227, 2015. [p248, 252, 254]

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006. [p248, 249]

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937. [p248]

S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064, 2010. [p248, 249]

S. García and F. Herrera. An extension on 'Statistical comparisons of classifiers over multiple data sets' for all pairwise comparisons. *Journal of Machine Learning Research*, 9:2677–2694, 2008. [p248, 249, 250]

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6: 65–70, 1979. [p248]

J. A. Parejo, J. García, A. Ruiz-Cortés, and J. C. Riquelme. STATService: Herramienta de análisis estadístico como soporte para la investigación con metaheurísticas. In *Actas del VIII Congreso Expañol sobre Metaheurísticas, Algoritmos Evolutivos y Bio-inspirados*, 2012. [p248]

I. Rodríguez-Fdez, A. Canosa, M. Mucientes, and A. Bugarín. STAC: A web platform for the comparison of algorithms using statistical tests. In *Proceedings of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2015. [p248]

G. Santafé, I. Inza, and J. A. Lozano. Dealing with the evaluation of supervised classification algorithms. *Artificial Intelligence Review*, 44(4):467–508, 2015. [p248, 249]

*Borja Calvo*
*Department of Computer Science and Artificial Intelligence*
*University of the Basque Country (UPV/EHU)*
*Manuel de Lardizabal, 1*
*20018, Donostia (Spain)*
borja.calvo@ehu.eus

*Guzmán Santafé*
*Department of Statistics and Operations Research*
*Public University of Navarre*

*Campus de Arrosadía*
*31006, Pamplona (Spain)*
guzman.santafe@unavarra.es