# STAC: a web platform for the comparison of algorithms using statistical tests

Ismael Rodríguez-Fdez, Adrián Canosa, Manuel Mucientes and Alberto Bugarín
Centro de Investigación en Tecnoloxías da Información (CITIUS)
Universidade de Santiago de Compostela
ismael.rodriguez@usc.es, adrian.canosa@rai.usc.es, {manuel.mucientes, alberto.bugarin.diz}@usc.es

***Abstract*—One of the most suited techniques for comparing results obtained from computational intelligence algorithms is the statistical hypothesis testing. This method can be used to contrast if the difference between the algorithm with the best results and other algorithms is actually significant. In this paper, we present STAC (Statistical Tests for Algorithms Comparison), a new platform for statistical analysis to verify the results obtained from computational intelligence algorithms. STAC consists of three different layers for performing statistical tests: a Python library, a set of web services and a web client. We show several use cases, in which both non-expert and expert users interact with the web client and use the web services in different programming languages.**

***Index Terms*—Statistical analysis, soft computing sofware**

## I. Introduction

The validation of the results obtained from new computational intelligence algorithms is not a simple task. The well-known "no free lunch" theorem [1] states that a high performance over one class of problems usually causes a similar negative effect over another class. Thus, it is expected not to have a clear winner when comparing different algorithms over a sufficiently large number of data sets. Therefore, there is a growing interest in a systematic way of deciding whether an algorithm performs better than another one [2], [3].

One of the most suited techniques for comparing results of an experiment is the statistical hypothesis testing [4], [5]. This is a method of statistical inference for a hypothesis that is testable on the basis of observing a process that is modeled via a set of random variables. The final objective is to determine if a sample of results supports a certain hypothesis and whether the conclusions achieved can be generalized beyond what was tested.

In the field of computational intelligence, this method can be used to contrast if the difference between the algorithm with the best results and other alternatives is significant. For example, a statistical test can be used in order to determine the best learning algorithm when obtaining a fuzzy rule system using different approaches like ad-hoc methods, genetic algorithms or neuro-fuzzy systems. For that, usually a two-step approach is applied [6], [7]. Firstly, the hypothesis that the results obtained by the algorithms are significantly different is contrasted. This step usually involves generating a ranking of the algorithms, from the best approach to the worst. Then, if more than two algorithms are compared, a post-hoc process is used in order to contrast the difference between each pair of algorithms, controlling the family-wise error rate. With this method, the best algorithm can be compared with each of the other algorithms searching for significant differences in the results.

Several papers [8], [9], [10], [11] have studied the methodology for conducting comparisons among various computational intelligence algorithms through statistical tests. However, currently there is still a lack of rigorous validation of the results and conclusions obtained [12]. Moreover, the complexity and variety of tests that can be applied in different situations becomes a great difficulty for extending their use mostly for non-expert users. For this reason, there is a need for software tools that assist the process of comparing computational intelligence algorithms using statistical tests [13].

Nowadays, there exist different software alternatives that perform statistical tests and can be used for comparing algorithms. They can be divided into two main categories: software packages that can be used as a library in a particular programming language, and software clients that have a graphical user interface.

Python and R have become the two main programming languages for data analysis. Both of them have libraries for doing statistical tests in an easy way. On the one hand, R contains the `stats` package that provides functions for statistical calculations and random numbers generation, as well as the most useful statistical tests. On the other hand, the SciPy library is an open-source Python library that provides many user-friendly and efficient numerical routines. The module `scipy.stats` contains a large number of probability distributions as well as a growing library of statistical functions which contain several statistical tests. More recently, an open-source Java library that contains 40 non-parametric tests has been made available [14].

In the recent years, some software clients have been presented. KEEL [15] has a module for non-parametric testing through its desktop client that includes both ranking tests and post-hoc analysis. In [13] a web client for statistical analysis is presented. Furthermore, the statistical tests are also available through XML web services described using WSDL (Web Services Description Language).

In this paper, we present STAC (Statistical Tests for Algorithms Comparison), a new platform for statistical analysis for verifying the results obtained from computational intelligence algorithms. The platform consists of three different layers for

doing statistical tests: a Python library, a set of web services and a web client. Each of the layers is publicly accessible from the STAC webpage http://tec.citius.usc.es/stac/, and the web services can be used by any programming language or platform using the API.

The main contributions of STAC are: i) to provide with the best suited statistical tests for comparing computational intelligence algorithms with a considerable flexibility in its use, ii) to guide the user in each step of the testing process, focusing on when to use each statistical test, and iii) to ease the utilization for a variety of users, from non-experts that use statistical analysis for the first time, to experts that need to automate the process of testing.

This paper is structured as follows. First, section II gives a short introduction to the use of statistical tests in computational intelligence. The details of the platform and each layer are presented in section III, and section IV describes some use cases of both the client and the web services. Finally, section V presents the conclusions of this work.

## II. Statistical Tests for Results Validation in Computational Intelligence

A common task in computational intelligence is the comparison of the results of a set of algorithms. Usually, the question to be answered is: *Which of the following algorithms is the best for solving a problem?* To answer this question, the algorithms are executed on a set of problem related data sets. Then, a specific measure is used to evaluate the results. When no algorithm clearly outperforms the others, a statistical test helps to answer the question.

When a statistical test is used for this purpose, the results obtained for each algorithm are interpreted as a probability distribution. Then, the question to be answered becomes *Are the distribution of the algorithms performances the same?* Therefore, the most suited statistical tests for this task are those that contrast this hypothesis [4], [5].

Two types of statistical tests can be applied in this problem: parametric and non-parametric tests. The differences between them are that the parametric tests presuppose that the results of the algorithms are independent, follow a particular distribution (usually the normal distribution), and their deviations are similar (homocedasticity). The first condition depends on the experimental setup, but the other two can be tested using also statistical tests. If these conditions are fulfilled, then a parametric test can be applied. These tests are recommended as they have more statistical power.

In both cases, different tests are available for two groups (e.g. two algorithms) or for multiple groups. Moreover, the groups can be paired or unpaired. This means that the samples of each algorithm are related (e.g. results over the same data sets) or not (e.g. results obtained from different users).

When a multiple comparison is performed, the statistical test only contrasts if there are at least two algorithms with a different distribution. Thus, a post-hoc analysis is needed to contrast if the difference between any pair of algorithms is significant, controlling the family wise error rate [5]. Moreover, this can be done in two different ways: using a control method, i.e., one algorithm versus the others, or comparing all possible pairs of algorithms.

In STAC there are several useful tests for each situation (table I). These tests do not represent the whole set of statistical tests, but are the most commonly used in the literature [7]. A detailed explanation of these tests can be found in [4], [5], [7].

## III. STAC platform

The STAC platform consists of three different layers: a Python library, a list of web services and a web client. Fig. 1 shows the architecture of STAC with the different parts and the connections between them.

The Python library contains an implementation of the statistical tests. This implementation is used directly by a list of web services to facilitate the use of the tests from any programming language. The web services are implemented using the REST (Representational State Transfer) model, thus they can be called using the HTTP protocol. Moreover, the data used to communicate with the web services is described using JSON (Javascript Object Notation) in order to maintain simplicity. Finally, the web client uses AJAX (Asynchronous Javascript And XML) to perform the statistical tests using the web services.

In the following subsections, each of the layers is described in depth.

### A. Python Library

The first layer of STAC is an implementation in Python of several statistical tests. Only those tests not currently implemented in the `scipy.stats` module are included in this library. Moreover, statistical tests for multiple comparison (ANOVA, Friedman, etc.) are reimplemented in STAC. This is due to the lack of information returned by the corresponding implementation in `scipy.stats`, which is needed in order to do the post-hoc analysis. Table I shows the list of tests implemented for each module in the Python library, including those imported from *scipy.stats*.

The function signature (i.e. inputs and outputs) for each test follows the `scipy.stats` model. Thus, only the sample for each algorithm is needed for performing the test. Also, the functions return all the valuable information obtained in the statistical test: the calculated statistics, the p-value, and the information required to do a post-hoc analysis if necessary. Table II summarizes the signatures of all the tests[1].

### B. Web Services

In order to make accessible the use of the statistical tests to several programming languages, a list of web services are provided within STAC. These services are both for the tests of the Python library (sec. III-A), and those implemented in `scipy.stats`.

[1]More detailed information about the Python library and its use is available in http://tec.citius.usc.es/stac/doc/.

Table I
TESTS AVAILABLE IN STAC

| Type of test | Implemented Tests | Python module |
|---|---|---|
| Normality | Shapiro-Wilks, Kolmogorov-Smirnov, D'Agostino-Pearson | scipy.stats |
| homocedasticity | Levene | scipy.stats |
| Parametric comparison between 2 | t-test independent, t-test related | scipy.stats |
| Parametric multiple comparison | ANOVA between cases, ANOVA within cases | stac.parametric |
| Parametric post-hoc | Bonferroni | stac.parametric |
| Non-parametric comparison between 2 | Wilcoxon, Mann-Whitney-U | scipy.stats |
| Non-parametric multiple comparison | Friedman, Friedman Aligned Ranks, Quade | stac.nonparametric |
| Non-parametric post-hoc 1 vs. All | Bonferroni-Dunn, Holm, Finner, Hochberg, Li | stac.nonparametric |
| Non-parametric post-hoc All vs. All | Nemenyi, Holm, Finner, Hochberg, Shaffer | stac.nonparametric |

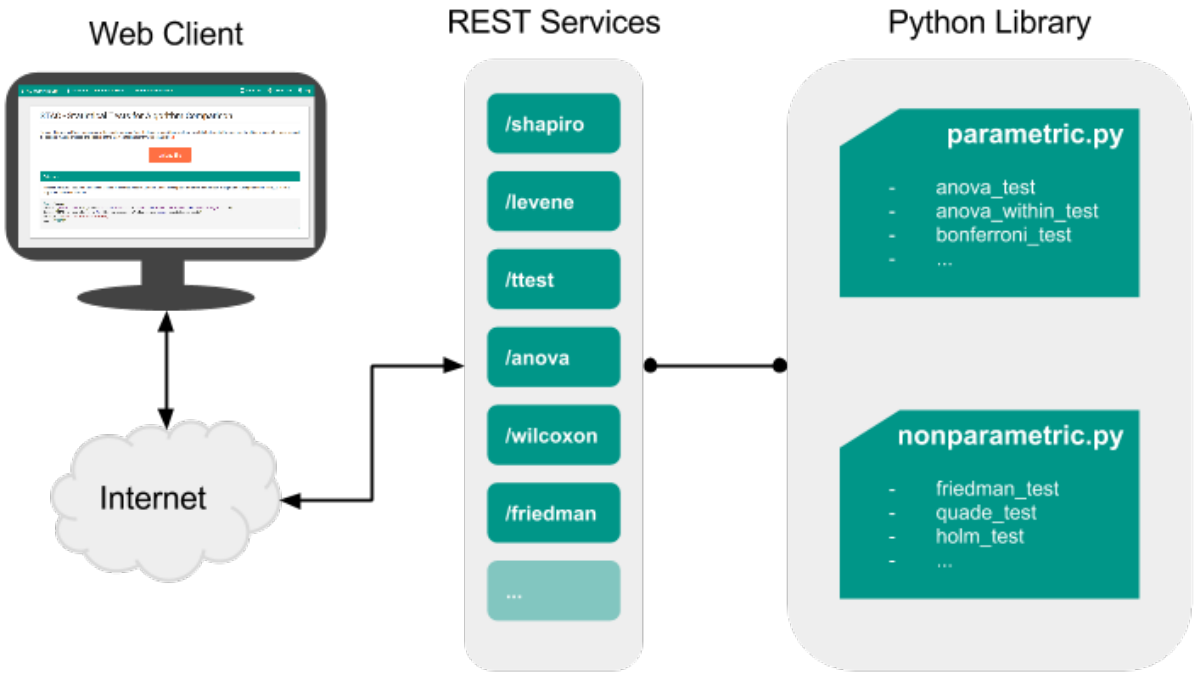


Figure 1. STAC platform architecture.

Table II
STAC PYTHON LIBRARY SIGNATURES

| Type of Test | Function Signature |
|---|---|
| Parametric multiple comparison | (alg1, alg2, ...) → (statistic, p-values, pivots) |
| Non-parametric multiple comparison | (alg1, alg2, ...) → (statistic, p-values, rankings, pivots) |
| Non-parametric post-hoc 1 vs. All | (pivots, control) → (comparisons, statistics, p-values, adjusted p-values) |
| Non-parametric post-hoc All vs. All | (pivots) → (comparisons, statistics, p-values, adjusted p-values) |

The web services are implemented following the REST architectural style:

- Each service is fully described by its URL. Only the data sample values are sent through the body of the request. This is because it is impractical to send variable length data through the URL. It makes it unreadable and unusable in many browsers and frameworks.
- The communication is done using a HTTP request without state. Each service does a single statistical test using only the information given by the request.

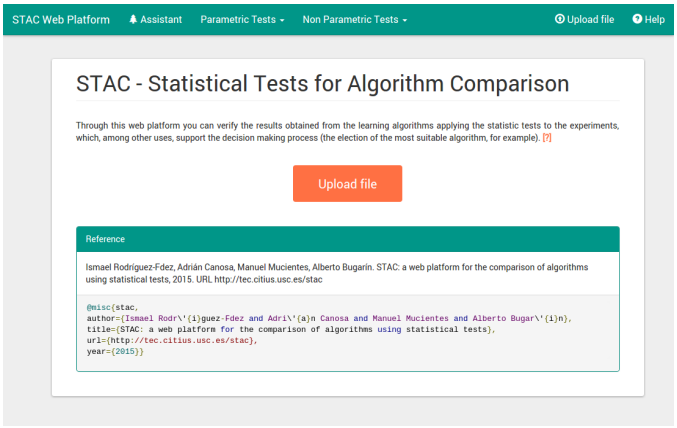


Figure 2. STAC web platform main page.

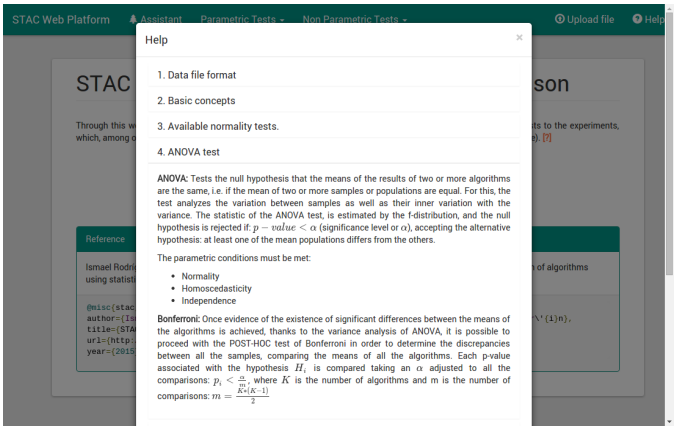


Figure 3. STAC web platform help.

- The POST HTTP operation is used for the request message. This is due to the need of sending data through the body.
- The data of the request and the response are codified using an Internet media type, in this case JSON.

The URL definition for each web service depends on the type of statistical test performed. Moreover, when a ranking test is applied, the post-hoc method is also a URL parameter. Table III shows how the request message can be constructed using these resources: it describes how to construct the URL to call the service and what is expected from the body of the request and the results obtained from it. When a parameter is surrounded by < > means that it has to be substituted by a particular value.

For the `test` and `post-hoc` parameters, Table IV can be used to select the particular value. Furthermore, `control` indicates the case (or algorithm) that is going to be the control method in a 1 vs. All post-hoc procedure. The body is built using JSON with only one parameter: `values`. This parameter contains a dictionary of cases whose key is the name of the case and the value is the list of samples for this case. Finally, the result is sent in another JSON that contains the result of the statistical test: statistics, p-values, and the result of the hypotheses[2].

### C. Web Client

The last layer is a web-based front-end of the STAC web services API. This web client is designed to facilitate the full process of statistical analysis for those users without any knowledge of the web services or the Python library. Moreover, this platform contains an assistant to decide the best suitable test for the data provided by the user, and a continuous help in each step of the process.

The website is designed to be user-friendly and efficient. Fig. 2 shows the index page of the STAC web platform. The top bar contains all the actions that can be done by the user. Each step of the process displays simple, complete and precise information to guide the user. Furthermore, a help link is always available in both the top bar and the text content of the web (Fig. 3). This help does not imply a slow down in the workflow execution by expert users, since all the tests are easily accessible from the top menu.

The assistant provided by the web platform can also be accessed from the top bar. This process takes into account the following data:

- The number of groups available ($k$), i.e., the number of cases or algorithms used.
- The number of samples per group ($n$).
- If the groups are paired, i.e., the samples of each group are related. A typical example in computational intelligence is when different algorithms are applied to the same data sets.
- The normality of each group (tested using a Shapiro-Wilks test with alpha 0.1).
- The homocedasticity between groups (tested using Levene test with alpha 0.1).

From these data values, a simple decision tree is implemented in order to select the statistical test that best fits to the data provided by the user. Figure 4 shows the implemented decision tree. The criteria used in this decision process follows the recommendations described in [7]. This tree, with the data used and the branches selected, is shown to the user when the assistant process is selected. Thus, the user always knows why a particular statistical test has been selected.

The different parameters of the selected test are shown with the default values, which can be modified by the user (Fig. 5). Then, when the test is applied, the results are shown in a table below. This table contains all the information valuable to perform the statistical analysis and the result of the hypothesis contrast. This information can also be exported to CSV (comma separated values) or LaTeX, in order to facilitate its use in reports and scientific papers. Finally, each test has a list of references in the bottom of the page.

[2]A full description of the STAC web services is available at http://tec.citius.usc.es/stac/apidoc/.

Table III
STAC WEB SERVICES DESCRIPTION

| Type of Test | URL | Body | Result |
|---|---|---|---|
| Comparison between 2 | `/<test>` | `{values:{<case1>,<case2>}}` | statistic, p-value |
| Parametric multiple comparison | `/<test>` | `{values:{<case1>,...}}` | statistic, p-value, comparisons, p-values |
| Non-parametric multiple comparison and Post-hoc 1 vs. All | `/<test>/<post-hoc>/<control>` | `{values:{<case1>,...}}` | statistic, p-value, comparisons, p-values |
| Non-parametric multiple comparison and Post-hoc All vs. All | `/<test>/<post-hoc>` | `{values:{<case1>,...}}` | statistic, p-value, comparisons, p-values |

Table IV
STAC WEB SERVICES RESOURCES

| Resource | Type of test | Resource available |
|---|---|---|
| `<test>` | Comparison between 2 | ttest, ttest_rel, wilcoxon, mannwhitneyu |
| | Parametric multiple comparison | anova, anova_within |
| | Non-parametric multiple comparison | friedman, friedman_aligned_ranks, quake |
| `<post-hoc>` | Post-hoc 1 vs. All | bonferroni, holm, finner, hochberg,li |
| | Post-hoc All vs All | nemenyi, holm, finner, hochberg, shaffer |

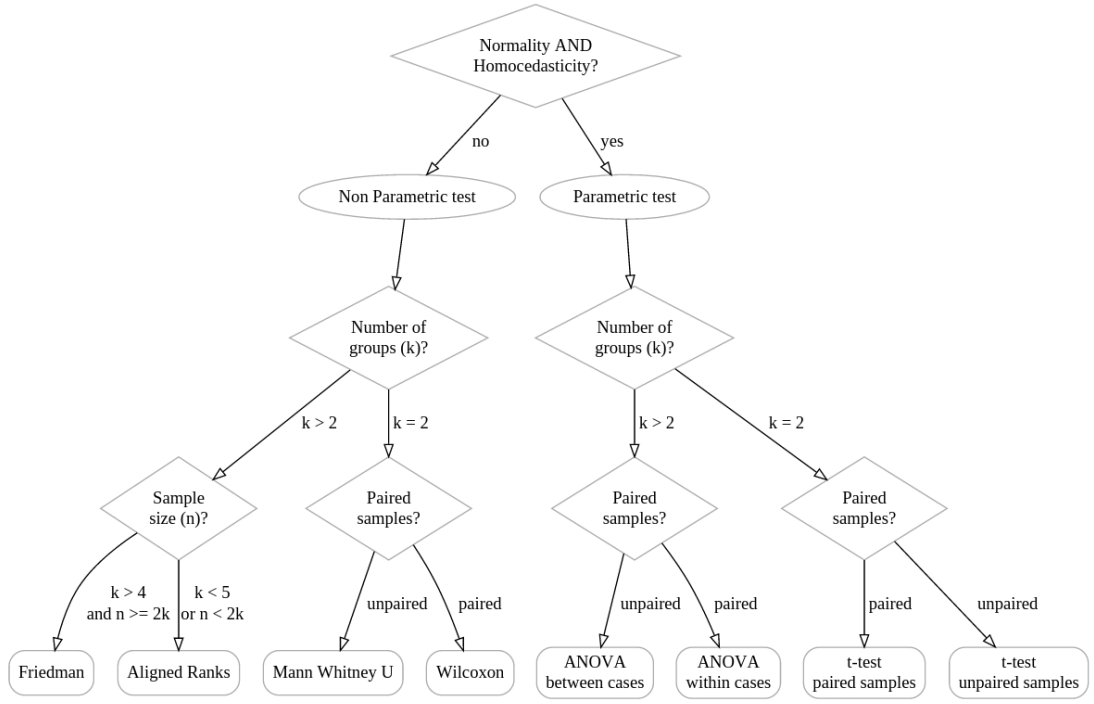


Figure 4. STAC assistant decision tree.

## IV. EXAMPLE OF USE OF STAC AND REAL USE CASES

This section describes an example of use of STAC that shows users how to proceed with the tool and several real use cases in a number of application fields. We firstly focus on the description of the web client and the web services. The Python library is not commented, since its use is very similar to the `scipy.stats` module.

### A. Use of the web client

The STAC web platform was developed to help both experts and non-experts in the field of algorithms analysis. The typical workflow a new user has to follow consists of several independent steps (which may be skipped by expert users):

1) Upload the file that contains the data (Fig. 6). This can be done directly from the home page. A help content

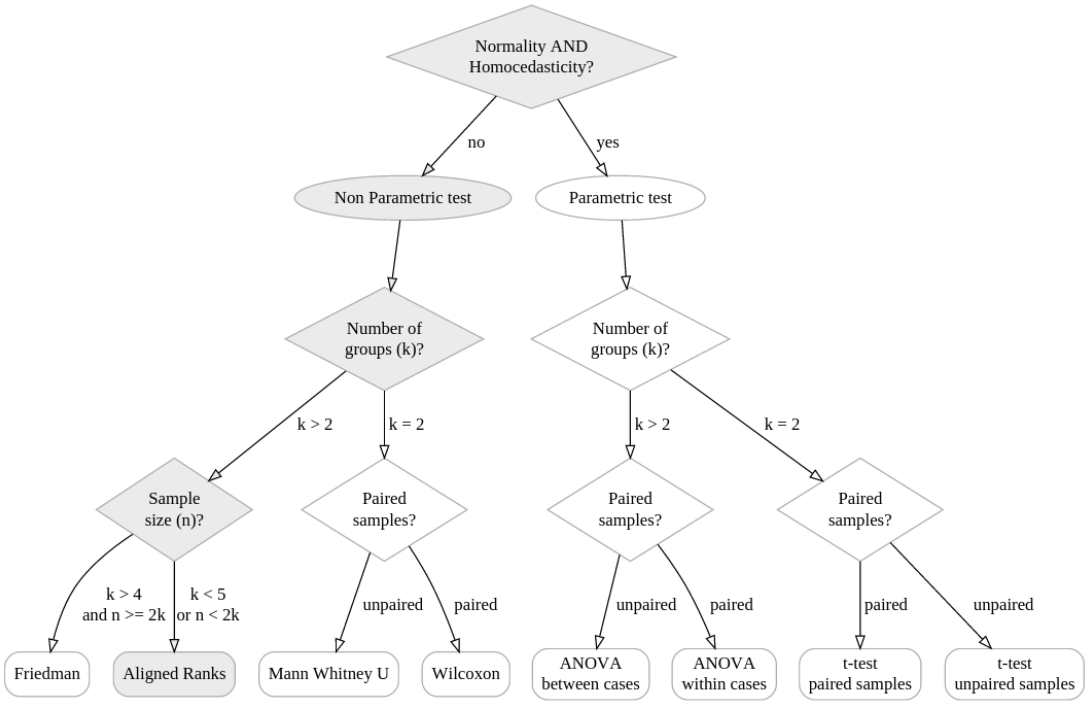


Figure 7. Selection of the statistical test and its parameters.

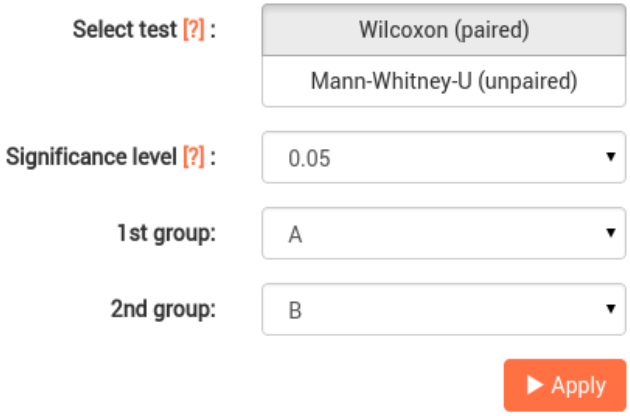


Figure 5. STAC web platform test application.

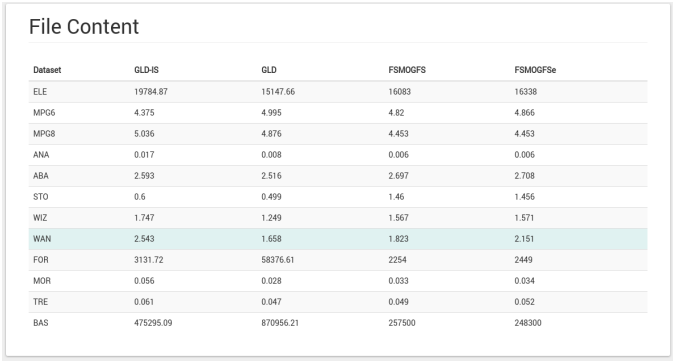


Figure 6. Content of the uploaded example file.

is available to select the file format. In what follows, we will use as example a comparison of the test error of four genetic fuzzy systems taken from [16]. Figure 6 shows the test errors of the genetic fuzzy systems for each of the twelve datasets.

2) The user can start the assistant process if not sure about the type of test to be applied (Fig. 7).
3) Once the statistical test has been selected, the user has to give the values of the parameters. Fig. 8 shows the selection of the Friedman test with Holm post-hoc and a significance level of 0.05.
4) When clicking on the *Apply* button, the results are shown in a table. Following with the example, Fig. 9 displays a table with the contrast hypothesis of the Friedman test. Then, the following table indicates the ranking of the algorithms and, finally, the last table summarizes the pairwise comparisons performed using the post-hoc method.
5) Finally, the results table can be exported to LaTeX(Fig. 10).

At any moment of the process, the user can access the help content, and have access to further information about the test and the implications of its use.

### B. Use of the web services

The STAC web services can be used in multiple programming languages. In fact, the web client uses

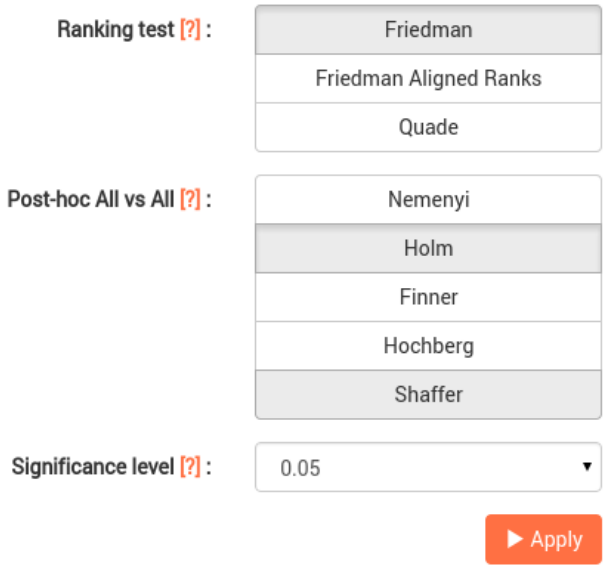


Figure 8. Statistical test performed together with its results.

Javascript to call the services in order to perform the statistical tests. This can be done easily using the jQuery library. [3] Figure 11 shows a code example to use the web service `/wilcoxon` from javascript through an AJAX call. Each element of the request (url, header, body...) is passed to the call function, and a callback is implemented when the call is successful and returns the result.

In the same way, Fig. 12 shows how to use the web services from Java. In particular, the unirest library[4] was used in order to show the simplicity in the use of REST services. In this case, the method blocks the flow of the program, thus there is no need of a callback function. Thus, any user can implement a simple program to perform statistical tests in a few lines.

Finally, another use case employs the web services to perform statistical analysis directly on a spreadsheet. We selected Google Sheets[5] due to its increasing use and popularity. In order to apply the statistical analysis through STAC web services, we codify a Google Script. An example for the Wilcoxon test is shown in Fig. 13. This code can be called as a spreadsheet function, and the result is printed in the same spreadsheet using the nearby cells (Fig. 14).

### C. Real use cases

The STAC platform has already been used in a number of research experiments both in the computational intelligence and machine learning fields. Each of them present a different scenario with different needs from the algorithms analysis point of view, that have been very satisfactorily covered by STAC.

In [17] a statistical analysis was used to compare the time needed to assess the students in e-learning with two different

[3] jQuery is a fast, small, and feature-rich JavaScript library (http://jquery.com/).

[4] unirest is a simplified, lightweight HTTP client library available in multiple languages (http://unirest.io).

[5] Google Sheets is an on-line spreadsheet application that allows to create and format spreadsheets, also in a collaborative way (https://docs.google.com/spreadsheets/).

Friedman test (significance level of 0.05)

| Statistic | p-value | Result |
|---|---|---|
| 2.46939 | 0.07919 | H0 is accepted |

Ranking

| Rank | Algorithm |
|---|---|
| 2.08333 | GLD |
| 2.08333 | FSMOGFS |
| 2.58333 | FSMOGFSe |
| 3.25000 | GLD-IS |

Friedman test (significance level of 0.05)

| Comparison | Statistic | Adjusted p-value | Result |
|---|---|---|---|
| GLD-IS vs GLD | 2.21359 | 0.16114 | H0 is accepted |
| GLD-IS vs FSMOGFS | 2.21359 | 0.16114 | H0 is accepted |
| GLD-IS vs FSMOGFSe | 1.26491 | 0.82361 | H0 is accepted |
| GLD vs FSMOGFSe | 0.94868 | 1.00000 | H0 is accepted |
| FSMOGFS vs FSMOGFSe | 0.94868 | 1.00000 | H0 is accepted |
| GLD vs FSMOGFS | 0.00000 | 1.00000 | H0 is accepted |

Figure 9. Results of the statistical test.

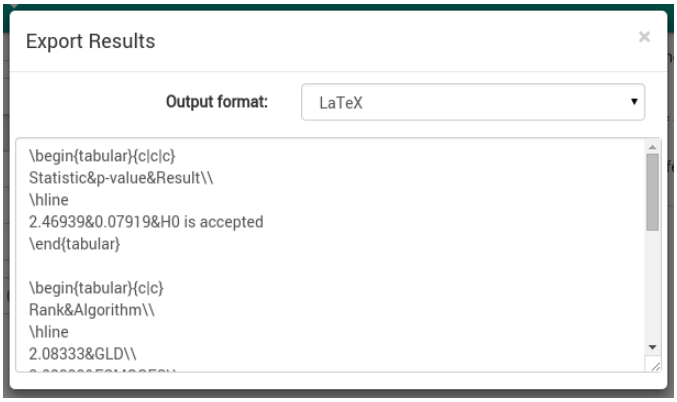


Figure 10. Result exported in LaTeXformat.

```
$.ajax({
   url: "http://tec.citius.usc.es/",
   type: "POST", dataType: "json", contentType: "application/json",
   data: JSON.stringify({values: {group1: data1, group2: data2}}),
   success : function(data) {
      $("#result").html(JSON.parse(data)).show();
   },
});
```

Figure 11. STAC web service call from Javascript.

```
HashMap<String, double[]> values = new HashMap<>();
values.put("A", new double[]{1, 2, 3, 5});
values.put("B", new double[]{3, 2, 1, 1});

JSONObject response = Unirest.post("http://tec.citius.usc.es/stac/api/wilcoxon")
        .header("Content-Type", "application/json")
        .body("{\"values\": " + JSONObject.valueToString(values) + "}")
        .asJson()
        .getBody().getObject();

response.get("p_value");
```

Figure 12. STAC web service call from Java.

```
function wilcoxon(data) {
  /* Transform data to the format accepted by STAC */
  var url = 'http://tec.citius.usc.es/stac/api/wilcoxon';
  var options = {
    'method': 'post',
    'contentType': 'application/json',
    'payload': JSON.stringify(payload)
  };
  var response = UrlFetchApp.fetch(url, options);
  var result = JSON.parse(response.getContentText());

  return [["Wilcoxon Test"], ["statistic", "p-value"], [result["statistic"], result["p_value"]]];
}
```

Figure 13. STAC web service call from Google Script.

=wilcoxon(A1:B5)

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | A | B | | | |
| 2 | 1 | 3 | | | |
| 3 | 2 | 2 | | Wilcoxon Test | |
| 4 | 3 | 1 | | statistic | p-value |
| 5 | 5 | 1 | | 1.5 | 0.4142161782 |

Figure 14. Google Spreadsheet result after applying a STAC web service.

tools. Since data are not paired, i.e. the students are different for each tool, an unpaired test was needed. In this case, the Mann-Whitney-U test was used for comparing the results. [18] presents a comparison of five process mining algorithms. In order to perform a comparison between all of them, a Friedman ranking test was used together with the Holm post-hoc procedure. In the field of Linguistic Descriptions of Data, [19] presented a performance comparison of five classifiers applied to the task of automatically generating textual weather forecasts from raw meteorological data. A Friedman test with a Finner post-hoc method was used to validate the obtained results. Finally, in the field of autonomous robotics, in [20] a Friedman Aligned-Rank test is applied to compare the use of different sensors for robot localisation in crowded environments. Those sensors that show no significant differences using the Holm post-hoc procedure, are grouped together to increase the performance.

## V. Conclusions

We have presented a new platform that performs statistical analysis for the comparison of results obtained by computational intelligence algorithms. This platform has been designed with three main characteristics: i) to provide with the best suited statistical tests for comparing computational intelligence algorithms, ii) to guide the user in each step of the testing process, focusing on the selection of the appropriate statistical test, and iii) the usability for a wide variety of users, from non-experts that use statistical analysis for the first time, to experts statistical tests. The platform has three different layers for doing the statistical tests: a Python library, a set of web services and a web client. STAC has already been used in a number of papers in several research fields and by users with different degrees of expertise. Furthermore, STAC allows the use of the web services from different programming languages and also in a spreadsheet application.


## Acknowledgements

This work was supported by the Spanish Ministry of Economy and Competitiveness under projects TIN2011-22935, TIN2011-29827-C02-02 and TIN2014-56633-C3-1-R, and the Galician Ministry of Education under the projects EM2014/012 and CN2012/151. I. Rodriguez-Fdez is supported by the Spanish Ministry of Education, under the FPU national plan (AP2010-0627).


## References


[1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[2] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[3] A. P. Engelbrecht, *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[4] J. H. Zar *et al.*, *Biostatistical analysis*. Pearson Education India, 1999.

[5] D. J. Sheskin, *Handbook of parametric and nonparametric statistical procedures*. crc Press, 2003.

[6] S.-Z. Zhao and P. N. Suganthan, "Comprehensive comparison of convergence performance of optimization algorithms based on nonparametric statistical tests," in *2012 IEEE Congress on Evolutionary Computation (CEC)*, 2012, pp. 1–7.

[7] 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*, vol. 180, no. 10, pp. 2044–2064, 2010.

[8] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms," *Neural computation*, vol. 10, no. 7, pp. 1895–1923, 1998.

[9] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Springer, 2015.

[10] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[11] A. M. Palacios, L. Sanchez, and I. Couso, "Ci-lqd: A software tool for modeling and decision making with low quality data," in *Proceedings of IEEE International Conference on Fuzzy Systems 2013*, 2013, pp. 1–8.

[12] T. Bartz-Beielstein and M. Preuss, "The future of experimental research," in *Experimental Methods for the Analysis of Optimization Algorithms*. Springer, 2010, pp. 17–49.

[13] 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.

[14] J. Derrac, S. García, and F. Herrera, "Javanpst: Nonparametric statistical tests in java," *arxiv:1501.04222 [stats.CO]*, 2015.

[15] J. Alcalá-Fdez, L. Sánchez, S. García, M. J. del Jesús, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. M. Rivas *et al.*, "Keel: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing*, vol. 13, no. 3, pp. 307–318, 2009.

[16] I. Rodríguez-Fdez, M. Mucientes, and A. Bugarín, "An instance selection algorithm for regression and its application in variance reduction," in *Proceedings of the 2013 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2013.

[17] A. Rodriguez Groba, B. Vazquez Barreiros, M. Lama, A. Gewerc, and M. Mucientes, "Using a learning analytics tool for evaluation in self-regulated learning," in *Frontiers in Education Conference (FIE), 2014 IEEE*. IEEE, 2014, pp. 1–8.

[18] B. Vázquez-Barreiros, M. Mucientes, and M. Lama, "Prodigen: Mining complete, precise and minimal structure process models with a genetic algorithm," *Information Sciences*, vol. 294, pp. 315–333, 2015.

[19] J. Janeiro, I. Rodríguez-Fdez, A. Ramos-Soto, and A. Bugarín, "Data mining for automatic linguistic description of data," in *7th International Conference on Agents and Artificial Intelligence*, 2015, pp. 556–562.

[20] A. Canedo-Rodríguez, V. Álvarez Santos, C. Regueiro, R. Iglesias, S. Barro, and J. Presedo, "Particle filter robot localisation through robust fusion of laser, wifi, compass, and a network of external cameras," *Information Fusion*, Accepted, 2015.