# Interactive comparison of hypothesis tests for statistical model checking

Pieter-Tjerk de Boer
University of Twente
Enschede, The Netherlands
p.t.deboer@utwente.nl

Daniël Reijsbergen
University of Edinburgh
Edinburgh, Scotland
dreijsbe@inf.ed.ac.uk

Werner Scheinhardt
University of Twente
Enschede, The Netherlands
w.r.w.scheinhardt@utwente.nl

## ABSTRACT

We present a web-based interactive comparison of hypothesis tests as are used in statistical model checking, providing users and tool developers with more insight into their characteristics. Parameters can be modified easily and their influence is visualized in real time; an integrated simulation engine further illustrates the behaviour of the tests. Finally, since the source code is available, it can serve as a framework in which newly developed tests can be tried.

## Categories and Subject Descriptors

Mathematics of computing [**Probability and statistics**]: Probabilistic inference problems—*Hypothesis testing and confidence interval computation*

## General Terms

Performance, verification

## Keywords

Statistical model checking, hypothesis testing

## 1. INTRODUCTION

Statistical model checking (SMC) is increasingly seen as a powerful alternative to numerical model checking, as witnessed by its implementation in many tools (such as UPPAAL, PRISM and MRMC) and libraries (such as COSMOS and PLASMA) and many publications (such as [1, 5, 10, 11]). The core idea is to repeatedly simulate the behaviour of a stochastic system model, checking each time whether some event of interest happens, such as the (in)validity of a formula in a temporal logic such as PCTL or CSL. The results are then used to draw a conclusion regarding whether the probability $p$ of that event is above or below some threshold $p_0$. This is where *hypothesis testing* comes in: based on the statistical evidence from the simulation runs, one wants to accept or reject the "hypotheses" that $p > p_0$ or that $p < p_0$, with some confidence level.

As it turns out, different tools implement different tests from the many available (cf. Tables 1 and 2). Most tools do not allow the user to choose a test, which may lead to the misconception that one test is the only correct way of drawing conclusions in SMC. The same happens in many research papers, which may partly explain the popularity of the SPRT: it is the one used in the first paper on statistical model checking [11]. Furthermore, without understanding a test's properties, parameterizing it correctly is hard, yet all tools allow the user to set parameters manually. Since the tests have fundamentally different properties (cf. Section 2 of this paper), all of this may cause unclarity for the tool users, and misinterpretation of the results.

In previous work [6], the authors have given an overview of different hypothesis tests for SMC and their characteristics. Although that overview should help users to select the right test and the right tool for their problem, we feel a static explanation of the different tests does not do the topic justice, since each test depends on parameters that strongly influence both how many samples are needed and the interpretation of the outcome. Therefore, we have developed a web-based tool in which the user can easily try different parameter settings and see graphically how these influence the tests' decision boundaries. Further understanding is provided by also (optionally) running example simulations and seeing what decisions the different tests take. It[1] can be found at `http://wwwhome.ewi.utwente.nl/~ptdeboer/hyptest-for-smc/`.

In a sense, our tool is a "meta-tool": it does not by itself answer qualitative or quantitative questions about system models, but it helps users better understand what the actual statistical model checking tools do, and thus choosing the right one. Also, our tool helps users to parameterize the tests used in real tools, and can be a testbed for newly developed tests.

We start this paper by giving some background information on statistical model checking and hypothesis testing in Section 2, followed by describing the tool web page in Section 3, and illustrating its use in Section 4. Section 5 contains a note about the use of web-technologies for tool development, and Section 6 provides conclusions.

[1]The tool originally started out as a companion website to [6] and thus is already mentioned there, but has since been enhanced with new features, such as the simulator.

VALUETOOLS 2015, December 14-16, Berlin, Germany

DOI 10.4108/eai.14-12-2015.2262571

| Test | Class | UPPAAL (< 4.1.15) | UPPAAL (≥ 4.1.15) | PRISM | MRMC | COSMOS | PLASMA | YMER | PVeStA | APMC |
|---|---|---|---|---|---|---|---|---|---|---|
| SPRT | I | ✓ | ✓ | ✓ | | | ✓ | ✓ | | |
| Gauss-SSP | I | | | | | | | | ✓ | |
| Chernoff-SSP | I | | | | | | | | | ✓ |
| Gauss-CI | II | | | ✓ | | | | | | |
| Chow-Robbins | II | | □ | ✓ | ✓ | □ | | | □ | |
| Chernoff-CI | II | □ | | ✓ | | | □ | | | |
| Azuma | III | | | | | | | | | |
| Darling-Robbins | III | | | | | | | | | |

✓: the procedure is implemented as a hypothesis test.

□: the procedure is only implemented for making quantitative statements (i.e., estimating $p$ and a confidence interval around it, rather than deciding whether $p$ is above or below $p_0$).

**Table 1: Implementation of tests in SMC tools. Note that in many tools and papers, the tests are not given names, so some of the test names were assigned by the authors in [6].**

**SPRT** "Sequential Probability Ratio Test", draws samples until it can decide, based on a result by Wald, that a conclusion can be drawn. [9, 11]

**Gauss-SSP** "Single Sampling Plan": fixes number of samples in advance; correctness guarantees based on using Gauss distribution. [8]

**Chernoff-SSP** Same, but using Chernoff bound for correctness guarantees.

**Gauss-CI** "Confidence Interval": fixes number of samples in advance; computes confidence interval using Gauss distribution and decides based on whether $p_0$ is inside, above, or below.

**Chernoff-CI** Same, but using Chernoff bound for confidence interval computation [4].

**Chow-Robbins** Again based on Gaussian confidence interval, but number of samples is decided on the fly, based on a result by Chow and Robbins [2].

**Azuma** Class-III test, with correctness guarantee based on the so-called generalized Azuma inequality [7].

**Darling-Robbins** Same but correctness guarantee based on a result by Darling and Robbins [3, 6].

**Table 2: Brief description of the different hypothesis tests**

## 2. BACKGROUND: STATISTICAL MODEL CHECKING AND HYPOTHESIS TESTING

As explained above, in statistical model checking, the investigator simulates a system many times, checking each time whether some event of interest happens, and wants to use this evidence to conclude whether the true but unknown probability $p$ of the event is above or below some threshhold $p_0$. This conclusion should be accompanied by a statistical guarantee, typically in a form like "the risk that our procedure draws a wrong (or no) conclusion, is less than (e.g.) 5%". The 5% in this example is the allowed error probability, and its complement, 95%, is the *confidence level*.

Clearly, if the true probability $p$ is very different from the threshold value $p_0$, only few samples (simulation runs) are needed to confidently draw a conclusion about whether $p > p_0$ or $p < p_0$. On the other hand, if the true probability is close to the threshold, many samples will be needed to confidently decide on which side of $p_0$ the true $p$ lies. In our previous work [6], we identified three classes of hypothesis tests, which differ in their behaviour if $p \approx p_0$:

I. tests whose risk of drawing a *wrong conclusion* exceeds the allowed error probability if $p \approx p_0$;

II. tests whose risk of drawing *no conclusion* exceeds the allowed error probability if $p \approx p_0$;

III. tests which risk drawing a *very large number of samples* if $p \approx p_0$, but will not violate the confidence level guarantees.

Which of these is most appropriate, depends on the investigator's area of application. However, as Table 1 shows, SMC tool support for the different tests is limited; no tool offers a class-III test, and many offer only class-I or II but not both. Users are therefore likely to be unaware of the existence of different tests, and of the exact meaning of the results given by their particular SMC tool.

In mathematical terms, the problem is formulated as deciding between three hypotheses:

$$H_{+1} : p > p_0,$$
$$H_{-1} : p < p_0,$$
$$H_0 : p = p_0.$$

The latter is called the null hypothesis, and cannot be proven correct by statistical means [6]. So practically speaking, any statistical test will either accept $H_{+1}$, or accept $H_{-1}$, or say that there is not enough evidence yet to choose either of them. A statistical test can make two kinds of error: accepting a hypothesis which is not true (e.g., accepting $H_{+1}$ while actually $p \leq p_0$), or not accepting a hypothesis which is in fact true (e.g., terminating undecidedly while $p > p_0$). A practitioner typically will set an (application-dependent) upper bound on the acceptable probability of each of these

errors; in the current context, these are called $\alpha$ for errors of the first kind, and $\beta$ for errors of the second kind. Then the three classes of tests can be described as:

I. tests whose probability of drawing a *wrong* conclusion exceeds $\alpha$ when $|p-p_0| < \delta$, where $\delta$ is an "indifference level";

II. tests whose probability of drawing *no* conclusion exceeds $\beta$ when $|p - p_0| < \zeta$, where $\zeta$ is again an indifference level;

III. tests which risk drawing a *very large number of samples* if $p \approx p_0$, but will ultimately draw the correct conclusion with probability at least $1 - \alpha$.

# 3. THE TOOL

Figure 1 shows what our web-based tool, accessible at `http://wwwhome.ewi.utwente.nl/~ptdeboer/hyptest-for-smc/` looks like. It consists of four parts.

The main part is a **graph of decision boundaries**. The horizontal axis shows $N$, the number of samples gathered so far. Recall that each such sample is the result of an entire simulation run of the model checking tool. On the vertical axis is $Z_N$, which is the number of those $N$ samples in which the event of interest (of which $p$ is the unknown probability) occurred, minus $p_0 \cdot N$, which is the *expected* number of those samples if $p$ would be equal to $p_0$. Clearly, $Z_N$ has a positive drift if $p > p_0$, and negative drift if $p < p_0$. As more and more samples are gathered, $N$ increases and $Z_N$ changes as a function of the simulation results; thus, a random path is traced through the graph. Examples of such paths can be drawn as wiggly lines using the "Simulate" button, see below. When this path hits the decision boundary of the test under consideration, the test terminates and draws a conclusion based on the sign of $Z_N$; or, for class-II tests, it terminates undecidedly if the grey section of its decision boundary is hit.

Secondly, there is a part with **controls**. Sliders are used to set parameters of interest, such as the value of the decision threshold $p_0$, the confidence level and the indifference level that some tests need (cf. Section 2). As the user moves these sliders, the graph is adapted in real time.

Thirdly, the webpage has a built-in **simulator**, which generates a random trace in the $Z_N$ vs. $N$ space, using a user-controlled value of $p$, and shows this as a line in the graph. Note that one such a trace represents doing $N$ simulations in the underlying system-simulation tool (UPPAAL, PRISM, etc); instead of actually performing these system-simulation runs, they are replaced here by coin flips with some success probability $p$, which can be manipulated by the user.

This is useful in two ways. Firstly, showing a few of these lines gives the user a quick insight into what is happening, which tests take long to reach a conclusion and why, etc., and how this depends on $p$ (which is normally unknown, but in this simulation can be set by the user). Secondly, by repeating this say 1000 times and keeping track of the individual tests' decisions, one can check that the tests indeed live up to their promised statistical guarantees, and show that many of them are actually rather conservatively designed. In fact, the results from Tables 6–8 in [6] can be reproduced directly on this webpage.

Fourthly, the webpage shows some **calculated parameters** of the tests. This is useful because many existing tools require the user to set a parameter which has no clear relationship to the resulting confidence and correctness guarantees, e.g., the number of samples for the Gauss-CI test. Our web tool can be used to compute such parameters from user-specified correctness bounds.

# 4. EXAMPLES

In this section, we use the screenshot shown in Figure 1 to illustrate how the important differences between the different tests become evident on our tool. A rough description of the tests mentioned can be found in Table 2; more details can be found in [6] and in references given in the table.

As a first example, compare the Gauss-SSP and Gauss-CI tests: both are fixed-sample-size tests, meaning the number of samples needed is set in advance; therefore, their decision boundaries are vertical lines. For the same confidence level, Gauss-SSP is seen to need far fewer samples than Gauss-CI, at first glance suggesting Gauss-SSP is superior. However, this difference stems from Gauss-SSP taking a more serious risk: it is a class-I test, which risks incorrect conclusions when $p \approx p_0$, while Gauss-CI, being a class-II test, would rather risk terminating undecidedly. The different classes of tests are emphasized by their different colours in the graph.

As a second example, compare the SPRT and Gauss-SSP test. Both are class-I tests, so they only guarantee the confidence level if $p$ is at least some indifference level away from $p_0$. However, the way they achieve this is different, as is clear from the different shape of the corresponding curves. Gauss-SSP is a fixed sample size test, where the sample size $N$ is set appropriately in advance. In contrast, the SPRT is a sequential test: after each sample, it checks whether a decision can already be made, using $Z_N$ thresholds calculated appropriately. As a consequence, if the true $p$ is far from $p_0$, the SPRT typically can draw a conclusion much earlier than the Gauss-SSP, as illustrated by the wiggly lines in the graph. Only when p is really close to $p_0$, the SPRT will typically take longer.

As a third example, compare the Gauss-CI and Chow-Robbins tests, both of which are class-II tests. As seen in the figure, the Chow-Robbins test may require either more or fewer samples than the Gauss-CI test, for the same confidence level, depending on where $Z_N$ ends up; in the example sample paths shown in the figure, Chow-Robbins decides slightly later than Gauss-CI. As one interactively modifies $p_0$ however, it becomes apparent that at $p_0 = 0.50$, the decision boundary of the Chow-Robbins test is such that it will always terminate earlier than Gauss-CI. Clearly, this is useful information for choosing between the two tests.

Finally, take a look at the simulation results at the bottom of the figure. In this case, we chose to simulate for $p = p_0 + \delta$, i.e., $p$ was just at the border of the indifference region. Furthermore, the acceptable probability of errors of first and second kind was set to $\alpha = \beta = 5\%$. Indeed, the table shows that the class-I tests (SPRT and Gauss-SSP) have about 5% probability of drawing the wrong conclusion, and that some class-II tests (Gauss-CI and Chow-Robbins) also have about 5% probability of terminating undecidedly; the third class-II test, Chernoff-CI, is seen to be rather conservative test with much less than 5% probability of terminating undecidedly. Finally, the two class-III tests (Azuma and Darling) terminate mostly undecidedly here; that is because the simulations were stopped after about 22000 samples (the right edge of the graph), which for these tests was not enough

to draw a conclusion. Expanding the $N$ range shows them to draw the correct conclusion in 100% of the simulations, at an average $N$ of around 40000.

# 5. HTML5 AND JAVASCRIPT FOR TOOL DEVELOPMENT

In contrast to most tools, our (meta-)tool is not implemented as application software for a specific computing platform (hardware, operating system, libraries), but as a webpage. Thus, it can be used in any modern web browser, regardless of the computing platform. It is built using HTML5 for the user interface (mainly the "canvas" as a drawing area) and JavaScript[2] for the program logic itself (graphics primitives, calculation of the curves, simulation). Note that although it is a webpage, the computations are done entirely on the user's computer, not on the web server. Besides the universal accessibility, another advantage is the easy development: much of the user interface is taken care of by the web browser, and e.g. other tests could easily be added with just a plain text editor, without requiring a compiler or other tools.

Of course, the disadvantage of an interpreted language like JavaScript is its lower execution speed compared to native code. For this particular purpose, that is no objection; on modern computers and in modern web browsers, it is fast enough for comfortable use: the graph lines move essentially in real time with the sliders.

Number crunching in the form of running simulations, as our webpage can also do, is a rather unusual application of JavaScript. Even for this, JavaScript is often fast enough, thanks to on the one hand today's fast computers, and on the other hand the effort browser developers have made to make JavaScript run fast. Of course, at large $N$ and large numbers of simulations runs, the run time would increase to a level beyond what is expected from a webpage, causing warning messages ("unresponsive script") in some browsers. To solve this, the simulations can be run in a separate thread in the background, using the so-called "web worker" technique made available on modern browsers. For very large $N$, a native program would technically still be a better choice; but the convenience of running it inside a web browser, without needing to install any specific software, may well be worth the price of a somewhat slower simulation.

# 6. CONCLUSION

Various hypothesis tests are available for statistical model checking, each with their own merits and disadvantages, and it is not always clear which one is best for some particular situation at hand. We have presented a "meta-tool" which does not by itself evaluate system models, but helps tool users and developers to better understand the different tests, and thus aids them to select the most appropriate test, and also to adequately parameterize the chosen test. Our tool comes in the form of an interactive webpage, allowing universal access on any modern computing platform.

[2]Note that, despite its name, JavaScript is not related to Java, nor to the Java applets that, a decade or so ago, were a popular way of deploying scientific animations on the web. While Java code is compiled into bytecode that requires separate software to run, JavaScript is an interpreted language running inside the web browser. In fact, due to security issues, Java applets nowadays are often disabled in browsers.

We are currently working on developing better hypothesis tests, i.e., tests that can draw a conclusion with fewer samples, while still satisfying the user-set confidence level. Since our tool is in the form of a webpage, new tests can be added to it in the future.

# 7. ACKNOWLEDGMENTS

This work is partially supported by the EU projects SENSATION, 318490, and QUANTICOL, 600708.

# 8. REFERENCES

[1] P. Ballarini, H. Djafri, M. Duflot, S. Haddad, and N. Pekergin. COSMOS: a statistical model checker for the hybrid automata stochastic logic. In *Proceedings of the Eighth International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 143–144. IEEE, 2011.

[2] Y. S. Chow and H. Robbins. On the asymptotic theory of fixed-width sequential confidence intervals for the mean. *The Annals of Mathematical Statistics*, 36(2):457–462, 1965.

[3] D.A. Darling and H. Robbins. Iterated logarithm inequalities. *Proceedings of the National Academy of Sciences of the United States of America*, 57(5):1188–1192, 1967.

[4] T. Hérault, R. Lassaigne, and S. Peyronnet. APMC 3.0: Approximate verification of discrete and continuous time markov chains. In *Proceedings of the Third International Conference on the Quantitative Evaluation of Systems (QEST)*, pages 129–130. IEEE, 2006.

[5] A. Legay, B. Delahaye, and S. Bensalem. Statistical model checking: an overview. In *Runtime Verification*, pages 122–135. Springer, 2010.

[6] D.P. Reijsbergen, P.T. de Boer, W.R.W. Scheinhardt, and B.R. Haverkort. On hypothesis testing for statistical model checking. *International Journal on Software Tools for Technology Transfer*, 17(4):377–395, 2015.

[7] D.P. Reijsbergen, W.R.W. Scheinhardt, and P.T. de Boer. A sequential hypothesis test based on a generalized azuma inequality. *Statistics & Probability Letters*, 97:192 – 196, 2015.

[8] K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic systems. In *Computer Aided Verification*, pages 266–280. LNCS Volume 3576, Springer, 2005.

[9] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[10] H.L.S. Younes, M. Kwiatkowska, G. Norman, and D. Parker. Numerical vs. statistical probabilistic model checking. *International Journal on Software Tools for Technology Transfer (STTT)*, 8(3):216–228, 2006.

[11] H.L.S. Younes and R.G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *Computer Aided Verification*, pages 223–235. LNCS Volume 2404, Springer, 2002.

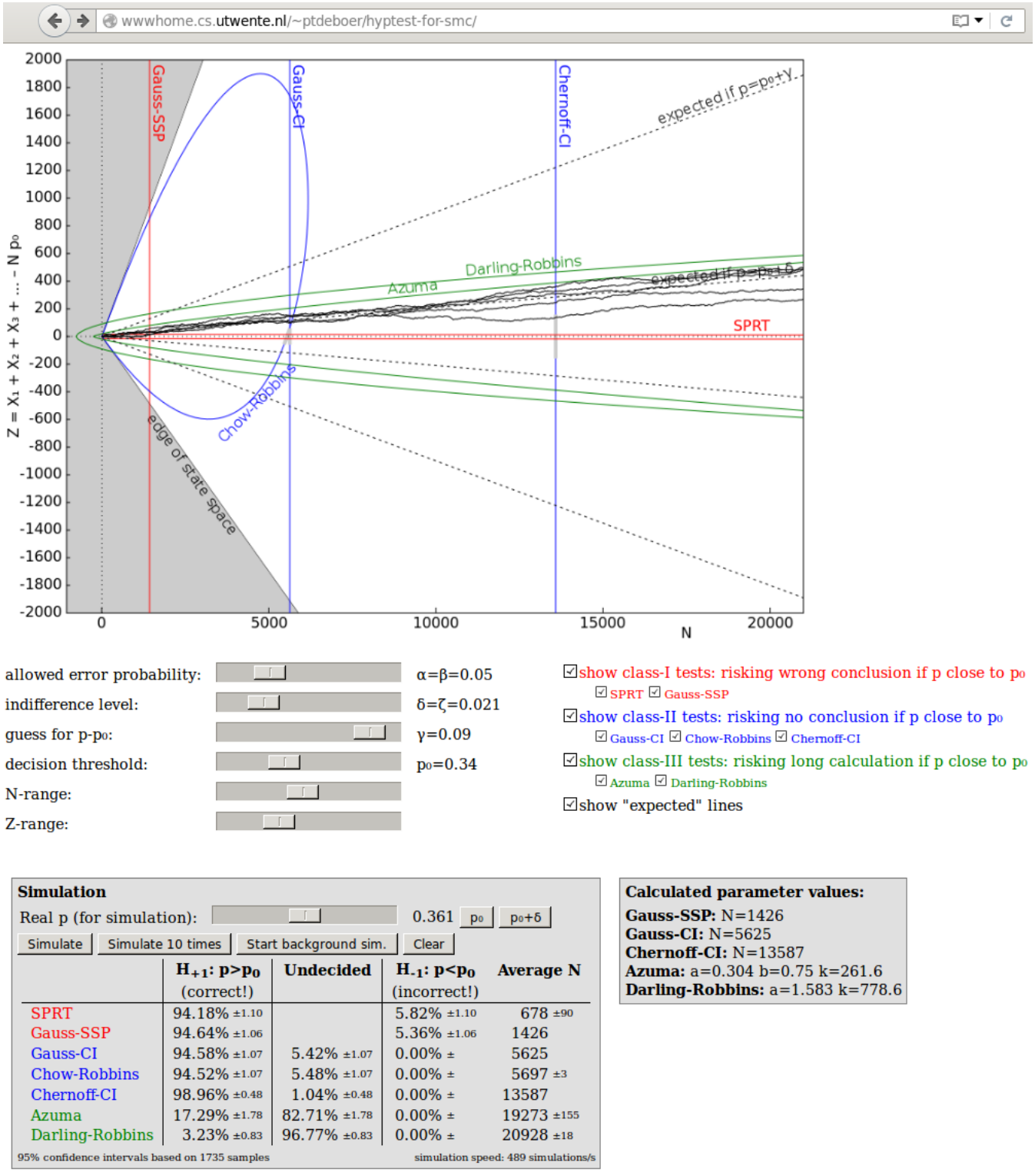


**Figure 1: Screenshot of the website being displayed in the Firefox web browser**