Treatment and Combination of Data Quality Monitoring Histograms to Perform Data vs. Monte Carlo Validation

Colin Nolan*†
cn580@york.ac.uk
Supervised by: Marco Rovere* and Giovanni Franzoni*

October 11, 2013

Abstract

In CMS's automated data quality validation infrastructure, it is not currently possible to assess how well Monte Carlo simulations describe data from collisions, if at all. In order to guarantee high quality data, a novel workflow was devised to perform 'data vs. Monte Carlo' validation. Support for this comparison was added by allowing distributions from several Monte Carlo samples to be combined, matched to the data and then displayed in a histogram stack, overlaid with the experimental data.

1 Introduction

CMS's data quality validation software is designed for the analysis of known variable distributions, allowing discrepancies to be identified either directly or by comparison to a reference distribution [1]. The reconstruction software and CMS detector conditions are validated in the same workflows that produce the data and Monte Carlo (MC) samples used for physics analyses [2]. In CMS, the Data Quality Monitoring (DQM) team is responsible for the infrastructure that delivers histograms of the observables; whilst the Physics Data and Monte Carlo Validation (PdmV) team takes care of the validation using 'data vs. data' and 'MC vs. MC' comparisons, across different versions of software and detector conditions.

In order to guarantee high quality data, it must be known whether, and how well, MC simulations describe the data; this type of comparison study is currently performed by most analysis teams [3]. However, such comparison is not possible centrally in the DQM given the current workflows: in order to perform 'data vs. MC' validation, distributions for several MC samples, each one representing a specific physics process, need to be combined (based on the cross sections of the MC sub-processes), matched to the integrated luminosity of the real data and then shown in a stack of overlapped distributions, overlaid with the data.

2 Goal

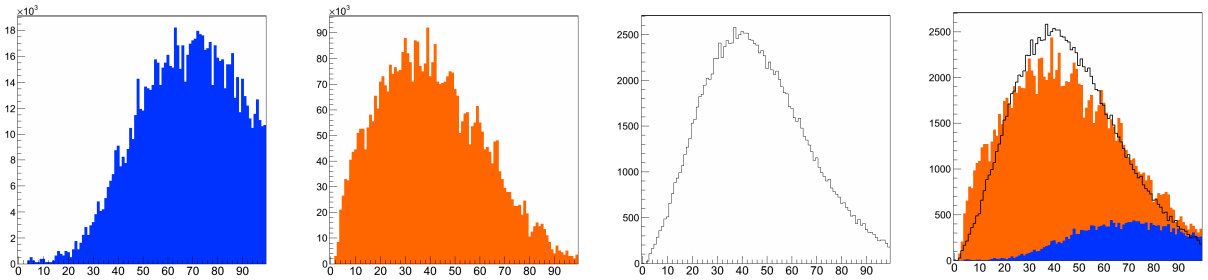
Having coordinated, CMS-wide support for automated 'data vs. MC' comparisons will aid validation campaigns promoting the study and monitoring of reconstruction performance, and the accuracy of Monte Carlo representations of the real data. The aim of this CERN summer student project was to extend the Data Quality Monitoring Graphical User Interface (DQM GUI) platform - a central component of CMS's DQM system [1] - so that histograms, extracted from CMS Software (CMSSW) DQM ROOT files, can be combined to allow comparisons between a stack of MC distributions and data. This work will contribute towards the efforts to complete a suite of new comparison tools by the time of restart of the LHC in 2015 [4], when several unprecedented data taking conditions will require close validation of physics performance

*CERN, PH-CMG-CO
†University of York, UK

and Monte Carlo representation accuracy.

3 Producing the Stacked Histogram

A stacked histogram is produced by taking a number of histograms and drawing them on top of one another, such that the frequencies of each bin of the stacked histogram are equal to the sum of the frequencies of the same bins in the constituent histograms. To make this stacked histogram comparable to the observed data, the constituent MC histograms are scaled such that the stack matches the integrated luminosity of the data. The data histogram is then drawn over the MC histogram to allow 'data vs. MC' comparisons to be made. This whole process is illustrated in Fig. 1.



(a) First stacked histogram constituent.

(b) Second stacked histogram constituent.

(c) Data histogram.

(d) Stacked histogram (produced by treating and combining histograms 1a and 1b) overlaid with the data histogram 1c.

Figure 1: Illustration of the process required to produce a stacked histogram that can be compared against the data.

The number of events recorded in the observed data N_D is equal to the product of the integrated luminosity L and the cross section of the physical process r_D:

$$N_D = L \cdot r_D$$

To compare the set of stacked MC histograms, $M = \{M_1, M_2, \ldots, M_K\}$, to the data D, the sum integrated luminosity of the MC histograms must be set to equal the integrated luminosity of the data, i.e.:

$$\sum_{i}^{K} L_{M_i} = L_D$$

The number of entries in the given MC histogram is proportional to the cross section of their associated physical process. For a histogram M_n, the sum of the frequencies at all bins equals the number of events represented by the histogram:

$$N = \int M_n(k)\,\mathrm{d}k$$

The Monte Carlo histograms model physics processes so, when combined, a single distribution should affect the overall distribution in proportion to its distinct occurrence probability. Therefore, the number of events in the set of samples that are to be combined is set proportionally to their associated cross section (the creation of these histograms is the responsibility of another system). To fix the integrated luminosity of the MC distributions when they are to be compared to a histogram of the data D, the scaling factor s, is applied to all bins of all MC histograms, where:

$$s = \frac{\int D(k)\,\mathrm{d}k}{\sum_i^K \int M_i(k)\,\mathrm{d}k}$$

Once the MC histogram is scaled to the integrated luminosity of the observed data, the integral of the histogram stack will equal the integral of the data histogram:

$$\sum_{i}^{K} \int M_i(k)\,\mathrm{d}k = \int D(k)\,\mathrm{d}k$$

4 Design and Development

4.1 Prototyping a Solution

A prototype of the solution was first created independent to the DQM GUI system, which has a relatively large codebase and a complex setup and build process. The creation of the prototype exposed requirements that needed to be more precisely defined, such as how data was to be input; it also lead to the identification of unanticipated problems, particularly with the use of ROOT[1]. An agile software development strategy [5] was used so that the solution could be regularly assessed and iteratively improved.

Fig. 2 shows an example plot produced by the prototype: it contains a histogram representing the pseudo data distribution (displayed by the solid black line) overlaid onto of a stacked histogram made by combining distributions from a number of different pseudo Monte Carlo simulations (identifiable in the stack by their alternate fill colours). Although the prototype system did not use real data (each of the distributions is a random Gaussian distribution), the system did use the ROOT objects required in the final solution and it correctly solved the issues surrounding the scaling and combination of distributions. The Application Programming Interface (API) and code written for the prototype were used as a guide for producing the code for the DQM GUI.

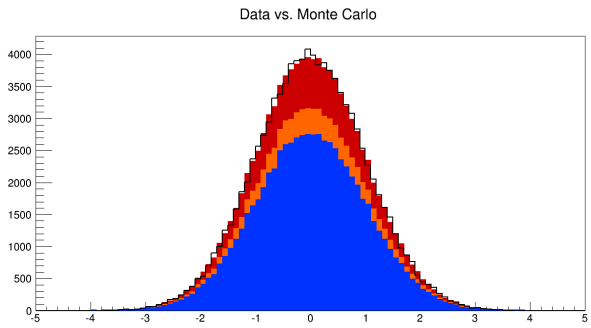


Figure 2: Plot produced by prototype.

4.2 The DQM GUI

The DQM GUI (shown in Fig. 3) is a web-based tool, accessible using a web browser [6]. To meet the new requirements, the system was modified such that it could:

1. Indicate to the plot renderer that a stacked histogram was to be built with the given distributions.
2. Capture the histograms of interest and stop them being drawn as normal.
3. Perform appropriate scaling of MC histograms so that they could be compared to the data.
4. Draw the MC stacked histogram, overlaid with the data histogram.

To allow the user to indicate that a 'data vs. MC' comparison should be made, the view settings input box was changed: in the modified section of the settings box, shown in Fig. 4, the user can define the dataset(s) containing reference MC histograms that are to be used and then select the option to position them 'stacked' against the data. The diagrams in the GUI (of which 8 are shown in Fig. 3) are embedded onto the HTML page using JavaScript via XHR (an API that allows JavaScript to send requests to a web server),

[1] `http://root.cern.ch/`

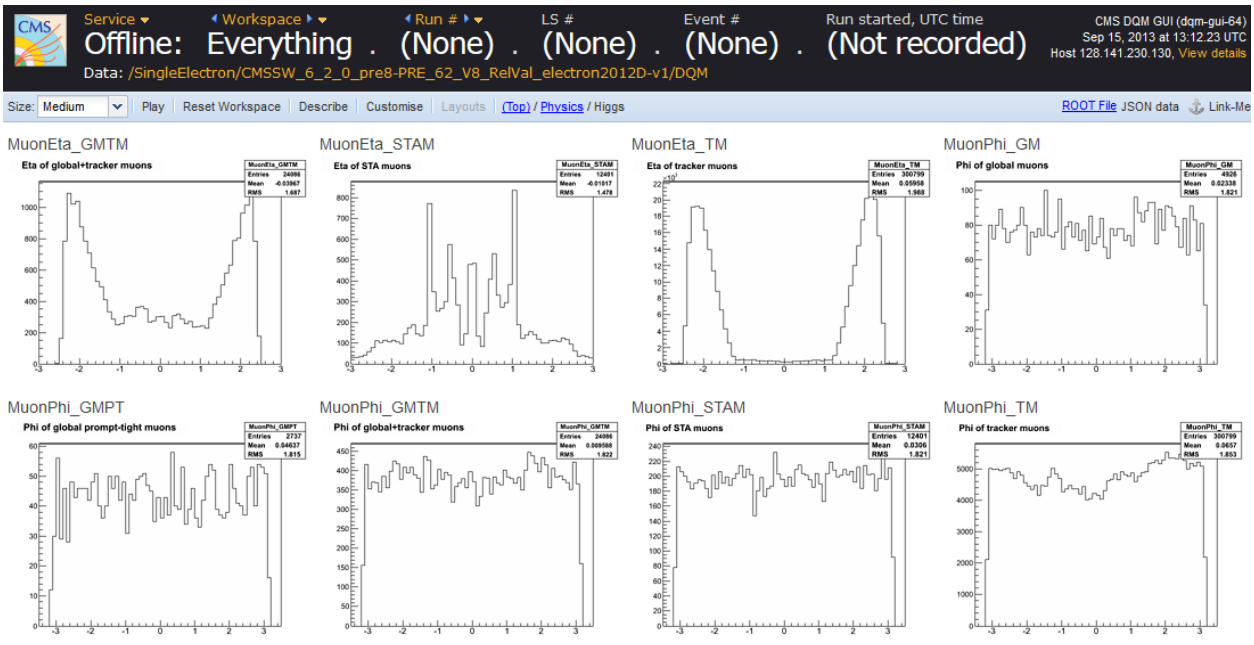


Figure 3: Screenshot showing the DQM GUI.

the executing script reads a JSON representation of the DQM GUI's session state, then uses variables in the session state when communicating to the 'plotfairy' service how the images it returns should be rendered through the images' HTTP GET request parameters [7]. In order to reflect the user's settings in the rendered plots, when changes are made in the settings box, the session state is modified.

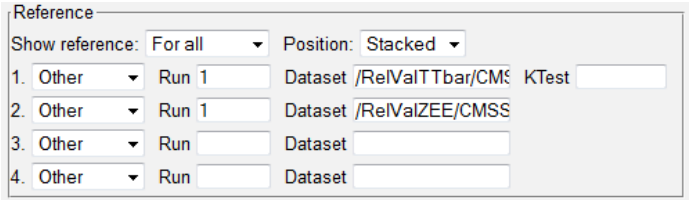


Figure 4: Screenshot showing the option to create a stacked histogram via the DQM GUI's view settings.

Once the challenge of understanding the build process and the relevant parts of the existing codebase was overcome, the system for creating and displaying the stacked histograms was designed and written. A Unified Markup Language (UML) class diagram, showing the structure of the C++ code, can be seen in Appendix A[2]. The system was designed not to be coupled the existing codebase, which allowed it to be tested locally in a test project that ran within the developer's Integrated Development Environment (IDE). To aid future development and maintenance, best efforts were made to ensure that the new code was modular, well documented and closely followed CERN's C++ coding standards[3].

The implemented MC histogram treatment and combination functionality was verified by comparing merged sets of MC histograms against histogram stacks made from the same sets of MC histograms: it was observed in all cases that the merged histogram exactly matched the stacked histogram. An example plot showing

[2]Unused classes that allow the weighting of MC histograms to be explicitly defined are included in the code and the class diagram; it was suggested that this functionality may be useful in the future so it has been kept.

[3]`http://pst.web.cern.ch/PST/HandBookWorkBook/Handbook/Programming/CodingStandard/c++standard.pdf`

this agreement is seen in Fig. 5.

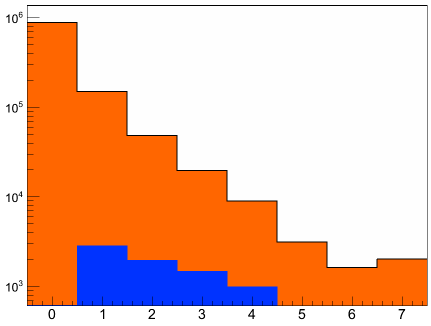


Figure 5: Plot showing the perfect match between a histogram created from merging two MC histograms (black line) and a stacked histogram (colour filled).

5 Conclusion

The DQM GUI was successfully extended to support 'data vs. MC' validation, with multiple MC sample sources. The user can now compare the data to a specified set of MC histograms via use of the modified view settings options box. Changing the settings prompts the DQM GUI to setup the appropriate display, where the renderer treats, combines and draws the MC histograms, overlaid with the observed data.

6 Future Work

RelMon, a Python-based tool used for automated histogram comparisons [8, 2], could be updated to use the new functionality, allowing it to perform statistical tests between the data and the stacked MC distributions. RelMon could continue invoking the DQM GUI plot renderer but, for this new type of comparison, it would set the correct HTTP request parameter to allow the stacked histogram to be used.

Considering the DQM GUI system's non-functional requirement of high performance [1]: there are numerous opportunities to improve the code by allowing parallelism to occur. An example of a reasonably computationally expensive procedure that could benefit from being made concurrent is the scaling of histograms - a process that requires the modification of the frequency value of each histogram's many bins. However, the introduction of concurrency should be done cautiously as not all ROOT objects and functions are fully thread-safe: as of the latest release version of ROOT, histograms are not thread-safe [9].

Aside from achieving the functional requirements, focus was placed on ensuring that any new C++ code did not lead to memory leaks. To help with this task, Cppcheck[4] (a static code analyser) and Valgrind tools[5] (a framework that contains profiling tools able to detect memory management and threading bugs [10]) were used. Unfortunately, not all problems flagged up by these tools were solved: however, they should be before the modified system is put into production. Most issues surround ROOT objects, where their ownership and lifespan is made unclear due to ROOT's use of global state.

[4] `http://cppcheck.sourceforge.net/`
[5] `http://valgrind.org/`

To validate that the new system meets the requirements of the target user and to solve any usability issues, user testing could be conducted, coupled with further, iterative redesign. Such process would ensure that, when the comparison functionality is made available in the DQM GUI by the time of the LHC restart, it is intuitive to use and maximally useful.

References

[1] M. Rovere, "The CMS data quality monitoring & validation software: Experience, workflows and tools," CERN, May 2013. [Online]. Available: https://indico.cern.ch/getFile.py/access?contribId=51&sessionId=5&resId=0&materialId=slides&confId=236650

[2] M. Rovere and G. Franzoni, "CMG summer student projects 2013," Feb. 2013. [Online]. Available: https://twiki.cern.ch/twiki/bin/viewauth/CMS/CMGSummerStudents2013#8_Marco_Rovere_and_Giovanni_Fran

[3] F. De Guio, S. Dutta, M. Rovere, G. Franzoni, and C. Ferro, "Data vs. MC comparison for physics validation," CERN, Jul. 2013. [Online]. Available: https://indico.cern.ch/getFile.py/access?contribId=4&resId=0&materialId=slides&confId=226661

[4] C. Ferro, G. Franzoni, and F. De Guio, "Data vs. monte carlo for validation," Jul. 2013. [Online]. Available: https://indico.cern.ch/getFile.py/access?contribId=11&resId=1&materialId=slides&confId=251030

[5] K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries *et al.*, "Manifesto for agile software development," 2001. [Online]. Available: http://agilemanifesto.org/

[6] J. Stupak, "Data quality monitoring (DQM) for physics analysis," Jun. 2013. [Online]. Available: https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookDQM

[7] L. Tuura, G. Eulisse, and A. Meyer, "CMS data quality monitoring web service," *J. Phys.: Conf. Ser.*, vol. 219, no. 7, p. 072055, Apr. 2010. [Online]. Available: http://iopscience.iop.org/1742-6596/219/7/072055

[8] A. Norkus, "RelMon: a tool to perform goodness of fit tests of large sets of histograms," Jul. 2013. [Online]. Available: https://twiki.cern.ch/twiki/bin/view/CMSPublic/RelMon

[9] M. Rovere, "DQM GUI - current status and future plans," Nov. 2012. [Online]. Available: https://indico.cern.ch/getFile.py/access?resId=0&materialId=slides&contribId=24&sessionId=9&subContId=4&confId=215512

[10] "Valgrind home," Sep. 2012. [Online]. Available: http://valgrind.org/

Glossary

CERN The European Organization for Nuclear Research is an international organization focused on nuclear research.

CMS The Compact Muon Solenoid experiment is one of two large general-purpose particle physics detectors built on the LHC at CERN.

DQM GUI The Data Quality Monitoring Graphical User Interface system is a web site for browsing data quality histograms.

JavaScript JavaScript is a prototype-based scripting language, most commonly used in HTML web pages.

JSON JavaScript Object Notation is a standardised data exchange format, derived from JavaScript.

LHC The Large Hadron Collider is a high-energy particle collider built by CERN.

RelMon The Release Monitoring tool performs goodness of fit tests on large sets of histograms.

ROOT An object oriented framework developed at CERN for large scale data analysis.

XHR XMLHttpRequest is an API available in JavaScript that is used to send requests to and to get responses from a web server.

Acronyms

API Application Programming Interface.

CMSSW CMS Software.

DQM Data Quality Monitoring.

GUI Graphical User Interface.

HTML HyperText Markup Language.

HTTP HyperText Transfer Protocol.

IDE Integrated Development Environment.

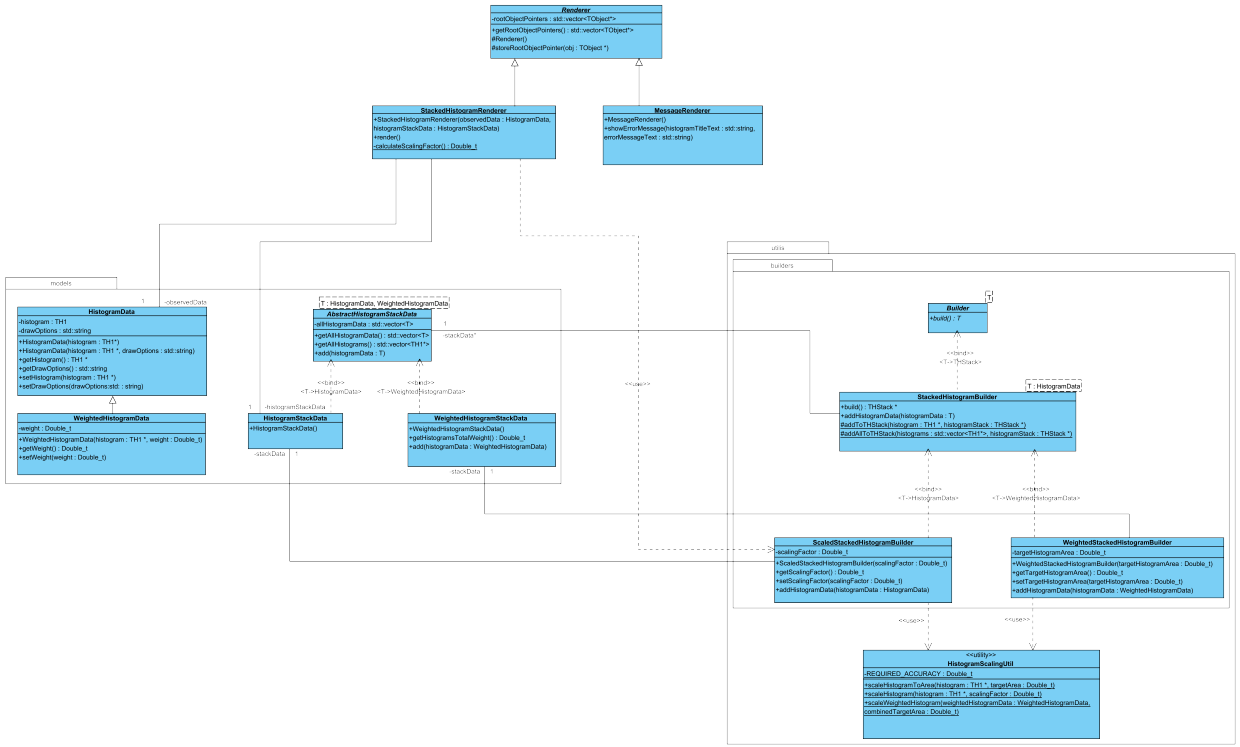
MC Monte Carlo.

PdmV Physics Data and Monte Carlo Validation.

UML Unified Markup Language.

Appendix

A UML Class Diagram for New Code



∞