Development of a Tool to Automate Warm-up Analysis in Discrete Event Simulation (ExtendSim 2024)

CAMIM5

Final Year Project - Thesis

Name: Harry Caffrey

Student ID: 21437872

Supervisor: Dr. John Geraghty

March 2026

DECLARATION

I understand that the University regards academic misconduct as grave and serious.

I have read and understood the DCU Academic Integrity Policy. I accept the penalties that may be imposed should I engage in academic misconduct.

I have identified and included the source of all facts, ideas, opinions and viewpoints of others in the assignment references. Direct quotations, paraphrasing, discussion of ideas from books, journal articles, internet sources, module text, or any other source whatsoever are acknowledged and the sources cited are identified in the assignment references.

I have not made unauthorised use of artificial intelligence aids.

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

I have used the DCU library referencing guidelines (available at https://www.dcu.ie/library/citing-referencing and/or the appropriate referencing system recommended in the assignment guidelines and/or programme documentation.

By signing this form or by submitting material online I confirm that this assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.

By signing this form or by submitting material for assessment online I confirm that I have read and understood the DCU Academic Integrity Policy

Signature: Harry Caffrey

Date: 19/03/2025

ACKNOWLEDGEMENTS

I would like to take this opportunity to say a huge thank you to my supervisor, Dr. John Geraghty for his guidance and expertise throughout the course of this project. I would also like to thank any of my peers who have assisted me in my project. Finally, like to extend my love and appreciation to my family who have been nothing but supportive and helpful over the past two semesters.

ABSTRACT

In steady-state discrete event simulation, output collected at the beginning of a run is often influenced by unrealistic initial conditions such as empty queues and idle resources. This initial transient period, commonly referred to as the warmup period, can introduce bias into estimated performance measures if it is not identified and removed prior to analysis. Although a range of warmup detection methods has been proposed in the literature, their practical use is often limited by subjectivity, statistical complexity, and the lack of accessible implementation within commercial simulation workflows.

This project presents the development of an automated warmup analysis tool for simulation output exported from ExtendSim 2024. The tool was implemented as an R Shiny application, allowing users to upload simulation data, apply selected warmup detection methods, and compare the resulting truncation recommendations through an interactive graphical interface. The methods implemented were Welch's graphical smoothing procedure, the Marginal Standard Error Rule-5 (MSER-5), and a Statistical Process Control (SPC) based approach. A final comparison view was developed to allow the user to assess the recommended truncation points against a Welch-smoothed representation of the data and select a preferred warmup period before downloading a cleaned dataset with the transient observations removed.

TABLE OF CONTENTS

1. INTRODUCTION

Discrete event simulation (DES) is widely used to analyse complex systems whose behaviour evolves over time through the occurrence of discrete events. Rather than relying on analytical solutions, DES models represent system dynamics through probabilistic event sequences, enabling performance measures such as work-in-progress (WIP), queue length, throughput, and utilisation to be evaluated under realistic operating conditions. This makes DES particularly valuable in engineering applications where direct experimentation on real systems is impractical, costly, or disruptive.

In steady-state simulation studies, the objective is to estimate long-run performance measures that are independent of the system's initial conditions. However, simulation models are typically initialised in an artificial empty-and-idle state, where queues are empty and resources are fully available. These starting conditions do not reflect the normal operating behaviour of real systems, which usually exist in a partially loaded and dynamically fluctuating state. As a result, the early portion of simulation output is dominated by transient behaviour as the system transitions from its initial state towards steady state. This period is referred to as the warmup period or initial transient.

If this transient data is not removed prior to analysis, it introduces initialisation bias into performance estimates. For example, queue lengths and WIP values are typically underestimated during the early stages of a simulation because the system has not yet reached its normal operating level. This bias cannot be eliminated through additional replications, as it is systematic rather than random, and it may lead to incorrect conclusions if not properly addressed. Consequently, the identification and removal of the warmup period is a critical step in ensuring the validity of steady-state simulation results.

Despite its importance, warmup detection remains a challenging and often inconsistent process in practice. Graphical methods such as Welch's approach rely on visual interpretation of smoothed output data, which introduces subjectivity and variability between analysts. More advanced statistical methods, including the Marginal Standard Error Rule (MSER) and Statistical Process Control (SPC) approaches, offer objective alternatives but are not commonly implemented in a user-friendly form. In many cases, these methods require external data processing and statistical expertise, creating a barrier to their routine use in applied simulation studies. This results in a gap between the availability of rigorous warmup detection techniques in the literature and their practical adoption within industry and engineering workflows.

This thesis addresses this gap through the development of an automated warmup analysis tool designed to improve the accessibility, consistency, and reliability of steady-state simulation output analysis. The tool is implemented as an R Shiny web application and is designed to process data exported from ExtendSim 2024 models. Users can upload simulation output data, apply selected warmup detection methods, and compare

the resulting truncation points through an interactive interface. The tool then enables the user to select an appropriate warmup period and export a cleaned dataset with the transient observations removed.

The central research question of this thesis is:

- Can established warmup detection methods be effectively automated within an accessible tool to reduce subjectivity and improve the accuracy and repeatability of steady-state simulation output analysis?
- To address this question, the following objectives are defined:
- To review the theoretical and methodological foundations of warmup detection in steady-state discrete event simulation.
- To implement established warmup detection methods, including Welch's method, MSER-5, and a Statistical Process Control (SPC) approach.
- To develop an interactive R Shiny application that allows users to upload simulation data, apply these methods, and compare their outputs.
- To enable the selection and export of cleaned datasets with the warmup period removed.
- To validate the performance of the implemented methods using analytically tractable queueing models and assess their effectiveness using metrics such as bias and mean squared error (MSE).

Through this combination of methodological implementation, interface design, and validation, the project aims to provide a practical tool that enhances the usability of warmup detection methods while improving the reliability of steady-state simulation studies.

2. LITERATURE REVIEW

2.1. Introduction To Discrete Event Simulation (DES)

2.1.1. Simulation

Simulation is the imitation of a real-world system over time through the generation of an artificial history from which performance measures about the system's operating characteristics can be taken[1], [2]. Rather than solving a mathematical model analytically, simulation generates sequences of system states based on probability distributions, allowing complex interactions that resist closed-form treatment to be examined through controlled experimentation. This approach is particularly useful where direct experimentation on the real system is impractical, costly, or disruptive [3], [4].

In discrete event simulation specifically, the system changes state only at discrete points in time due to the occurrence of system events such as arrivals, service completions, or failures [2], [5]. Between events, the

state is assumed constant, so the simulation clock advances directly from one event time to the next rather than progressing in fixed increments. This next-event mechanism is computationally efficient and has made DES a widely used approach for modelling manufacturing systems, logistics networks, healthcare pathways, and service operations [4], [5]. The output of such models typically takes the form of time-series performance measures, including queue length, throughput, and resource utilisation. As stochastic inputs produce variable outputs from one run to another, these measures must be interpreted using appropriate statistical analysis [2], [6].

2.1.2. Classification of Simulation Models

A fundamental distinction in simulation output analysis is that between terminating and non-terminating simulations [2], [5]. A terminating simulation has natural starting and end points defined by the problem itself. For example, the simulation of a bank branch over a single business day begins when the doors open and ends when the last customer departs, and an empty branch at opening is a realistic description of the system rather than a modelling convenience. Performance estimates in terminating simulations are therefore interpreted with respect to that specific finite horizon [2], [7].

Non-terminating simulations, by contrast, represent systems that operate continuously without a natural end point. A manufacturing line running all the time, a server processing internet requests, or a hospital emergency department do not have a natural finishing point, so the analytical interest lies in their long-run average behaviour [2], [8]. The objective of steady-state simulation is to estimate this long-run mean, as a property of the system that is independent of the time at which observation begins and the initial state of the model [1], [9]. It is in this non-terminating setting that the warm-up problem arises, because the model must be started in some initial state that will almost always differ from the true steady-state distribution of the system [8].

2.1.3. The Role of Output Analysis

Because simulation models incorporate random inputs, the outputs they produce are also treated as random variables. A single run of a simulation provides only one version of the output process, but drawing reliable inferences requires accounting for the variability of that process [1], [6]. The most common approach to minimise the randomness in the system is to perform multiple independent replications using different random number seeds so that the variation between replications can be used averaged out to produce data closer to that of steady state behaviour [2], [5]. Output analysis is the branch of simulation methodology concerned with extracting statistically valid results from the data produced by simulation runs, and it includes the removal of initialisation bias [6], [8]. Without output analysis, simulation results can be misrepresented as more statistically precise than is justified, leading to overconfidence in biased or unstable performance estimates [2].

2.2. The Initialisation Bias Problem

2.2.1. Start-up Problem Definition

The start-up problem, also referred to as the initialisation bias problem or the warm-up problem, arises when the chosen initial conditions of a steady-state simulation are not representative of the system's steady-state distribution [2], [10]. The error this introduces into steady-state performance estimators is called the initialisation bias. The expected value of a performance metric based on the full simulation run differs from the steady-state mean, by an amount that depends on how far the initial state is from steady state and how rapidly the system converges [10]. This bias cannot be reduced by running more replications because it is systematic rather than random, and it is not eliminated by longer run lengths unless the transient phase becomes a negligible fraction of the total output [2], [8].

The deletion method addresses this problem by designating an initial warm-up period during which during which output data are excluded from analysis [2]. Data collection begins only once the warm-up period is over, on the assumption that the system has by then stabilised sufficiently close to its steady-state distribution for the remaining output to be representative. The key challenge is determining how long this warm-up period should be for a given system [8], [10].

2.2.2. Initial Transient Vs. Steady State

The initial transient is the period during which simulation output is strongly influenced by the system moving from its starting state towards its steady-state operating conditions [11], [12]. For an M/M/s queue initialised in the idle state, this transient may appear as a sustained upward trend in the tracked performance metric as the system builds from zero towards its steady state mean, a process that can take thousands of simulated time units in heavily loaded systems [11]. The transition to steady state is not marked by a sharp transition but instead occurs gradually. At any point in time, the output distribution may still contain residual transient effects, although their influence decays as the run progresses [1], [8].

The rate at which this transient effect on the data decays, depends critically on the traffic intensity, ρ. At low utilisation the system approaches steady-state behaviour relatively quickly, whereas at utilisation levels close to unity the adjustment can be very slow [11]. This dependence on traffic intensity helps explain why fixed warm-up rules of thumb, such as discarding the first ten percent of the run, are unreliable. A truncation rule that is adequate for a lightly loaded system may be completely inadequate for a heavily loaded one [12], [13].

2.2.3. Causes and Effects of Initial Bias

The empty-and-idle starting condition is widely used in simulation practice because it is simple to specify and does not require prior knowledge of the system's steady-state distribution [2], [10]. A factory model

may begin with no work in progress, a hospital model with empty wards and no patients, and a call centre with all agents available and no callers in queue. In most real systems, however, these conditions do not reflect normal operation. The output collected while the simulation moves away from this artificial starting point is therefore systematically unrepresentative of steady-state behaviour [11], [12].

The practical effect is that performance estimates computed from runs that include the transient period are often biased downward for queue-length estimates, because the queue has not yet built to its steady-state level, and may be biased in either direction for other measures depending on the system structure [12], [13]. The magnitude of this bias can be substantial. In high utilisation systems, the transient period may persist for very long times and lead to large errors in steady state estimates if left in the dataset [11], [14]. The factors effecting the warm-up duration include traffic intensity, initial conditions, and the specific performance measure being estimated [12].

2.2.4. Balancing the Effects of Bias and Variance

Choosing the truncation point, d, in the deletion method involves a statistical trade-off between two sources of estimation error [10], [14]. Truncating too early, before transient effects have completely diminished, leaves residual initialisation bias in the post-truncation dataset. Truncating too late discards observations that are already representative of steady-state behaviour, reducing the effective sample size available for estimation and increasing the variance of the resulting mean estimate [14], [15]. From this perspective, the preferred truncation point is the value of d that minimises the mean squared error (MSE) of the post-truncation dataset, balancing the competing effects of bias and variance [14].

A key theoretical result is that the MSE-minimising truncation point depends on both the shape and duration of the transient and the autocorrelation structure of the steady-state output, all of which are system-specific and unknown in advance [8], [10]. This explains why there is no universal warm-up detection rule that performs optimally across all systems. Any method that does not have prior knowledge of the transient must instead approximate this balance from the output itself, and the quality of that approximation will vary from one system to another. It is this idea that underpins MSER, which seeks to approximate the MSE criterion directly from the simulation output without requiring prior knowledge [14].

2.3. Warm-up Identification Methods

2.3.1. Welch's Method

Welch's graphical approach is one of the oldest and most widely used methods for detecting the warm-up period. The method is based on constructing the ensemble mean of the output series across multiple independent replications and then applying a centred moving average smoother to reveal the underlying transient trend [16]. This approach works by averaging a symmetrical window of w, averaging w

observations on either side of a point together with the point itself to produce a smoothed value [16], [17]. Near the beginning of the series, where fewer than *w* previous observations are available, a growing window is used so that early observations are retained. The analyst then inspects the smoothed curve visually and selects the point beyond which it appears to have levelled off as the estimated warm-up period [2], [17].

The choice of window size, *w*, is an important aspect of the method. If the window is too small, high-frequency oscillations remain in the smoothed curve, making the point of stabilisation difficult to judge visually. If the window is too large, the curve may be over-smoothed, and the apparent location of convergence may be shifted [18]. The ensemble averaging step is the main mechanism for noise reduction, since the variance of the ensemble mean decreases as the number of replications increases. With a sufficient number of replications, the smoothed curve can therefore provide a clearer indication of the transient trend [17]. Comparative studies have found graphical methods to be effective when the transient is short and clearly defined, but less reliable when the transient is long or gradual, in which case different analysts may identify different truncation points from the same plot [17].

2.3.2. MSER and MSER-5

The Marginal Standard Error Rule (MSER) was introduced as one of the first fully automated and objective procedures for warm-up detection, recasting the problem as the minimisation of a statistic that approximates the mean squared error of the post-truncation mean estimator [19]. At a candidate truncation point *d*, the MSER statistic is defined as the sum of squared deviations of the post-truncation observations from their post-truncation mean, divided by the square of the number of remaining observations. The value of *d* that minimises this statistic is taken as the preferred truncation point, since it identifies the stage at which the remaining sample is most statistically homogeneous and the influence of the initial transient has largely dissipated [14], [19].

MSER-5 was developed to address a limitation of the original MSER when applied to autocorrelated output. When the statistic is computed directly from the raw time series, residual autocorrelation can reduce its reliability as the denominator does not reflect the effective sample size under dependence [19]. To address this, the raw output is first batched into non-overlapping groups of five consecutive observations, and the criterion is then applied to the resulting batch means. The batch size of five was selected empirically as the smallest value that reduces autocorrelation sufficiently while preserving enough resolution to identify the truncation point [19]. Batching also improves computational efficiency by shortening the series over which the statistic must be evaluated. Sampling distribution studies for the M/M/1 queue show that the variability of the MSER truncation point increases substantially with traffic intensity, reinforcing the need for multiple replications under high-load conditions [20]. The method performs best for monotone transients, but in

systems with high stochastic variability it may incorrectly identify early local minima as the truncation point [17], [21].

2.3.3. Statistical Process Control

The use of SPC charts for warm-up detection is based on a clear similarity between the two settings: a simulation in its initial transient can be viewed as analogous to a manufacturing process that is out of statistical control, whereas a simulation in steady state behaves like a process in control [22]. Control charts, originally developed for industrial quality monitoring, use the statistical properties of a stable process to define limits within which in-control observations are expected to fall. Applied to simulation output, the warm-up period is taken as the last point where a control chart rule is violated, since repeated violations indicate that the process has not yet reached stationarity [22], [23].

The Nelson rules [23] define 'out of control' rules for Shewhart charts including the following: a single point beyond the three-sigma limits; two of three consecutive points on the same side beyond the two-sigma limits; four of five consecutive points on the same side beyond the one-sigma limits; and nine consecutive points on the same side of the centreline. Applying these rules to simulation output requires the data to be pre-processed into batch means that are approximately independent and normally distributed, since both autocorrelation and non-normality increase the probability in which a false truncation point is suggested [22]. The control limits must also be estimated from the steady-state portion of the series rather than from the full run, otherwise the biased data can displace the centreline away from the steady-state mean [22]. Comparative studies show that SPC generally produces more conservative truncation recommendations than MSER-5, which has been attributed to the sensitivity of the run-based rules to weaker systematic patterns that may persist after the main transient has passed [17], [21].

2.4. Automated Output Analysis

2.4.1. The Need for Automation

The case for automating warm-up analysis can be supported on multiple grounds. Methodologically, graphical approaches are subjective, and the substantial large variability reported in comparative studies means that warm-up estimates based on visual inspection are often difficult to reproduce consistently [17]. Two analysts examining the same Welch plot may identify truncation points that differ greatly, introducing variability into the results that is unrelated to the system itself [17], [24]. Automated procedures reduce this problem by replacing visual judgement with explicit computational criteria [24].

The practical case is also strong. Surveys of simulation practice consistently show that warm-up analysis is one of the stages most likely to be omitted or handled informally in industrial studies [17], [25]. The main barriers are the complexity of available statistical methods, the time required for manual analysis, and the

lack of integrated support in commercial simulation platforms [26]. Tools that automate this process of initialisation process removal and require minimal analyst effort are more likely to be applied consistently in practice. Even if a manual analysis could in some cases be more carefully tailored, a well-designed automated procedure would still improve the overall standard of simulation-based performance metrics by making the warm-up deletion a more structured, repeatable process [24], [25].

2.4.2. Sequential vs. Fixed Sample Procedures

A important distinction separates fixed sample procedures, which analyse a complete pre-specified dataset, from sequential procedures, which adapt their data requirements in response to the observed output [8], [17]. Fixed-sample procedures take the simulation data as given and return a truncation recommendation based on the observations available. All three methods implemented in this project are fixed sample procedures, meaning that they analyse the available dataset and return results without requesting additional simulation runs.

Sequential procedures are attractive in principle because they avoid the need to pre-specify run length and can adapt to the warm-up characteristics of the system being studied. The AutoSimOA software follows a sequential logic by attempting to identify a valid truncation point in the first half of the available series and requesting additional simulation output if the data are insufficient [24]. In practice, however, sequential procedures require a feedback loop between the analysis tool and the simulation platform, which is difficult to implement with ExtendSim 2024. The fixed-sample approach adopted in this project is therefore a pragmatic choice that prioritises implement ability while still remaining consistent with established warm-up detection methods [24], [25].

2.5. Software Selection & Reliability

2.5.1. Reliability of Statistical Software

The numerical accuracy of statistical software should not be assumed automatically. Early systematic evaluations showed that several widely used packages produced inaccurate results on benchmark problems, particularly for procedures involving ill-conditioned inputs or near-degenerate datasets [27], [28]. The National Institute of Standards and Technology (NIST) Statistical Reference Datasets provide a set of benchmark problems with certified reference values computed to high precision using multiple independent methods. Comparison against these datasets allows numerical weaknesses in statistical software to be identified more rigorously than would be possible through routine use on well-behaved data [27], [28].

A broader comparison of nine statistical software packages against NIST benchmarks found substantial differences in performance across both packages and procedure types [29]. Some of the largest discrepancies were observed in variance-related calculations, which is relevant to this project [29]. For a

tool whose output depends on the reliability of underlying statistical computations, these findings make the choice of software environment a relevant methodological consideration rather than a purely practical one [27], [29].

2.5.2. R and Python as Statistical Computing Environments

A comparison of R and Python against NIST reference datasets for univariate statistics and linear regression found that R's base statistical functions generally matched the reference values to higher numerical precision than commonly used Python libraries such as 'statsmodels' and 'pandas' [30]. The differences were most noticeable in more difficult estimation settings, including problems involving near-collinear predictors and heavy-tailed distributions, where R more often produced results closer to the certified reference values [29], [30]. For the procedures relevant to this project, including the Anderson-Darling test and the Von Neumann ratio statistic, numerical implementation quality is important because small computational differences can affect method performance and interpretation [29].

R was developed specifically as a language for statistical computing and graphics [31], and this is reflected in the structure and transparency of its statistical ecosystem. The code for base and recommended packages is openly available, and the wider package ecosystem includes many tools developed with a clear statistical focus. This makes R a suitable environment for implementing and inspecting the numerical procedures required for warm-up analysis. In addition, the ggplot2 package provides a robust and reproducible framework for visualising time-series output, making it well suited to presenting the graphical results produced by this tool [32].

2.5.3. R Shiny as an Interface for Statistical Analysis

R Shiny is a web application framework for R that enables the development of browser-based interactive applications without requiring end users to write R code [32]. The framework is based on a reactive programming model in which outputs are automatically recalculated whenever their inputs change. This makes it well suited to tools in which parameters such as the Welch window size or the significance level used in batch-size testing can be adjusted interactively while the analysis updates in real time. The value of Shiny for making statistically complex workflows more accessible to non-specialist users has been demonstrated across multiple scientific domains [33].

For warm-up analysis, this interface layer addresses a practical barrier identified in the automation literature. Frameworks such as AutoSimOA [24] and methods such as MSER and SPC [19], [22] are well established, but their use typically requires direct interaction with statistical code. A Shiny-based interface allows these methods to be applied through a graphical environment while retaining the underlying R computations. In

this way, the combination of R's statistical capabilities and Shiny's accessibility helps address the gap between the availability of rigorous methods and their routine use by practitioners [17], [25].

2.6. Theoretical Benchmarks

2.6.1. Analytical Queueing Models as Validation Benchmarks

Validating a warm-up detection tool requires test cases for which steady-state performance can be established independently of the simulation output, providing an objective benchmark against which post-truncation estimates can be assessed. Standard queueing models, particularly the M/M/1 and M/M/s queues, are well suited to this purpose because their steady-state behaviour is analytically well characterised and widely used in simulation research [2], [11]. For this reason, they provide a reliable basis for testing whether a warm-up detection method removes enough of the transient period for simulation estimates to converge towards known theoretical values. The analytical performance measures used for comparison in this study can be calculated using the formulae in Section 4.5.1. Analytical M/M/1 Performance Measures.

The use of M/M/1 queues at multiple traffic intensities as validation test cases is well established in the warm-up detection literature [17], [19], [21]. Testing across a range of utilisation levels exposes detection methods to different transient behaviours, from short and clearly defined warm-up periods at low utilisation to slower and more gradual convergence at high utilisation [11]. This makes analytical queueing models particularly useful for assessing whether a warm-up detection method remains reliable under different system conditions. The role of the M/M/1 queue analytical values is therefore not only to provide a convenient benchmark, but also to support a more detailed evaluation of how sensitive the implemented methods are to changes in traffic intensity and transient duration.

2.6.2. Verification of Truncation Accuracy

Beyond comparing post-truncation sample means against analytical values, a more rigorous verification approach is to assess whether the recommended truncation point improves overall estimation accuracy relative to the untreated dataset. For validation models, where the theoretical steady-state value is known, the post-truncation estimates can be compared directly against the analytical benchmark using bias, percentage error, variance, and mean squared error (MSE) [14], [20]. This allows the effect of warm-up removal to be evaluated not only in terms of closeness to the analytical mean, but also in terms of the trade-off between bias reduction and estimator variability.

In this project, truncation accuracy is verified by comparing the raw and post-truncation datasets against the analytical reference value and examining whether the selected method reduces MSE. Since MSE combines both squared bias and variance, it provides a single measure of overall estimator quality and is therefore particularly useful for assessing whether warm-up deletion has improved the result [14]. A

reduction in MSE after truncation indicates that the removed portion of the dataset was contributing more bias than useful steady-state information, whereas an increase in MSE suggests that the truncation point was inappropriate and removed too much representative data[21].

This form of verification is especially relevant for MSER-5, since the method is explicitly motivated by the minimisation of an MSE-based criterion [14], [20]. Agreement between the analytical benchmark and a reduction in observed post-truncation MSE provides practical evidence that the method is functioning as intended. For SPC, verification is complementary rather than identical. In this case, the method is assessed by whether the resulting truncation improves agreement with the analytical steady-state value and whether the identified warm-up removal reduces overall estimation error relative to the untreated series [17], [22]. Where post-truncation bias or MSE increases substantially, this indicates that the SPC-based truncation was overly conservative or misaligned with the actual transient structure of the system [21].

3. ETHICAL & SUSTAINABILITY CONSIDERATIONS

3.1. Introduction

This chapter examines the ethical, professional and sustainability considerations associated with the development of the automated warm-up detection tool. Although this project is software based and does not involve the physical risks or the material demands of workshop-based projects, it still carries important responsibilities. The app is designed to support the identification of the warm-up period from data acquired using discrete event simulation. As such, the recommendations made by the app may influence how the simulation results are interpreted and used in wider engineering decisions. For this reason, the project must be considered in terms of its impact on users, research quality and responsible engineering practice.

Attention is given to research integrity, professional responsibility, stakeholder impact, transparency and sustainability. As the tool is designed to make statistical warm-up analysis more accessible and consistent, it has the potential to improve simulation practice. However, it also introduces risks if outputs are over-relied upon or interpreted without sufficient understanding of the underlying methods. This chapter reflects on how the project aligns with ethical and sustainability principles, while also identifying limitations and areas where further development would be needed to support responsible long-term use.

3.2. Professional and Research Ethics

3.2.1. Research Integrity

Research integrity was an important consideration throughout this project because the purpose of the app is to support the analysis of simulation output and, in doing so, influence how that output is interpreted. This meant that the work could not just focus on whether the tool functioned technically, but also on whether the methods were represented and reported in an accurate way. The app was developed to apply established warm-up detection methods in a more structured and accessible format, but that did not remove the need to acknowledge that each method has its own assumptions, limitations, and scope of use. It was therefore important that the project did not present the app as producing a single unquestionable answer, but rather as a tool designed to support analysis in a more consistent way.

In practice, this required care both in the development of the app and in the writing of the thesis. The methods included in the tool had to be implemented in a way that remained consistent with their underlying logic, while the written discussion had to explain clearly what each method was doing and what its outputs could reasonably be taken to mean. Where limitations existed, these needed to be acknowledged rather than overlooked. This is particularly important because an overconfident presentation of the tool could create a

false impression of certainty, especially in cases where different methods might suggest different warm-up points or where the quality of the input data affects the reliability of the result.

It was also important to be realistic about the scope of the depth of this project. The tool was developed with the aim of improving the accessibility and consistency of the application of warm-up analysis methods, not to be used as a validated commercial software product. For this reason, it was important to distinguish what was achieved by this project and what areas require further testing or development in the future. This was important in a project of this kind, where the output of the app could appear more certain than it really is if the limitations of the underlying methods are not made clear. Maintaining research integrity therefore meant being honest about what the tool could do, where its boundaries lay, and how much interpretation remained the responsibility of the user.

3.2.2. Professional Responsibility in Engineering Analysis

Professional responsibility was an important consideration because the app was developed to support analysis that may ultimately inform wider engineering judgement. In steady-state simulation studies, the choice of warm-up period has a direct effect on the validity of the results that follow. If the warm-up point is selected poorly, biased data may be retained or useful data may be discarded, which can affect the quality of the conclusions drawn from the model. For that reason, the development of a tool to assist with warm-up detection carries the responsibility to ensure that the methods used are appropriate, that the outputs are presented clearly, and that the limits of the tool are not known to the user before the use of the tool.

this means the app should be understood as a decision-support tool rather than a replacement for technical judgement. While the purpose of the project was to make warm-up analysis more structured and accessible, it would not be professionally responsible to suggest that the output of the app should be accepted without question in every case. Different datasets may behave differently, methods may not always achieve the same answer, and the suitability of any recommended warm-up point still depends on the quality and nature of the simulation data being analysed. The responsibility of the user therefore remains central, particularly when interpreting results and deciding how much confidence can be placed in them.

This is closely related to the wider duty of engineers to act with competence, care, and integrity in the use of analytical methods. In this project, that meant selecting recognised warm-up detection techniques, implementing them as carefully as possible, and avoiding claims that went beyond what the tool could reliably support. It also meant recognising that even a relatively small software-based project can influence how evidence is interpreted in an engineering setting. Professional responsibility in this context was therefore not only about producing a working app, but about ensuring that the app supports sound analysis without encouraging overconfidence in automated results.

3.2.3. Transparency and Reproducibility

Transparency and reproducibility were important considerations in this project because the value of the app depends not only on whether it produces an output, but on whether that output can be understood, checked, and justified. Since the app applies statistical methods to support warm-up detection, it was important that the basis of each result was clear rather than hidden behind the interface. A user should be able to understand which method has been applied, what assumptions it relies on, and how the recommendation has been reached. Without this consideration, there is a risk that the app could be taken as being completely accurate which it is not.

Reproducibility is closely linked to this. If the same dataset is analysed under the same settings, the process should lead to the same result in a way that can be followed and explained. This is particularly important in an academic project, where the work must be open to review and where conclusions should be supported by a method that others could repeat for themselves. In this project, that meant ensuring that the logic of the implemented methods was clearly documented and that the relationship between the theoretical basis of each method and its practical application in the app was made explicit. The app was not intended to hide the analytical process, but to organise it in a more accessible and structured form.

Ethical Use of AI

In recent years, artificial intelligence has developed rapidly and has become increasingly integrated into many of the digital tools and platforms used in academic and professional work. As a result, its use is becoming more common within higher education, making it important that it is applied responsibly and within institutional guidelines. In line with DCU guidance, the use of AI in this project was restricted to the authorised tools Gemini and NotebookLM. These tools were used only in a supportive manner throughout the project. Their use was limited to helping with the structure of the thesis, improving the clarity of written sections, and assisting with the identification of relevant libraries and functions during software development. AI was not used to make engineering decisions, select the methodology, interpret results independently, or form the conclusions of the project. Any AI-generated output was treated only as support material and was critically checked before being included.

3.3.Stakeholder and Societal Impact

3.3.1. Primary Stakeholders

The primary stakeholders in this project are the users that would use the use the warm-up analyser and rely on its truncation suggestions. These include students, researchers and engineers working with Discrete Event Simulation who are looking for a more structured and accessible approach to the warm-up period identification. These stakeholders may benefit from a tool that simplifies the analysis process, improves

consistency in warm-up detection, and reduces the level of subjectivity involved in selecting truncation points. This could support more reliable simulation outputs and improve confidence in steady-state performance estimates.

At the same time, these stakeholders could be adversely affected if the analyser's recommendations are inaccurate or are applied without appropriate review. An incorrect truncation point may result in biased results, reduced accuracy, or false confidence in the simulation outputs. The tool should therefore be used as a support mechanism rather than a substitute for critical assessment. Responsibility remains with the user to evaluate the suitability of the recommendation in relation to the specific model and output behaviour being studied.

3.4. Potential Benefits

3.4.1. Potential Risk and Unintended Consequences

Despite the benefits of the Warm-Up Analyser, there are potential risks associated with its use. The main risk is that users may rely too heavily on the truncation suggestions without critically assessing whether they are appropriate for the system being analysed. If an unsuitable warm-up period is applied, biased data may remain in the results or too much valid data may be removed, reducing the accuracy of the final estimate.

There is also a risk that the implemented methods may not perform equally well for all simulation conditions. For example, some approaches may be less effective in low utilisation systems or in outputs with high variability, which could lead to less reliable recommendations. In addition, the use of an automated tool may lead some users to depend on the software rather than applying their own engineering judgement. The application should therefore be seen as a support tool, with final responsibility for interpretation and decision-making remaining with the user.

3.4.2. Accessibility

Accessibility was an important consideration in this project, as the Warm-Up Analyser was designed for use by students, researchers, and engineers with varying levels of experience in simulation and data analysis. A key aim of the application was to make warm-up period identification more approachable by reducing the need for manual analysis and specialist coding knowledge.

This was supported through the use of a clear user interface, simple workflow, and visual outputs that help the user interpret the results more easily. By presenting truncation suggestions and supporting graphs in an accessible format, the tool can make warm-up analysis more understandable and more practical for a wider

range of users. However, users must still apply critical judgement when reviewing the outputs, as accessibility should not come at the expense of proper interpretation.

3.5. Ethical Use of Automation

The project is built around automation, making the ethical use of automation a central issue. The app was designed to automate parts of the warm-up analysis process that would otherwise require manual processing and repeated calculation. This can be valuable, particularly where users need a more structured workflow. However, automation in this context should support human judgement, not replace it. The output of the app can assist the user in identifying a reasonable warm-up point, but it cannot remove the need to assess whether the recommendation makes sense for the dataset and simulation context being considered.

3.6. Sustainability Considerations

3.6.1. Environmental Impact

The overall environmental impact of this project was relatively low, as it focused on software development rather than physical manufacturing or material-intensive testing. The main impact came from electricity use during coding, testing, and running the application, as well as the wider energy demand associated with digital tools. The project was also resource efficient, as it relied mainly on existing software and digital data rather than physical materials. By automating part of the warm-up analysis process, the application can reduce manual effort and improve efficiency for users carrying out simulation studies.

3.6.2. Long Term Sustainability and Maintainability

The long-term sustainability of the project depends on how easily the software can be maintained and updated over time. As the application was developed in a structured and transparent way, it should be possible for future users to modify or extend it if required. From an economic perspective, the project is also sustainable, as it does not require specialised physical equipment and has the potential to save time for students, researchers, and engineers by simplifying simulation output analysis. This gives the tool practical value while keeping operating costs low.

4. THEORY

4.1. Introduction

This chapter outlines the theoretical basis of the three warm-up detection methods used in the automated warm-up detection tool. As the objective is to evaluate the steady-state performance of the system, data from multiple simulation replications are considered rather than relying on a single run. Averaging the

observations across replications at each time step reduces the influence of random variation in individual runs and produces a single representative output series for analysis. This averaged series forms the basis for each of the three warm-up detection methods considered in this work.

Y_{ji} is used to denote the observation point i from the replication j, where n is used to represent the total number of replications and m is the total number of timesteps per replication. The average at each time step therefore is found by the following:

$$\bar{Y}_i = \sum_{j=1}^{n} \frac{Y_{ji}}{n} \text{ for } i = 1, 2, \ldots, \text{m [21] [2]}$$

4.2. Welch's Method

Welch's method assists the user in the identification of the warm-up period by applying a moving average to the averaged output series to show the underlying trend of the system behaviour more clearly. Although the averaging process conducted to produce the $\bar{Y}$ for each value of i reduces some of the random variation present in individual replications, high-frequency oscillations may remain. A smoothing window is applied to the averaged series so that the transition from the initial transient period to steady-state behaviour can be more clearly observed.

A window size of w is first selected to determine the degree of smoothing applied to the averaged performance data. At the beginning of the series, a full centred window cannot be applied as there is not w number of observations available on either side of the point being considered. Because of this, Welch's method initially uses a growing window. This growing window is only used until there is enough data either side of the point in consideration to apply the steady Welch's method formula. The growing formula is denoted as the following:

$$\bar{Y}_i(w) = \frac{\sum_{s=-(i-1)}^{i-1} \bar{Y}_{i+s}}{2i-1} \quad \text{for } i = 1, 2, 3, \ldots, w \text{ [21] [2]}$$

The full centred moving average formula to be applied to the averaged data is applied to the data after $i = w+1$ is as follows:

$$\bar{Y}_i(w) = \frac{\sum_{s=-w}^{w} \bar{Y}_{i+s}}{2w+1} \qquad \text{for } i = w+1, w+2, \ldots, m-w \text{ [21] [2]}$$

The choice of window size affects the degree of smoothing on the data. Smaller w values show larger variations and larger w values produce a smoother overall trend. The smoothed series is then examined to identify the point beyond which no sustained trend is present, and the output fluctuates about a relatively stable level. This point is taken as the estimated warm-up period.

4.3.MSER5

MSER-5 identifies the warm-up period of the discrete event simulation output by selecting the truncation point that minimises the mean squared error of the data remaining after deletion. The first step in this method is to divide the averaged output series as described in 4.1 Introduction into non-overlapping complete batches of 5 observations. This is done by applying the following formula to the averaged data:

$$\bar{\bar{Y}}_j = \frac{1}{5} \sum_{i=5j-4}^{5j} \bar{Y}_i$$

Where:

- $\bar{\bar{Y}}_j$ is the jth batch mean
- $\bar{Y}_i$ is the averaged output at time step i

Once the batched series has been formed, successive truncation points are tested by progressively deleting batches from the start of the series. For each possible truncation point d, the MSER objective is evaluated using the remaining batched observations. The selected truncation point is taken as the value of $d$that minimises the MSER statistic:

$$d^* = \arg\min_{d \geq 0} \left[\frac{1}{(n-d)^2} \sum_{j=d+1}^{n} (\bar{\bar{Y}}_j - \bar{\bar{Y}}_{n,d})^2 \right]$$

Where:

- n is the total number of batch means
- $\bar{\bar{Y}}_{n,d}$ is the mean of the remaining batched observations after deletion up to point d.

The selected truncation point balances the removal of biased data against the retention of sufficient observations for reliable steady state estimation.

4.4.SPC

The Statistical Process Control (SPC) method is used to identify the point at which the simulation data can be considered statistically stable and representative of steady state behaviour. To begin the averaged data series $\bar{Y}_i$, introduced in Section 4.1, are put into non-overlapping batches, starting with a batch size of 1. This batching is conducted by using the following formula:

$$\bar{\bar{Y}}_j = \frac{1}{b} \sum_{i=bj-b+1}^{bj} \bar{Y}_i$$

Where:

- b is the batch size beginning with $b = 1$
- $\bar{\bar{Y}}_j$ is the j^{th} batched mean

The resulting batch means are then tested for approximate normality and independence using the Anderson Darling test and the Von Neumann test respectively. The calculation for the Anderson Darling test for normality for an ordered sample ($X_{(1)} \leq X_{(2)} \leq \cdots \leq X_{(n)}$) is as follows:

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}(2i-1)[\ln Z_i + \ln(1 - Z_{n+1-i})]$$

Where:

- A^2 is the Anderson-Darling Statistic
- n is the number of batched means present
- i is the time step
- X_i is the i^{th} ordered observation
- Z_i is the cumulative distribution function value of the fitted normal distribution evaluated at $X_{(i)}$
- Z_{n+1-i} is the cumulative distribution function value corresponding to the observation symmetrically positioned from the upper end of the ordered sample.

The Von Neumann's test for independence is carried out by applying the following to the batched means:

$$\frac{\delta^2}{s^2} = \frac{\frac{1}{n-1}\sum_{i=1}^{n-1}\left(\bar{\bar{Y}}_{j+1} - \bar{\bar{Y}}_j\right)^2}{\frac{1}{n}\sum_{i=1}^{n}\left(\bar{\bar{Y}}_j - \bar{\bar{Y}}\right)^2}$$

Where:

- δ^2 is the difference between successive mean squares
- s^2 is the sample variance of the batched means
- n is the number of observations within each batch
- $\bar{\bar{Y}}_j$ is the mean of the j^{th} batch
- $\bar{\bar{Y}}$ is the overall mean of the batched means

Once a batch size is found that passes both the Anderson Darling and Von Neumann Tests, the average and control limits are calculated using the data from the data in the second half of the batched means. The control limits are calculated using:

$$CL = \hat{\mu} \pm \frac{z\hat{\sigma}}{\sqrt{n}} \text{ for } z = 1, 2, 3 \text{ [21] [22]}$$

With $\hat{\mu}$ being the estimated mean calculated from the second half of the batched data:

$$\hat{\mu} = \frac{\sum_{h=b-\left\lfloor \frac{b}{2} \right\rfloor+1}^{b} \bar{Y}_h}{\lfloor b/2 \rfloor} \text{ [22] [6]}$$

And σ being the estimated standard deviation calculated from the second half of the batched means.

$$\hat{\sigma} = \sqrt{\frac{1}{\left\lfloor \frac{n}{2} \right\rfloor} \sum_{h=n-\left\lfloor \frac{n}{2} \right\rfloor+1}^{n} s_h^2}$$

- n is batch size
- s_h^2 is the standard deviation about the individual means

The system is deemed to be 'out of control' if any of the following are present in the batched data when graphed against the control limits:

- A single point plots outside a 3σ control limit
- Two of three consecutive points plot outside a 2σ control limit
- Four of five consecutive points plot outside a 1σ control limit
- Eight consecutive points plot on one side of the mean
- All initial points plot to one side of the mean

4.5. Validation Metrics for the Warm-up Removal

4.5.1. Analytical M/M/1 Performance Measures

For a simple M/M/1 queue, the average queue length, L_q, and the average number of items in the system (work in progress, WIP), L, can be calculated analytically using the rate of arrival and the service rate of the system. These used as reference values when comparing the raw data to the post-truncation data. These measures provide a known steady-state baseline, allowing the validation of the warm-up period removal process. Firstly, the utilisation (ρ) of the system must be calculated. This is done by the following:

$$\rho = \frac{\lambda}{\mu} \quad [2], [34]$$

Where:

- ρ is the utilisation, (ρ<1)
- λ is the arrival rate into the system
- μ is the service rate

After the utilisation of the system has been calculated, the average queue length and the WIP can be calculated by the following.

$$L_q = \frac{\rho^2}{1-\rho} \quad [2]$$

And

$$L = \frac{\rho}{1-\rho} \quad [2]$$

Where:

- L_q is the average number of items in the queue
- L is the average number of items in the system

4.5.2. Error and Accuracy Calculations

To assess the effectiveness of the warm-up period removal methods, the post-truncation simulation estimates were compared with the analytical M/M/1 benchmark values presented in Chapter 4.5.1. Error and accuracy measures were used to quantify how closely the simulation results matched the known steady-state values. These measures were bias, percentage bias, percentage error, variance, and mean squared error (MSE). These calculations provide a basis for evaluating both the accuracy and the consistency of the simulation estimates before and after warm-up removal. (note: $X_{Analytical}$ can represent either L or L_q, depending on which metric is being compared.)

4.5.2.1. Bias

Firstly, bias was calculated by finding the difference between the mean of the simulation estimates and the analytical value. This was used to determine whether the simulation output overestimated or underestimated the expected steady-state result. This was calculated by:

$$\text{Bias} = \bar{X} - X_{Analytical}$$

Where:

- $\bar{X}$ is the average of the replication estimates
- $X_{Analytical}$ is the analytical steady-state value

A positive bias means that the simulation estimate is greater than the analytical value, while a negative bias indicates that it is lower.

Percentage bias was also calculated to show the difference, relative to the analytical value. This allows multiple utilisations to be compared on the same scale. Percentage bias is calculated by:

$$\text{Percentage Bias} = \frac{\bar{X} - X_{Analytical}}{X_{Analytical}} \times 100$$

Where:

- $\bar{X}$ is the average of the replication estimates
- $X_{Analytical}$ is the analytical steady-state value

4.5.2.2. Variance

The variance of the replication estimates was calculated to show how much the results varied between simulation runs. This gives an indication of how consistent the estimates were after the warm-up period had been removed. This was done by:

$$s^2 = \frac{1}{n-2}\sum_{i=1}^{n}(\bar{X}_i - \bar{X})^2 \ [28]$$

Where:

- s^2 is the sample variance of the replication averages
- $\bar{X}_i$ is the average from replication i
- $\bar{X}$ is the total mean of the system
- n is the total number of replications

4.5.2.3. Mean Squared Error

The mean squared error (MSE) was calculated to combine both the bias of the estimate and the variation across replications into a single performance measure. This is relevant to show the trade-off between reducing initialization bias and retaining enough data to keep the estimate stable. Mean squared error

therefore provides a useful overall measure of performance, as it penalises both systematic error and excessive variability. The MSE was calculated by the following:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - X_{Analytical}\right)$$

Where:

- $\bar{X}_i$ is the average from replication i
- $X_{Analytical}$ *is the analytical steady state value*
- n is the total number of replications

5. METHODOLOGY

5.1. Introduction

This chapter outlines the steps taken to develop the automated Warm-up Analysis Tool for discrete event simulation data. The work involved in the design and implementation of an application, designed using R Shiny, capable of both handling imported time series performance metrics generated from ExtendSim models and applying studied techniques to identify the appropriate warm-up period. The objective of the tool is to automate the identification and the removal of initialisation bias from the output data using established warm-up identification techniques, resulting in improved reliability of the performance measures obtained from the simulation studies.

The development of the tool involved several stages. First, the overall architecture of the application and the user interface were designed to allow users to import the simulation data obtained from ExtendSim databases. Procedures were then implemented in the application to manage and process the output data to ensure that it could be properly processed by the different analysis algorithms. Several warm-up detection methods were then implemented within the application, these include Welch's graphical method, the Mean Squared Error Reduction (MSER5) approach and a Statistical Process Control (SPC) based method.

5.2. Application Architecture

The warm-up analysis tool was developed as an interactive application using the R Shiny framework. The architecture of the application was designed to support the full analysis workflow, from importing the time-series performance metric data to applying warm-up detection methods and presenting the resulting data visually. To achieve this, the application was structured into several components responsible for data input, preprocessing, execution of the detection methods and visualisation of results

A modular design approach was adopted to separate the user interface, data handling procedures and the method implementations. This structure enabled methods to be developed independently of the main application logic, enabling easier modification and extension of the code.

5.2.1. Platform Selection

The warm-up analysis tool was implemented using the R programming language and the Shiny web application framework. R was selected as the primary development environment due to its strong capabilities in statistical analysis and data processing, central requirements for implementing the warm-up detection methods. The language provides a wide range of libraries for time-series analysis, data manipulation and statistical computations, making it well suited for the processing of simulation output data.

The Shiny framework was chosen to enable the development of an interactive, graphical user interface directly within the R environment. This allowed the statistical analysis methods and the visualisation components of the tool to be integrated in a single platform, simplifying both development and maintenance of the application. Shiny also enables the creation of browser based interactive interfaces, allowing users to upload data, select analysis methods, and visualise results without requiring direct interaction with the underlying code.

Alternative environments such as Python-based frameworks were considered during the design process. However, to achieve the same interactive application that is possible on R Shiny, Python requires the combination of multiple libraries and web frameworks to support both the statistical analysis and the graphical interface. These factors would increase the complexity of the development and would require additional configuration to integrate the analytical and user interface components into a single application. In contrast, R Shiny provides native support for both statistical analysis and interactive application development, allowing the data processing, algorithm implementation and visualisation components to be integrated within a single framework.

Spreadsheet based tools such as Excel was also considered during the design process. Excel provides strong capabilities for numerical calculations and data manipulation and is often used for performing statistical analysis on simulation outputs. Throughout this project Excel was used to implement individual versions of each of the detection methods for verification purposes, allowing the results produced by the application to be cross-checked against independent calculations.

Excel was not selected as the primary platform for the development of the analysis tool as, while it is well suited to performing individual calculations, it is less capable for implementing an integrated analytical application that combines data import, warm-up detection method application and graphical output within

a single interactive interface. The development of a comparable tool on Excel would require the use of macros or Visual Basic for Applications (VBA) to manage user interaction and automate the workflow. As a result, a dedicated programming environment was considered more suitable for implementing the automated warm-up analysis tool, while Excel remained a useful tool for the validation of the algorithms in the application.

Criteria	Weight	R / R Shiny		Python		Excel / VBA	
		Score	Weighted score	Score	Weighted score	Score	Weighted score
Statistical analysis capability	10	5	50	5	50	5	50
Interactive UI Capabilities	9	5	45	4	36	3	27
Integration of analysis and interface	9	5	45	4	36	3	27
Availability of required libraries	7	5	35	4	28	3	21
Maintainability and extensibility	6	5	30	4	24	3	18
Suitability for validation	5	3	15	3	15	5	25
TOTALS		220		189		168	

Figure 1: Automation Method Selection Decision Matrix

5.2.2. R Libraries Used

Several R libraries had to be imported into the application to support its full functionality. The shiny package provides the core functions for constructing the interactive user interface, managing the reactive server logic and enabling the downloadable output. The shinyjs package was included to support interface control features, specifically allowing the user to clear the current session and return to the upload tab without restarting the application, using the shinyjs::reset() function to clear the file input area and updateTabsetPanel() to redirect to the upload tab.

The importing of data was supported by the 'readr' and the 'readxl' packages. The 'readr' library was used to support the importation of CSV files using the 'read_csv()' function and the exporting of the cleaned dataset using the 'write_csv()' function. The 'readxl' library was used to allow users to also upload Excel files to the app, using the 'read_excel' function. The graphs produced throughout the application such as the various method graphs were generated using the 'ggplot2' library.

In addition to these libraries being used in the main 'App.R' page, the 'nortest' package was also used in the 'SPC.R' script. This was used due its having the Anderson-Darling test function within the package.

```
1. library(shiny)
2. library(shinyjs)
3. library(readr)
4. library(readxl)
```

```
5. library(ggplot2)
```

5.2.3. Modular Application Design

A modular design approach was adopted in the development of the warm-up analysis tool. Each warm-up detection method was implemented as an independent R function contained within a separate script file: welch_apply() in welch.R, mser_apply() in mser5.R and spc_apply() in SPC.R. These scripts are loaded into the main application at runtime using source() calls wrapped in a tryCatch() block, allowing the analysis methods to be executed within the application environment while remaining independent of the core interface and data management logic. The separation between the detection method functions and the main App.R file meant that each method could be developed, tested and updated independently.

```
1. #load methods
2. tryCatch({
3.   source("welch.R",local=TRUE)
4.   source("mser5.R",local=TRUE)
5.   source("SPC.R",local=TRUE)
6. })
```

5.2.4. User Interface Structure

The user interface was implemented using the navbarPage() element of the Shiny framework, which organises the application into a series of named tabs accessible from a navigation bar. Six tabs were implemented: Instructions, Upload Data, Welch Analysis, MSER-5 Analysis, SPC Analysis and Combined Analysis. Each tab corresponds to a distinct stage of the analysis workflow, guiding the user from data upload through to comparison and download.

```
1. ui <- navbarPage(
2. title="Warm-up Analyser"…..
3.   tabPanel("Instructions",value="instructions_tab",…….),
4.   tabPanel("Upload Data",value="upload_tab",…….),
5.   tabPanel("Welch Analysis",value="welch_tab",…….),
6.   tabPanel("MSER-5 Analysis",value="mser_tab",…….),
7.   tabPanel("SPC Analysis",value="spc_tab",…….)
8. )
```

Figure 2: Example of How Tabs are Made

The Instructions tab provides guidance on the expected data format and the operation of the application. The Upload Data tab provides a fileInput() control that accepts CSV and Excel files, a numericInput() for specification of the timestep used in hours, and two checkboxInput() controls allowing the user to indicate whether the first row is a column header and whether the first column is a timestep identifier. A single

actionButton() labelled Scrap and Restart appears on every tab and calls shinyjs::reset() on the upload area followed by updateTabsetPanel() to return the user to the upload tab, clearing all stored data in the process.

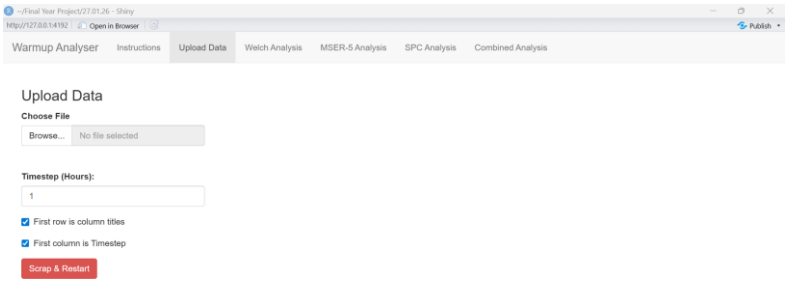


Figure 3: App Upload Page

The individual method tabs each contain a plotOutput() for the graphical result, together with any method-specific controls. The Welch tab includes a numericInput() to allow the window size to be change by the user depending on the amount of data being handled. The MSER-5 tab displays the recommended cutoff time as a textOutput() label in red. The SPC tab displays the passing batch size as a textOutput() label in blue. The Combined Analysis tab contains both the comparison plot and a fluidRow() with a selectInput() for choosing the truncation basis and a downloadButton() for exporting the cleaned dataset.

5.2.5. Application Structure

The computational logic of the application is contained within the server function and follows Shiny's reactive programming model. A central reactiveValues() object named data_store holds the uploaded raw data, the pre-processed clean data frame and the results returned by each of the three detection methods. This ensures that all method outputs are computed from the same preprocessed data and are consistent with each other.

The primary data loading and method execution is triggered by an observeEvent() that monitors both the file_upload input and the welch_window input simultaneously, using list() to combine both triggers. This means that whenever a new file is uploaded or the window size is changed, all three methods are recomputed automatically. Within this, the uploaded file is read using read_csv() or read_excel() depending on the file type uploaded, the timestep column is removed if the corresponding checkbox is ticked, all columns are coerced to numeric using lapply() with as.numeric(), and the ensemble mean is computed using rowMeans(). The three method functions are then called and their return values stored in data_store.

5.3.Data Processing and Pre-Processing

The warm-up analysis tool was designed to accept time-series performance data generated from ExtendSim simulation models. Prior to analysis, the data must be exported from the simulation and formatted into a structure compatible with the application. Each column of the dataset represents an independent replication of the simulation, and each row corresponds to a timestep within the simulation run. An optional first column may contain timestep identifiers and an optional first row may contain column headers.

After the dataset has been uploaded, the application performs a series of preprocessing steps. The file extension is extracted using tools::file_ext() and the appropriate read function is selected accordingly. If the file_has_header checkbox is ticked, the first row is used as column names; otherwise columns are given default names by the read function. If the file_has_timestep_column checkbox is ticked and the data frame has more than one column, the first column is dropped using df[,-1,drop=FALSE], removing the timestep identifier from the numeric data. All remaining columns are then coerced to numeric values using lapply() with as.numeric(as.character(x)), which handles any columns that were imported as character strings due to formatting in the source file.

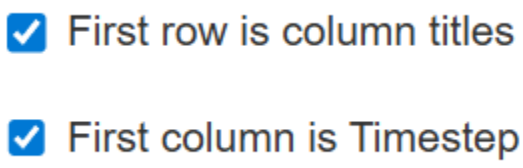


Figure 4: User Option to Select Uploaded Data Format

The preprocessed data is stored in data_store$clean and passed to all three detection method functions. The average value for each timestep across replications is computed from this data frame using rowMeans(data_store$clean, na.rm=TRUE) and stored as the variable ensemble. The number of columns in the clean data frame, representing the number of replications, is stored as number_of_runs and passed to the SPC function where it is used in the control limit calculation.

```
1.      #check if the first column is a timestep
2.      clean_df <- if(input$file_has_timestep_column && ncol(df) > 1) df[,-1,drop=FALSE] else df
3.
4.      clean_df[] <- lapply(clean_df,function(x) as.numeric(as.character(x)))
5.      data_store$clean <- clean_df
```

Figure 5: Code Checking for if First Column is Timestep

5.4. Warm-up Detection Methods

5.4.1. Selection of Warm-Up Detection Methods

A range of techniques have been proposed in the literature for identifying the warm-up period in steady state simulation outputs. These methods attempt to determine the point at which the effects of the initial conditions no longer influence the observed performance measures. Several approaches exist for addressing this problem, including graphical methods, statistical truncation rules, and hypothesis testing techniques.

Among the methods discussed in the literature are batch means approaches, randomisation tests, and other statistical techniques designed to detect steady-state behaviour in time-series data. Batch means methods involve dividing the simulation output into batches and analysing the stability of the batch averages. While these techniques can be effective for analysing steady-state data, they are primarily intended for estimating steady-state performance measures rather than directly identifying the warm-up period. As a result, their implementation within an automated warm-up detection tool is less straightforward.

Randomisation and hypothesis testing approaches have also been examined for identifying steady-state behaviour in simulation outputs. These methods generally involve statistical testing procedures that compare segments of the time-series data to determine whether the system has reached a stable operating condition. However, such techniques often require more complex statistical procedures and can be computationally intensive when applied to large simulation datasets.

Three warm-up detection methods were selected for implementation within the warm-up analysis tool: Welch's graphical method, the Mean Squared Error Reduction (MSER5) approach, and a Statistical Process Control (SPC) based method. These techniques were chosen because they are all both reliable in the identification of steady-state behaviour in simulation outputs and suitable for automation in the R Shiny environment.

Welch's method provides a graphical approach based on smoothing the ensemble average of multiple simulation replications, allowing the stabilisation of the system behaviour to be visually identified. The MSER-5 method provides an automated statistical approach that determines the truncation point by minimising the mean squared error of the remaining observations. The SPC-based method applies principles of statistical process control to detect when the time-series data stabilises within statistically controlled limits.

The implementation of multiple detection methods within the analysis tool allows the recommended warm-up periods generated by each technique to be compared. Consistency between the outputs of the different

methods provides greater confidence that the identified truncation point accurately represents the transition from transient behaviour to steady-state system performance.

Criteria	Weight	Welch's Method		Randomisation Test		SPC		MSER5		Cumulative Mean Rule	
		Score	Weighted Score	Score	Weighted Score	Score	Weighted Score	Score	Weighted Score	Score	Weighted Score
Automation Suitability	10	3	30	3	30	4	40	5	50	3	30
Ease of Implementation	9	4	36	2	18	3	27	4	36	3	27
Iterpretability of Results	8	5	40	3	24	4	32	4	32	4	32
Computational Efficiency	7	3	21	2	14	3	21	5	35	2	14
Acessibility	5	5	25	2	10	3	15	4	20	4	20
TOTALS		152		96		135		173		123	

Figure 6: Warm-up Detection Method Decision Matrix

5.4.2. Welch's Method

5.4.2.1. Overview

Welch's Method starts by averaging the data across all uploaded replications to create a single "ensemble" sequence, which helps to cancel out the random "noise" inherent in individual simulation runs. To further clarify the underlying trend, a moving average smoother is applied to this averaged data. This smoothing process involves replacing each data point with the average of the observations within a surrounding window of a specified size.

In the application, this smoothed series is plotted to allow for "eyeball" validation. The goal is to identify the specific time index where the initial transient fluctuations end and the series begins to oscillate around a constant horizontal mean. By adjusting the smoothing window, the user can filter out short-term volatility to more clearly see the point where the system's long-run behaviour begins.

5.4.2.2. Code Runthrough

Welch's graphical method was implemented in the welch_apply() function contained in welch.R. The function accepts the preprocessed data frame as its input, where each column is one replication, together with the window size w as a named parameter defaulting to 20.

The first step is to compute the ensemble mean across all replication columns at each timestep. This is done using rowMeans(df, na.rm=TRUE), which returns a numeric vector y_bar of length m, equal to the number of timesteps. The na.rm=TRUE argument ensures that any missing values in individual replications do not prevent the ensemble mean from being computed at that timestep.

The moving average is then computed in a for loop over all indices i from 1 to m. Three cases are handled. For i <= w, the mean is computed over observations y_bar[1:(2*i-1)] using mean(y_bar[1:(2*i1)], na.rm=TRUE). This implements the growing one-sided window formula from Chapter 4, where 2i1

observations are used symmetrically around position i near the start of the series. For w < i <= m-w, the mean is computed over y_bar[(i-w):(i+w)], giving 2w+1 observations in the symmetric centred window. For i > m-w, the moving average element is left as NA, since a full centred window cannot be formed when fewer than w future observations remain. Setting these tail values to NA is important because it prevents the smoothed curve from appearing to flatten artificially near the end of the series due to a reduced window.

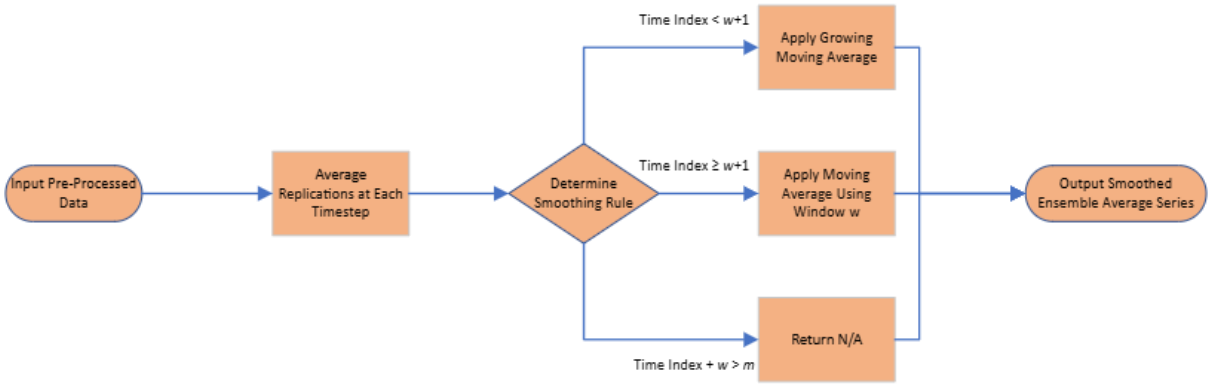


Figure 7: Welch's Method Flowchart

The function returns the input data frame with two new columns added at the end: Welch_Mean, containing the vector containing the means at each timestep, and Welch_MovingAvg, containing the moving average vector ma. In App.R, the returned data frame is stored in data_store$welch and retrieved in the renderPlot() call for the Welch tab. A time axis is created by multiplying the row index (from 0 to nrow-1) by the timestep_value input, and ggplot() is used to draw two line series, the raw ensemble mean in grey and the smoothed moving average in black.

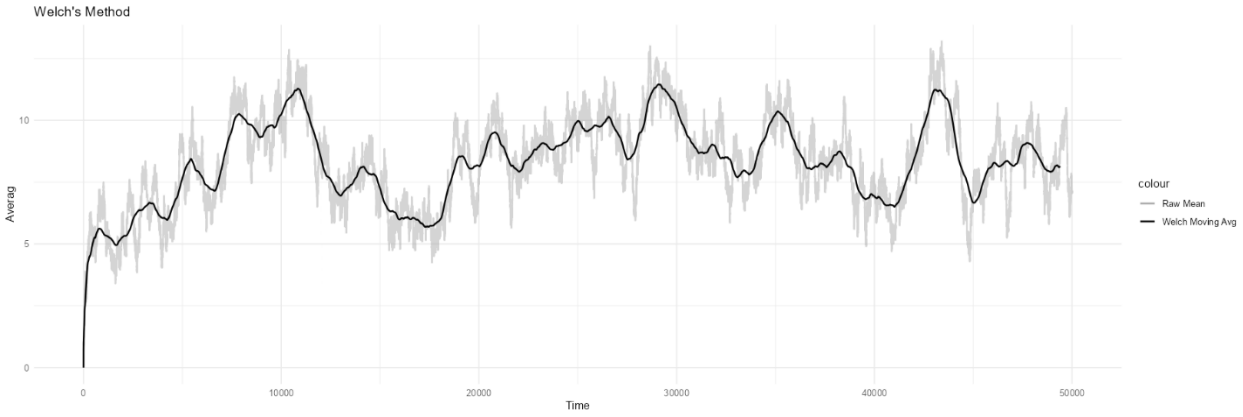


Figure 8: Example of Welch's Method Displayed in the Welch's Analysis Tab

5.4.3. MSER-5

5.4.3.1. Overview

The MSER-5 method provides an automated, objective way to find the truncation point that minimizes the bias in the steady-state mean. The process begins by grouping the raw simulation output into non-overlapping batches of five observations each. This initial batching helps to reduce the impact of high-frequency noise and autocorrelation.

The App then performs a systematic search through the data, testing every possible truncation point. For each point, it calculates the "Mean Squared Error" of the remaining data. The logic is designed to find a balance: it looks for a point that removes enough early "biased" data to flatten the mean, but not so much data that the estimate becomes unstable due to a small sample size. The application identifies the optimal warm-up period as the point that produces the minimum MSER value, effectively "punishing" both under-truncation (high bias) and over-truncation (high variance).

5.4.3.2. Code Runthrough

The MSER-5 method was implemented in the mser_apply() function in mser5.R. The function accepts the ensemble mean vector as its primary input, together with three guard parameters: the batch size b defaulting to 5 and min_batches_remaining defaulting to 10. The function begins by coercing the input to a numeric vector using as.numeric() and removing NA values. The total length n_total is checked against the minimum viable threshold b * (min_batches_remaining + 2). If the series is too short, the function returns immediately with cutoff = 0L and an empty stats_vector, preventing downstream errors in the application.

The series is then truncated to floor(n_total / b) * b observations, ensuring that the number of observations is an exact multiple of the batch size. The batch means are computed by forming a matrix using matrix(y_use, nrow=b, byrow=FALSE), which arranges the data so that each column contains b consecutive observations from one batch, and then applying colMeans(batch_matrix) to compute the mean of each column. This produces the batch means vector of length m = floor(n_total / b).

The maximum allowable truncation point max_d is computed as min(floor(max_frac * m) - 1, m - min_batches_remaining - 1), which simultaneously enforces both the fraction-based and the minimumremaining-batches constraints. The MSER statistic is then evaluated in a for loop over d from 0 to max_d. For each d, the remaining batch means are extracted as batch_means[(d+1):m], their length L and mean mu are computed using length() and mean() respectively, the sum of squared deviations sse is computed using sum((remaining - mu)^2), and the statistic is stored as sse / (L^2) in the mser_stats vector. The optimal truncation point best_d is identified using which.min(mser_stats) - 1, and the corresponding number of original observations to discard is computed as best_d * b.

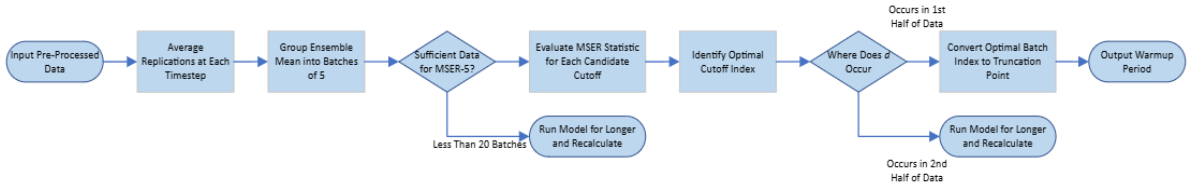


Figure 9: MSER-5 Method Flowchart

The function returns a list containing the integer cutoff value and the complete mser_stats vector. In App.R, the cutoff is stored in data_store$mser and used both to generate the textOutput() label showing the recommended cutoff in hours (computed as data_store$mser$cutoff * input$timestep_value) and to draw a vertical red dashed line on the MSER plot at the position cutoff_d = data_store$mser$cutoff / 5, which is the batch index of the optimal truncation point.

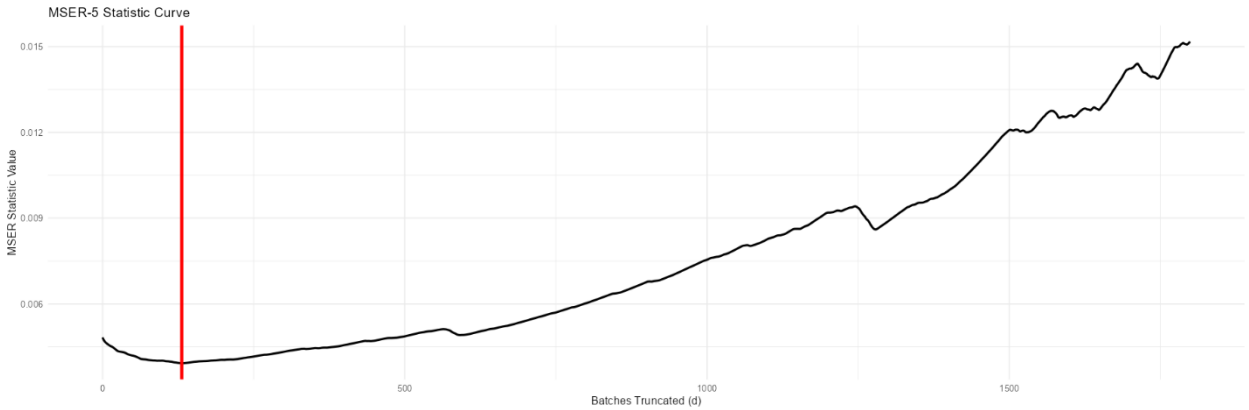


Figure 10: Example of MSER5 Results on App

5.4.4. Statistical Process Control (SPC)

5.4.4.1. Overview

The SPC approach treats the simulation output as a process that must reach a state of "statistical control" to be considered valid. The process begins by calculating the ensemble average across all simulation replications, resulting in a single time-series where each data point represents the mean value for that specific timestep. Because SPC requires independent and normally distributed data, the application uses an iterative batching procedure. It groups the data into increasingly larger batches until the batch means pass two specific statistical hurdles: the Von Neumann test for independence and the Anderson-Darling test for normality.

Once a valid batch size is found, the second half of the batched data is used to calculate "steady-state" control limits (mean ± 1, 2, and 3 standard deviations). The algorithm then scans the data from the beginning and applies a set of decision rules (outlined in 4.4 SPC). The warm-up period is identified as the point following the last "out-of-control" violation, marking the moment the simulation successfully transitioned into a stable operating state.

5.4.4.2. Code Runthrough

The SPC-based warm-up detection method was implemented in the spc_apply() function in SPC.R. The function accepts the ensemble mean vector and the number of simulation runs as inputs and returns a structured list containing the selected batch size, batch means, control chart limits, estimated steadystate mean, cutoff batch index and cutoff time.

The function begins by coercing the input vector to numeric using as.numeric(na.omit(data_vector)) and storing the total length as n_total. A while loop then iterates over candidate batch sizes starting from batch_size = 1, incrementing by 1 at each step, until both stationarity tests are passed or the constraint batch_size < n_total / 20 is violated. The upper limit of n_total / 20 ensures that at least 20 batches remain for control chart construction, consistent with guidance from Law [2] on minimum batch counts for reliable simulation output analysis.

Within each iteration of the while loop, the data is batched using matrix(y_use, nrow=batch_size, byrow=FALSE) and batch means b_means are computed using colMeans(). The Von Neumann ratio statistic is then computed as rvn = (sum(diff(b_means)^2) / (n-1)) / (sum((b_means - mean(b_means))^2) / n), where n = length(b_means). The p-value for the independence test is obtained from the two-tailed normal approximation vn_p = 2 * (1 - pnorm(abs((rvn - 2) / sqrt(4/n)))), which uses the fact that under the null hypothesis of independence the Von Neumann ratio is approximately normally distributed with mean 2 and variance 4/n [54]. The normality of the batch means is tested using ad_p = nortest::ad.test(b_means)$p.value, which returns the p-value from the Anderson-Darling test.

If both vn_p > 0.05 and ad_p > 0.05, the current batch size is accepted and the function proceeds to construct the control chart. The steady state mean and standard deviation are estimated from the second half of the batch means using the indices second_half_data = floor(n/2):n. The estimated mean mu_est is computed using mean(steady_means) and the standard deviation sigma_est using sd(steady_means). The control limits at one, two and three standard deviations are then computed as u1 = mu_est + sigma_est, u2 = mu_est + 2*sigma_est, u3 = mu_est + 3*sigma_est and the corresponding lower limits. If neither test is passed, the batch size is incremented by 1 and the loop continues.

Once the control limits have been established, the rules outlined in 4.4 SPC are applied to identify out-of-control observations. A logical vector is_out of length n is initialised to FALSE. The upper and lower sides are checked independently to ensure the rule applies only when violations are on the same side of the centreline.

The final warm-up cutoff is determined by extracting the indices of all out-of-control observations using out_of_limits = which(is_out). If no violations are found, cutoff_batch is set to 1. Otherwise, last_out_of_control = max(out_of_limits). If last_out_of_control equals n, meaning the final observation is still out of control, cutoff_batch is set to NA and the application displays a warning that steady state was not reached. Otherwise, cutoff_batch = last_out_of_control + 1. The cutoff time in original observation units is computed as cutoff_t = (cutoff_batch - 1) * batch_size

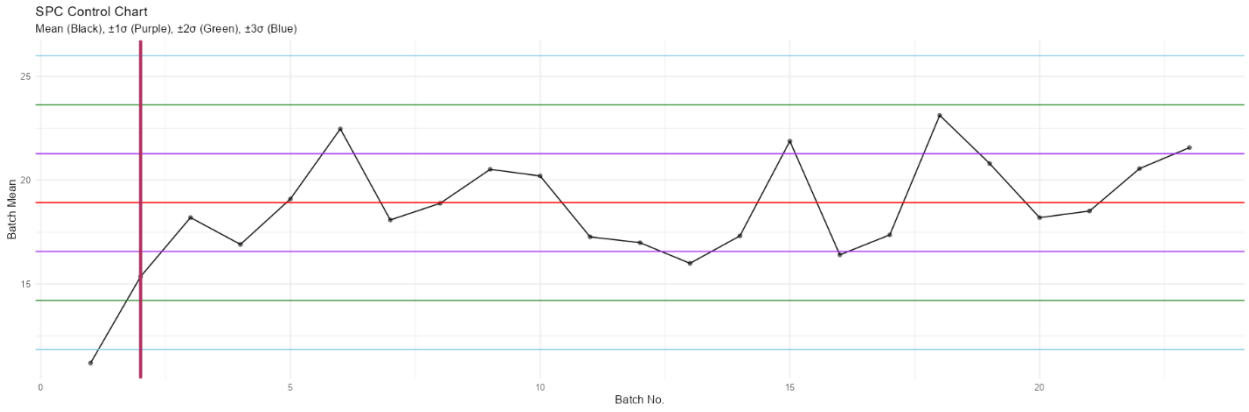


Figure 11: Example SPC in Application

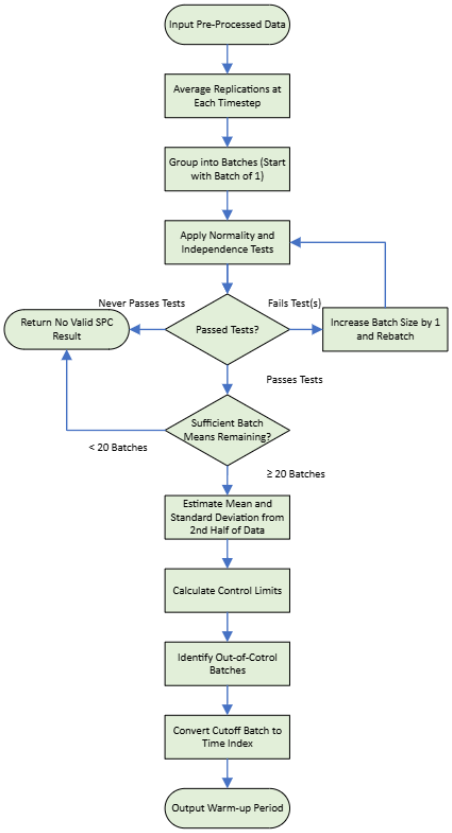


Figure 12: SPC Method Flowchart

5.5. Method Comparisons Tab

A tab dedicated to comparing the three warm-up detection methods was included in the tool. The purpose of this tab is to present all three method results in a single combined view, allowing the user to assess the consistency of the recommended warm-up periods and make an informed decision about which truncation point to apply to their dataset. The Combined Analysis tab is implemented as a separate tabPanel() within the navbarPage() user interface structure, following the same pattern as the individual method tabs. The tab contains a single combined graph generated using plotOutput() and renderPlot(), which displays the Welch smoothed ensemble mean curve as a background series in blue. Vertical lines are then overlaid on this curve using geom_vline() to indicate the truncation points recommended by MSER-5 and SPC: the MSER-5 recommendation is shown in green and the SPC recommendation is shown in dark red. Text labels positioned at each vertical line using annotate() identify which method produced which recommendation.

Where the vertical lines fall close together, the three methods are broadly consistent and the user can proceed with confidence. Where the lines are widely separated, this disagreement indicates that the transient behaviour is complex or that the run length may not be sufficient for all methods to produce reliable estimates.

Below the combined graph, a selectInput() dropdown allows the user to choose which method's truncation point to use as the basis for the downloadable cleaned dataset, with MSER-5 and SPC available as options. The downloadHandler() generates a CSV file using write_csv() containing only the rows of the original uploaded dataset that fall after the selected truncation point, with a filename that includes the name of the selected method for traceability

```
#Data Export
output$download_truncated <- downloadHandler(
  filename=function() { paste0("Cleaned ",input$trunc_method,".csv") },
  content=function(file) {
    req(data_store$raw)

    drop_until <- if(input$trunc_method == "MSER-5") {
      data_store$mser$cutoff
    } else {
      if(is.na(data_store$spc$cutoff_time)) 0 else data_store$spc$cutoff_time
    }

    if(is.null(drop_until) || drop_until >= nrow(data_store$raw) || drop_until <= 0) {
      write_csv(data_store$raw,file)
    } else {
      write_csv(data_store$raw[(drop_until+1):nrow(data_store$raw),],file)
    }
  }
)
```

Figure 13 Code for Downloading "Cleaned Data"

5.6.ExtendSim Discrete Event Simulation Models

5.6.1. Simulation Model Configuration

Models were constructed in ExtendSim using a basic single-server queueing structure consistent with M/M/1 assumptions. Arrivals were generated using a Create block and service was represented using an Activity block. Both the interarrival times and service times were set to follow exponential distributions, in keeping with standard M/M/1 queueing theory. This ensured that the ExtendSim models matched the assumptions used in the corresponding analytical formulations.

Several simple M/M/1 models were created, each with a different utilisation level. This was achieved by varying the relationship between the arrival rate and service rate while maintaining the same overall model structure. The use of multiple utilisations allowed the warm-up detection methods to be examined under conditions of both low and high congestion.

For each model, work-in-progress (WIP) and queue length were recorded as the primary output measures. These performance measures were selected because analytical steady-state values can be calculated for them, allowing a direct comparison between the simulation output and the theoretical benchmark. If the post-warm-up simulation estimates were found to be close to the corresponding analytical values, this was taken as evidence that the selected warm-up period was appropriate and that the implemented detection method was performing effectively.

5.6.2. Data Export Method

To enable consistent analysis within the developed application, simulation output data was collected from ExtendSim models using a structured and repeatable approach. A regular timestep was required to ensure that the output data could be treated as a time-series suitable for the implemented warmup detection methods. This was achieved using a Pulse block, configured to trigger at a fixed time interval. The current simulation time was divided by the same interval used in the Pulse block, producing a uniform timestep index for each recorded observation. This ensured that data points were recorded at consistent intervals throughout the simulation run.

Work-in-progress (WIP) was calculated indirectly using system flow information. The total number of entities that had entered the system was obtained from the Information block # output, while the total number of entities that had exited the system was obtained from the Exit block. WIP was then determined as the difference between these two values, representing the number of entities currently within the system at each timestep. This approach ensured that WIP was tracked continuously and reflected the dynamic state of the system over time.

Queue length data was obtained directly from the Queue block using its Length output, which provides the instantaneous number of entities waiting in the queue. This measure was recorded at each timestep alongside the WIP data, allowing both performance metrics to be analysed consistently.

The combination of a fixed timestep, calculated WIP, and directly measured queue length ensured that the exported dataset was in a suitable format for subsequent statistical analysis within the developed warmup detection tool.

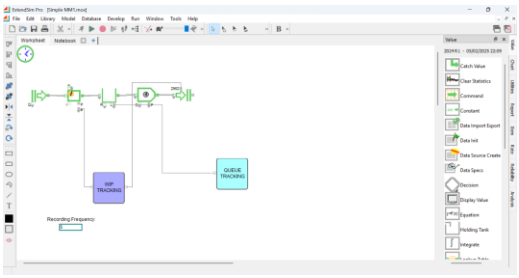


Figure 14: Sample M/M/1 Queue

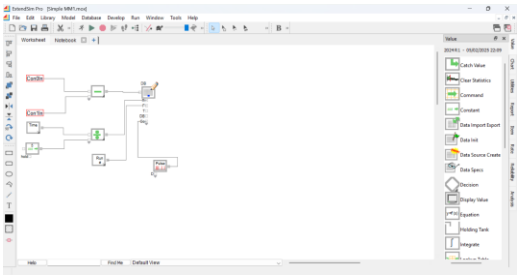


Figure 15: Method for Recording WIP

6. RESULTS & DISCUSSION

6.1. Test 1 – Low Utilisation M/M/1 (WIP)

To begin the testing on the application, a low utilisation, simple M/M/1 queue was initially tested with the following conditions:

- Mean Time Between Arrivals = 10
- Mean Service Time = 5
- Run Time = 20,000 hrs
- Recording Frequency = Every 1 Hr
- No. of Replications = 20

From the formula outlined in 4.5.1 Analytical M/M/1 Performance Measures the following table can be made:

MTBA	MST	Utilisation	Expected WIP	Expected Queue Length
10	5	0.5	1	0.5

Figure 16: Test 1 Conditions

The App displays the following graphs for the three methods applied:

6.1.1. Welch's Method (w=500)

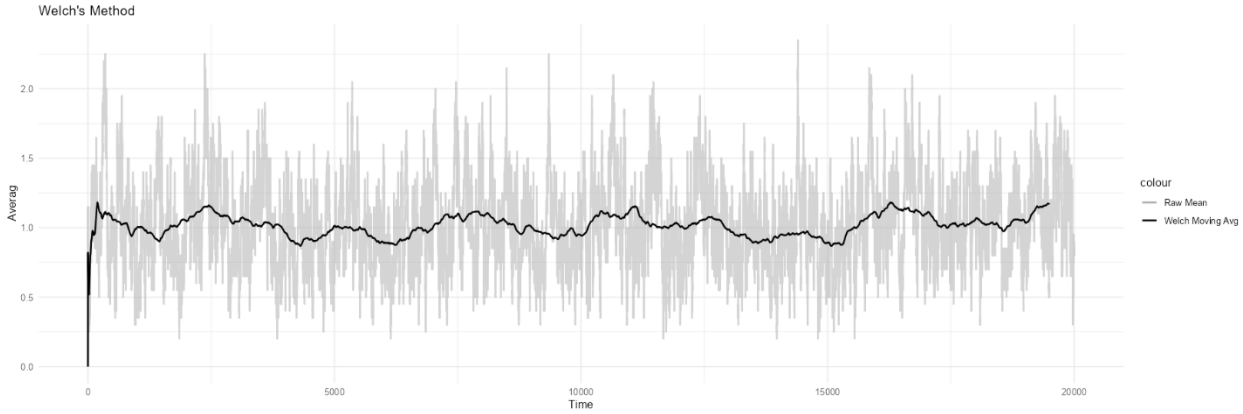


Figure 17: Test 1 (WIP) Welch's Method (w=500)

6.1.2. MSER-5

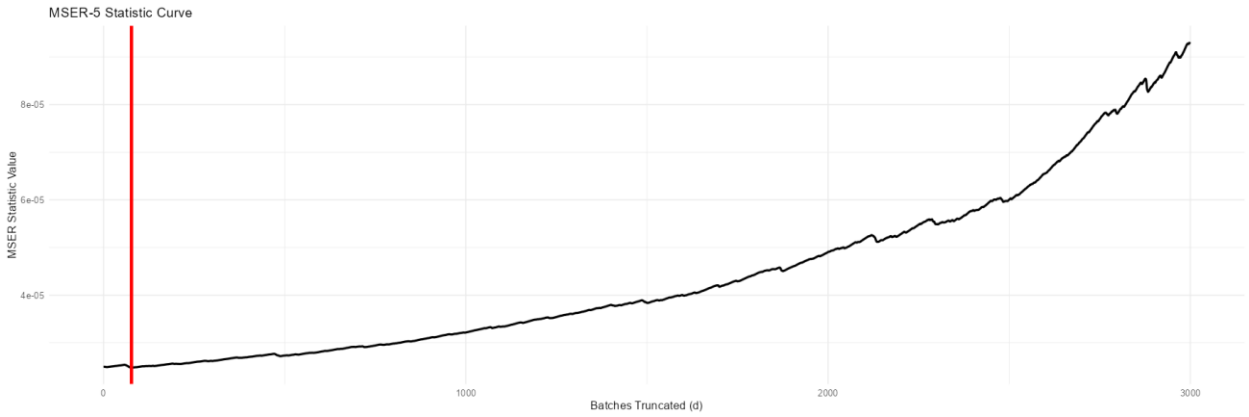


Figure 18: Test 1 (WIP) MSER-5

Recommended Truncation Time: 385 Hrs

6.1.3. SPC

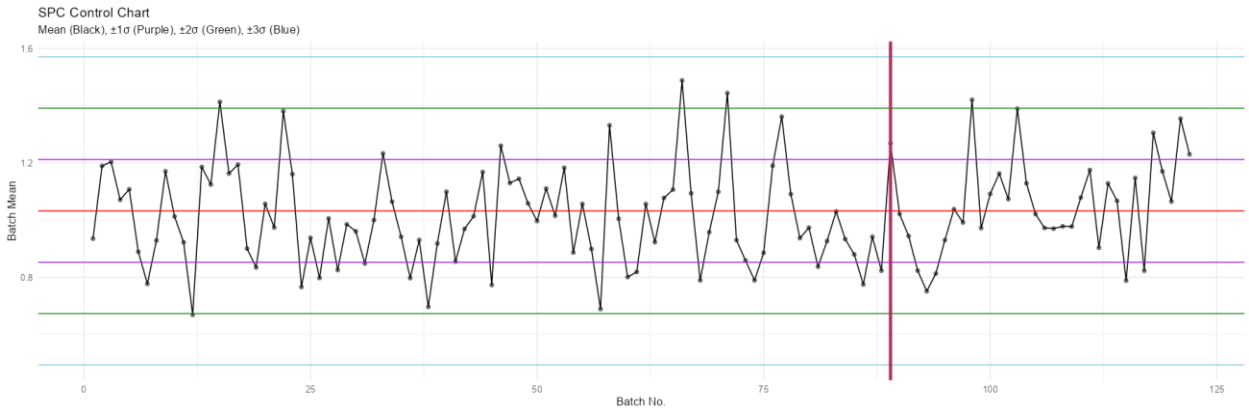


Figure 19: Test 1 (WIP) SPC

Recommended Truncation Time: 14344

6.1.4. Combined Methods

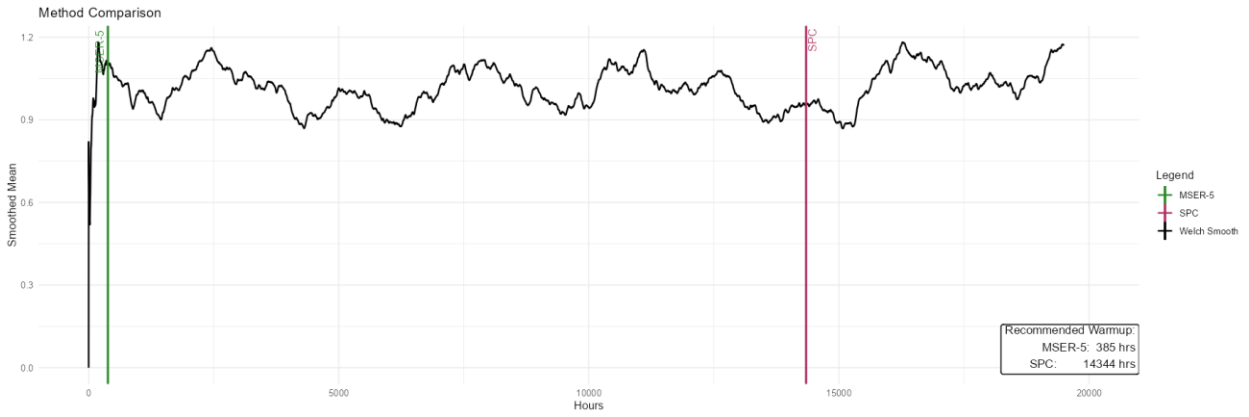


Figure 20: Test 1 (WIP) Combined Method Graph

6.1.5. Overall Analysis

From the Figure 16: Test 1 Conditions, we can see that the expected WIP value to be 1 item in the system. As it has a low utilisation, the system is quick to warm-up. Looking at Welch's method, the initial bias seems to level off similar to where the MSER-5 method recommends truncation (385 Hrs). The Statistical Process Control seems to have overshot the warm-up period by a large amount, leaving valuable unbiased data points behind.

WIP:	Average	Percentage Error	Bias	Variance	MSE	% Change in MSE
ANALYTICAL	1	0.000%	-	-	-	-
Raw Data	1.019709015	1.971%	0.01970901	0.004056	0.0044447	0.00%
MSER5 Data Removed	1.01653752	1.654%	0.01653752	0.000197	0.00047004	-89.42%
SPC Data Removed	1.056637794	5.664%	0.05663779	0.001255	0.00446297	0.41%

Figure 21: Test 1 - Analysis Table

The results in the Table above displays the impact of warmup removal on the accuracy of the estimated work-in-progress (WIP) relative to the analytical benchmark. The raw data produces an estimate that is already close to the analytical value, reinforcing the transient period in this low utilisation system is relatively short. The application of MSER-5 further improves the estimate, reducing both bias and mean squared error (MSE), which confirms that a small portion of the initial data was still influencing the result. In contrast, the SPC-based truncation performs poorly, introducing a large bias and increasing the MSE. This suggests that the truncation point selected by SPC is inappropriate for this system, due to the short and weak transient behaviour associated with low utilisation. Overall, these results indicate that while warmup removal has a limited effect in low utilisation systems, MSER-5 provides a great improvement, whereas SPC can degrade estimation accuracy if applied without consideration of system characteristics.

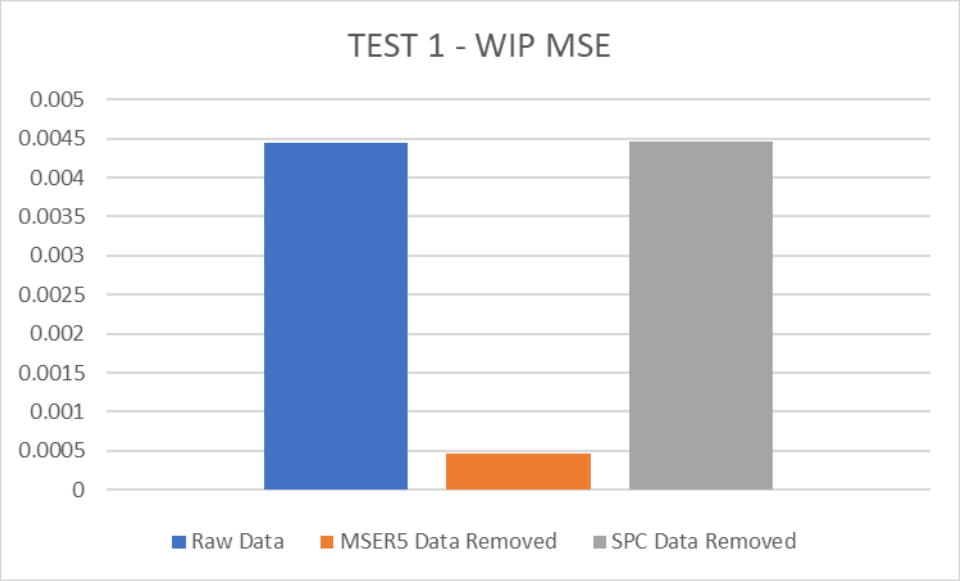


Figure 22: Test 1 (WIP) MSE Comparison

The results shown in Figure 22: Test 1 (WIP) MSE Comparison provide a visual comparison of the mean squared error (MSE) for the raw and post-truncation datasets. The raw data and SPC truncation exhibit similarly high MSE values, indicating that the SPC method does not improve estimation accuracy for this low utilisation system. In contrast, the MSER-5 method produces a substantially lower MSE, confirming a clear improvement in overall estimator quality. This visual trend supports the numerical results presented in Figure 21: Test 1 - Analysis Table and reinforces the conclusion that MSER-5 is more effective at identifying an appropriate warmup period in systems with short transient behaviour.

6.2. Test 2 – High Utilisation M/M/1 (Queue Length)

The same experimentation as above, was conducted again, this time on a system with a high utilisation. The conditions the simulation was run at is as follows:

- Mean Time Between Arrivals = 10
- Mean Service Time = 9
- Run Time = 50,000 hrs
- Recording Frequency = Every 1 Hr
- No. of Replications = 20

The Analytical Values are as follows:

MTBA	MST	Utilisation	Expected WIP	Expected Queue Length
10	9	0.9	9	8.1

Figure 23: Test 2 (Queue) Expected Values

As the

6.2.1. Welch's Method

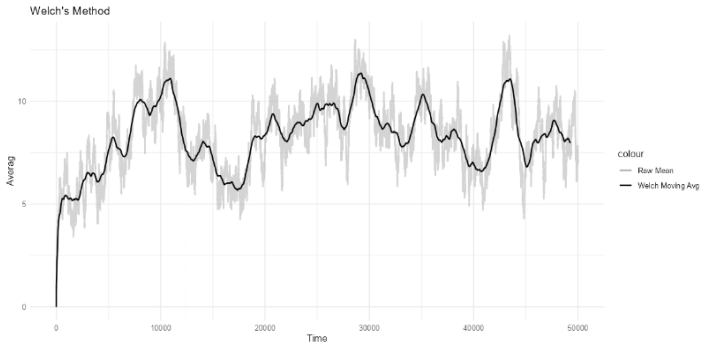


Figure 24: Test 2 Welch's Method (w=700)

6.2.2. MSER-5

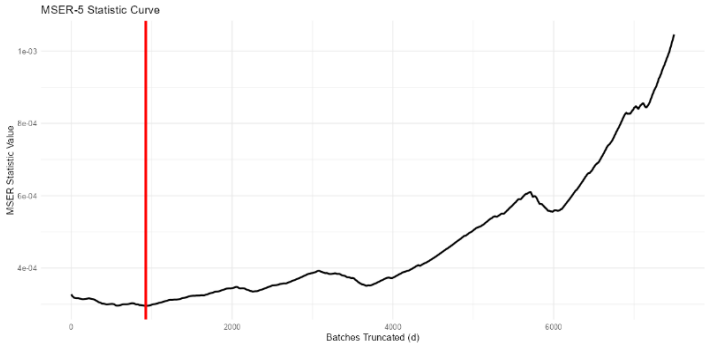


Figure 25: Test 2 MSER-5

6.2.3. SPC

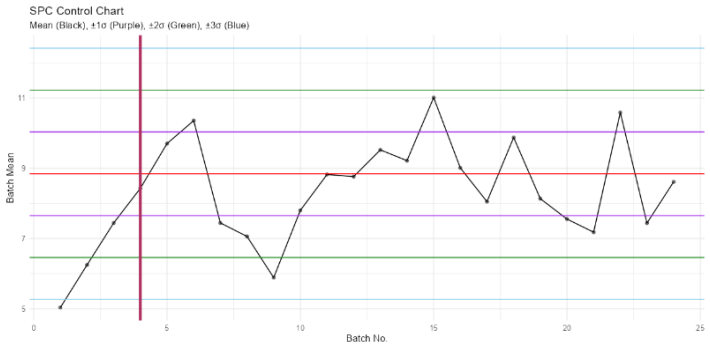


Figure 26: Test 2 SPC

6.2.4. Combined Methods

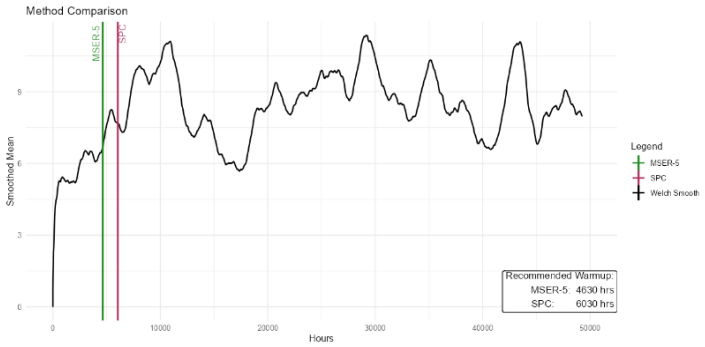


Figure 27: Test 2 Combined Methods

6.2.5. Overall Analysis

This system provided good results, i.e. both methods truncated the data at almost the same place. When the truncated data is analysed, the following table is produced:

WIP:	Average	Percentage Error	Bias	Variance	MSE	% Change in MSE
Raw Data	8.293068139	-7.855%	-0.7069319	0.178421	0.678174	0.00%
MSER5 Data Removed	8.562681008	-4.859%	-0.437319	0.216849	0.408097	-39.82%
SPC Data Removed	8.574168202	-4.731%	-0.4258318	0.2172	0.398533	-41.23%

Figure 28:Test 2 Results

This shows us that for this high utilisation system, SPC actually ends up being the best method to apply.

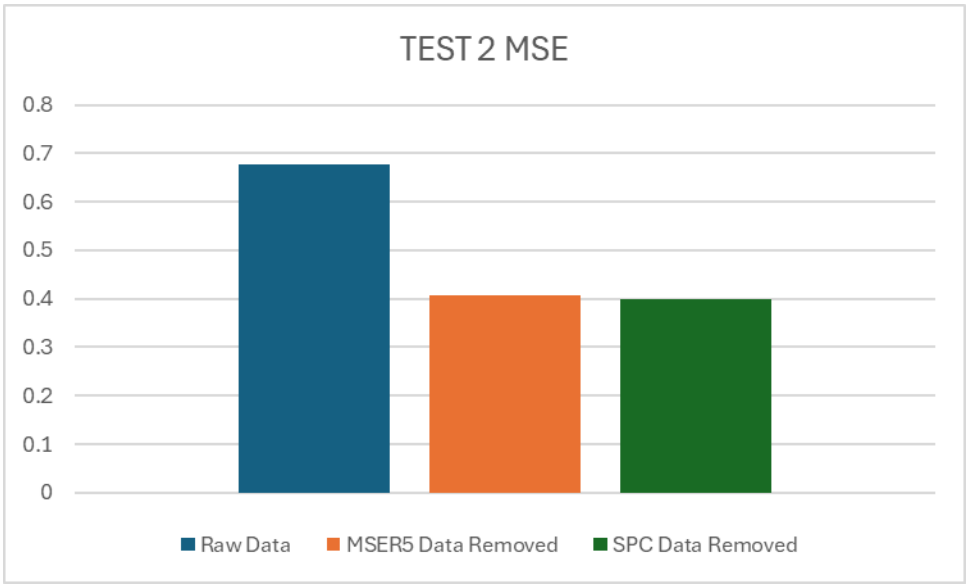


Figure 29: Test 2 MSE Results

It is evident here that, although barely, SPC is the better method to use for the initial bias removal here.

6.3. Test 3 – Multi Activity System M/M/1 (WIP)

In reality, a system where a machine undergoes just 1 activity is quite rare. To reflect real world conditions, a system was designed with five activities with a bottleneck station in the middle. The conditions that this five-machine system was run at are as follows:

- Mean Time Between Arrivals = 10
    - Machine 1 Mean Service Time = 6
    - Machine 2 Mean Service Time = 5
    - Machine 2 Mean Service Time = 9.5
    - Machine 2 Mean Service Time = 5
    - Machine 2 Mean Service Time = 6
- Run Time = 120,000 hrs
- Recording Frequency = Every 10 Hr
- No. of Replications = 30

The expected number of items in the system and the expected number of items in the queue can be seen in the following table.

	MTBA	MST	Utilisation	Expected WIP	Expected Queue Length
Activity 1	10	6	0.6	1.5	0.9
Activity 2		5	0.5	1	0.5
Activity 3		9.5	0.95	19	18.05
Activity 4		5	0.5	1	0.5
Activity 5		6	0.6	1.5	0.9
TOTAL		-	-	24	20.85

Figure 30: Test 3 % Machine System Expected Values

As the item has to go through more processes, similar to Test 2 – High Utilisation M/M/1 (Queue Length), the average WIP is longer and therefore it will take longer for the system to reach steady state. This means that the detection methods applied will have a better chance of being able to identify the period correctly without missing it, as is common in singe server M/M.1 queues and is probably what happened to the SPC method in Test 1 – Low Utilisation M/M/1 (WIP).

6.3.1. Welches Method

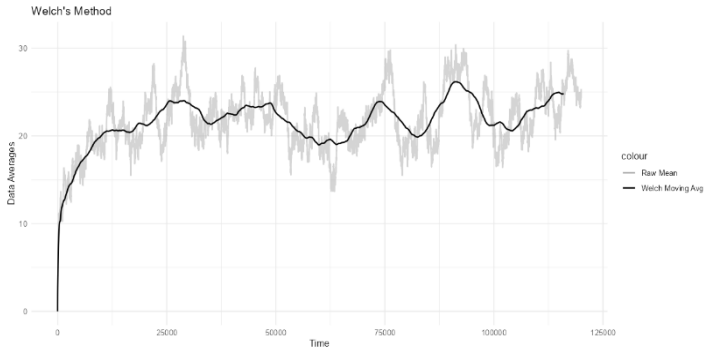


Figure 31: Test 3 Welch's Method (w=400)

6.3.2. MSER-5

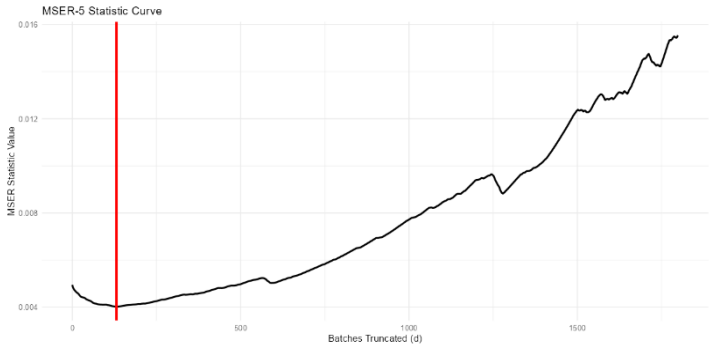


Figure 32: Test 3 MSER-5

6.3.3. SPC

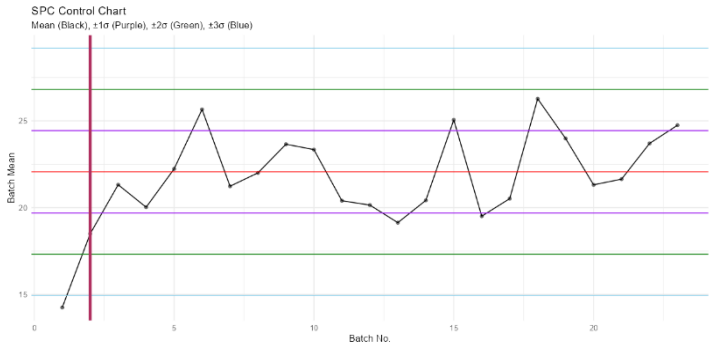
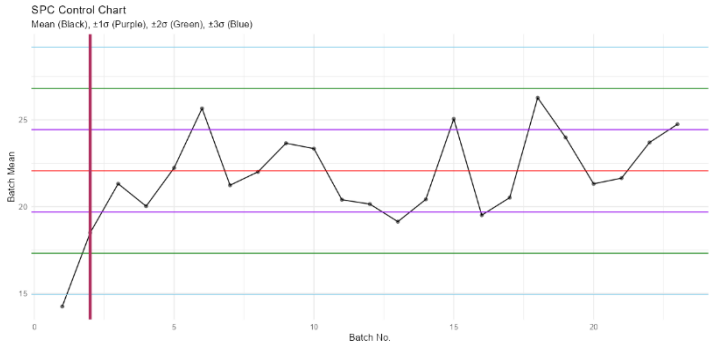


Figure 33: Test 3 SPC

6.3.4. Combined Methods



6.3.5. Overall Analysis

WIP:	Average	Percentage Error	Bias	Variance	MSE	% Change in MSE
Raw Data	21.75863261	-9.339%	-2.2413674	0.80515	5.828878	0.00%
MSER5 Data Removed	22.16609818	-7.641%	-1.8339018	0.9372	4.300396	-26.22%
SPC Data Removed	22.09389363	-7.942%	-1.9061064	0.918787	4.552028	-21.91%

It can be observed here that although this is a high utilisation system again, that MSER5 actually is the best method for the identification in this case.

7. CONCLUSIONS & RECOMMENDATIONS

This project set out to address the challenge of identifying and removing the warm-up period in steady-state discrete event simulation through the development of an automated analysis tool. By implementing and evaluating multiple warm-up detection methods within a R Shiny application, the study aimed to assess both the practical applicability and effectiveness of these approaches. The following conclusions summarise the key findings from the validation and analysis stages of the project.

- The developed application successfully integrates multiple warm-up detection methods and applies them to simulation output data in a consistent and automated manner.

- The implementation of Welch's method, MSER-5, and SPC allows users to compare different truncation recommendations within a single analytical framework.
- The results demonstrate that warm-up detection methods can effectively improve the accuracy of steady-state estimates when appropriately applied.
- MSER-5 was found to provide the most reliable and consistent truncation recommendations across the test cases considered, producing lower bias and reduced mean squared error in comparison to the untreated data.
- The SPC-based method showed inconsistent performance, particularly in low utilisation systems, where it frequently produced overly conservative or inappropriate truncation points.
- Across the validation experiments, the application performed best when MSER-5 was applied, with results consistently converging towards the analytical benchmark values.
- The comparison of methods highlights the importance of selecting an appropriate warm-up detection approach, as poor truncation can significantly degrade estimation accuracy.
- The use of an automated tool reduces subjectivity in warm-up selection and improves the repeatability of simulation output analysis.

Overall, the developed application may aid those analysing the effect of the initial bias on data. However it is important to note that the methods results on the system be analysed in a similar way to the calculations done in the Results & Discussion to ensure that the best method is chosen while balancing the bias and variation experienced by the system.

REFERENCES

[1] B. Nelson and L. Pei, *Foundations and methods of stochastic simulation*, vol. 187. Springer, 2013.
[2] A. M. Law, *Simulation modeling and analysis*, vol. 3.
[3] P. Sharma, "Discrete-event simulation," *Int J Sci Technol Res*, vol. 4, no. 04, pp. 136–140, 2015.
[4] S. Robinson, "Discrete-Event Simulation: From the Pioneers to the Present, What Next?," *J. Oper. Res. Soc.*, vol. 56, no. 6, pp. 619–629, 2005.
[5] J. Banks, "Introduction to simulation," in *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, Dec. 2000, pp. 9–16 vol.1. doi: 10.1109/WSC.2000.899690.
[6] W. D. Kelton, "Statistical analysis of simulation output," in *Proceedings of the 29th Conference on Winter Simulation*, in WSC '97. USA: IEEE Computer Society, 1997, pp. 23–30. doi: 10.1145/268437.268443.
[7] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer, 2007.
[8] K. Pawlikowski, "Steady-state simulation of queueing processes: survey of problems and solutions," *ACM Comput. Surv. CSUR*, vol. 22, no. 2, pp. 123–170, 1990.
[9] E. Parzen, *Stochastic processes*. SIAM, 1999.
[10] J. R. Wilson and A. A. B. Pritsker, "A survey of research on the simulation startup problem," *Simulation*, vol. 31, no. 2, pp. 55–58, 1978.
[11] W. D. Kelton and A. M. Law, "The Transient Behavior of the M/M/s Queue, with Implications for Steady-State Simulation," *Oper. Res.*, vol. 33, no. 2, pp. 378–396, 1985.
[12] W. K. Grassmann, "Factors affecting warm-up periods in discrete event simulation," *Simulation*, vol. 90, no. 1, pp. 11–23, 2014.
[13] W. Grassmann, "Rethinking the initialization bias problem in steady-state discrete event simulation," presented at the Proceedings of the 2011 Winter Simulation Conference (WSC), IEEE, 2011, pp. 593–599.
[14] R. Pasupathy and B. Schmeiser, "The initial transient in steady-state point estimation: Contexts, a bibliography, the MSE criterion, and the MSER statistic," presented at the Proceedings of the 2010 Winter Simulation Conference, IEEE, 2010, pp. 184–197.
[15] M. A. McClamon, "Detection of steady state in discrete event dynamic systems: An analysis of heuristics," 1990.
[16] A. M. Law, "Statistical Analysis of Simulation Output Data," *Oper. Res.*, vol. 31, no. 6, pp. 983–1029, 1983.
[17] K. Hoad, S. Robinson, and R. Davies, "Automating warm-up length estimation," *J. Oper. Res. Soc.*, vol. 61, no. 9, pp. 1389–1403, Sep. 2010, doi: 10.1057/jors.2009.87.
[18] D.-J. Jwo, W.-Y. Chang, and I.-H. Wu, "Windowing Techniques, the welch method for improvement of Power Spectrum Estimation.," *Comput. Mater. Contin.*, vol. 67, no. 3, 2021.
[19] K. P. White, M. J. Cobb, and S. C. Spratt, "A comparison of five steady-state truncation heuristics for simulation," in *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, Dec. 2000, pp. 755–760 vol.1. doi: 10.1109/WSC.2000.899843.
[20] K. P. White and S. N. Hwang, "Delay times in an M/M/1 queue: Estimating the sampling distributions for the steady-state mean and MSER truncation point," presented at the 2015 Winter Simulation Conference (WSC), IEEE, 2015, pp. 493–504.
[21] P. S. Mahajan and R. G. Ingalls, "Evaluation of methods used to detect warm-up period in steady state simulation," in *Proceedings of the 2004 Winter Simulation Conference, 2004.*, Dec. 2004, pp. 1–671. doi: 10.1109/WSC.2004.1371374.
[22] S. Robinson, "A statistical process control approach to selecting a warm-up period for a discrete-event simulation," *Eur. J. Oper. Res.*, vol. 176, no. 1, pp. 332–346, 2007.
[23] L. S. Nelson, "The Shewhart control chart—tests for special causes," *J. Qual. Technol.*, vol. 16, no. 4, pp. 237–239, 1984.
[24] K. Hoad, S. Robinson, and R. Davies, "AutoSimOA: a framework for automated analysis of simulation output," *J. Simul.*, vol. 5, no. 1, pp. 9–24, Feb. 2011, doi: 10.1057/jos.2010.22.

[25] S. M. Sanchez, P. J. Sanchez, and H. Wan, "Work smarter, not harder: A tutorial on designing and conducting simulation experiments," presented at the 2020 winter simulation conference (wsc), IEEE, 2020, pp. 1128–1142.
[26] K. Hoad and S. Robinson, "Implementing MSER-5 in commercial simulation software and its wider implications," presented at the Proceedings of the 2011 Winter Simulation Conference (WSC), IEEE, 2011, pp. 495–503.
[27] B. McCullough, "Is it safe to assume that software is accurate?," *Int. J. Forecast.*, vol. 16, no. 3, pp. 349–357, 2000.
[28] G. Sawitzki, "Testing numerical reliability of data analysis systems," 1994.
[29] K. B. Keeling and R. J. Pavur, "A comparative study of the reliability of nine statistical software packages," *Comput. Stat. Data Anal.*, vol. 51, no. 8, pp. 3811–3831, 2007.
[30] C. Hill, L. Du, M. Johnson, and B. D. McCullough, "Comparing programming languages for data analytics: Accuracy of estimation in Python and R," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 14, no. 3. 2024. doi: 10.1002/widm.1531.
[31] R. Ihaka and R. Gentleman, "R: A Language for Data Analysis and Graphics," *J. Comput. Graph. Stat.*, vol. 5, no. 3, pp. 299–314, 1996, doi: 10.2307/1390807.
[32] R. A. M. Villanueva and Z. J. Chen, "ggplot2: elegant graphics for data analysis," 2019.
[33] L. Jia *et al.*, "Development of interactive biological web applications with R/Shiny," *Brief. Bioinform.*, vol. 23, no. 1, p. bbab415, Oct. 2021, doi: 10.1093/bib/bbab415.
[34] M. Eickhoff, D. C. McNickle, and K. Pawlikowski, "Detecting the duration of initial transient in steady state simulation of arbitrary performance measures," in *ValueTools*, 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:7467632

Appendix 1 :

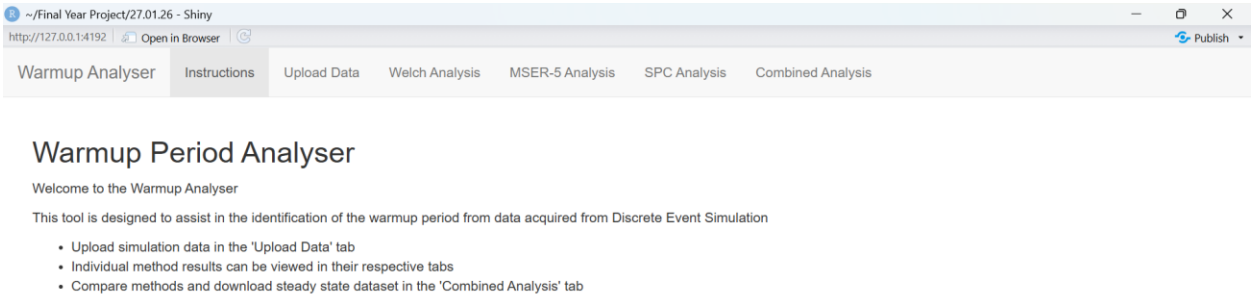


Figure 34: App Introduction Tab

Appendix 2 :

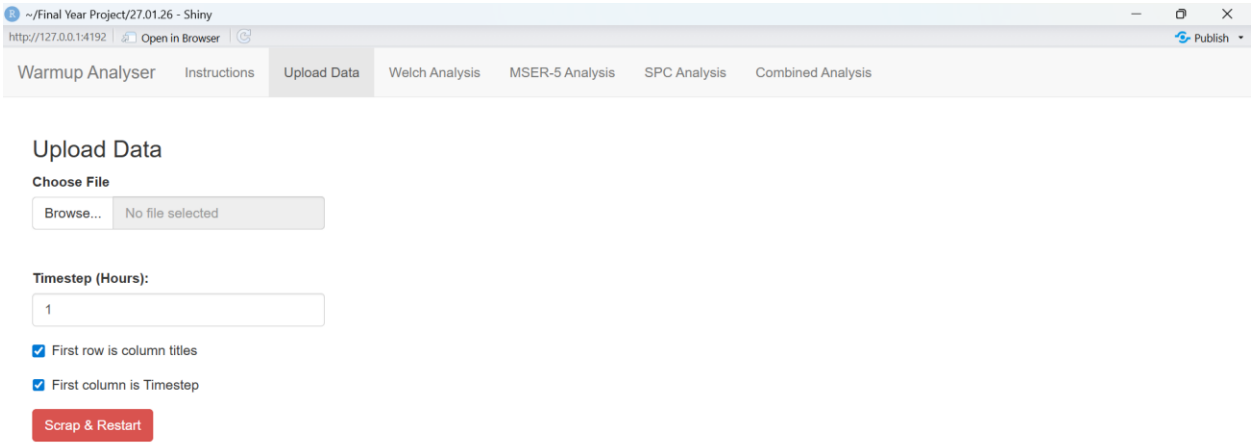


Figure 35: Upload Data Tab

Appendix 3

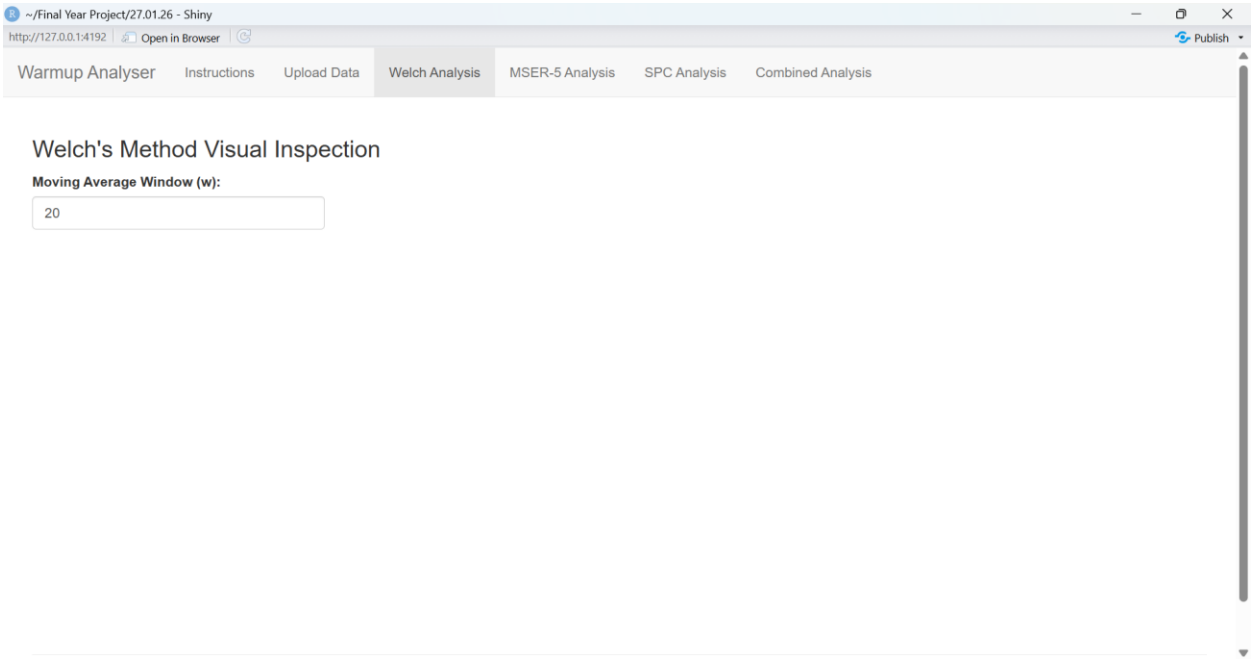


Figure 36: Welch's Method Tab (No Data)

Appendix 4 :

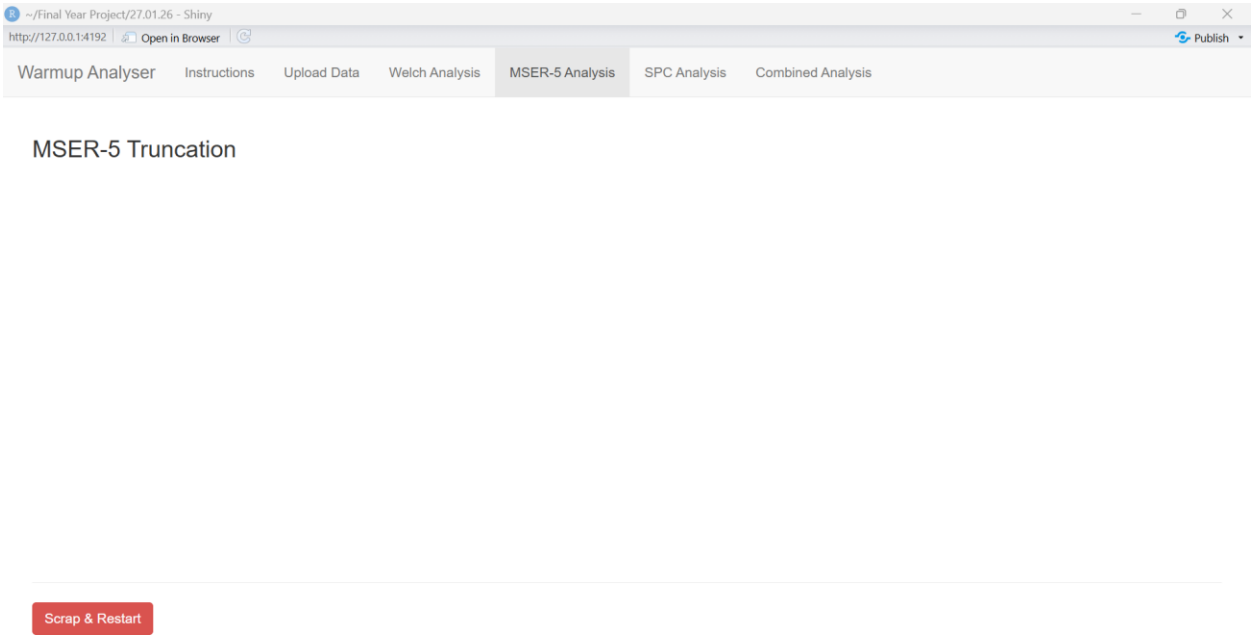


Figure 37: MSER5 Tab (No Data)

Appendix 5

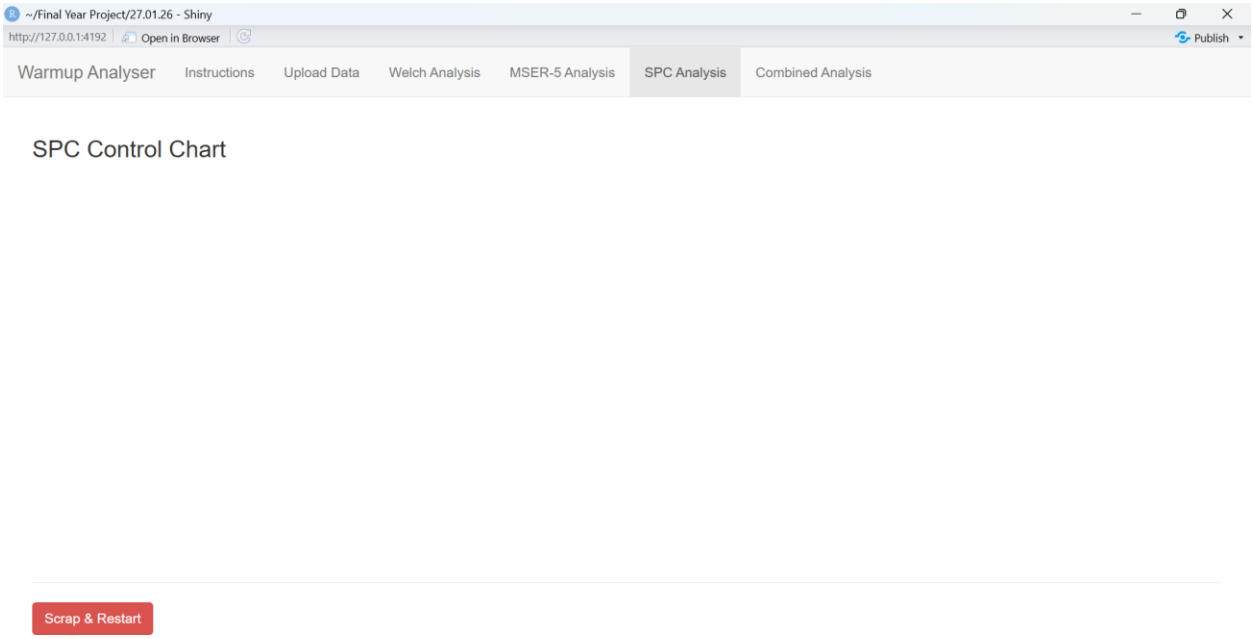


Figure 38: SPC Tab (No Data)

Appendix 6

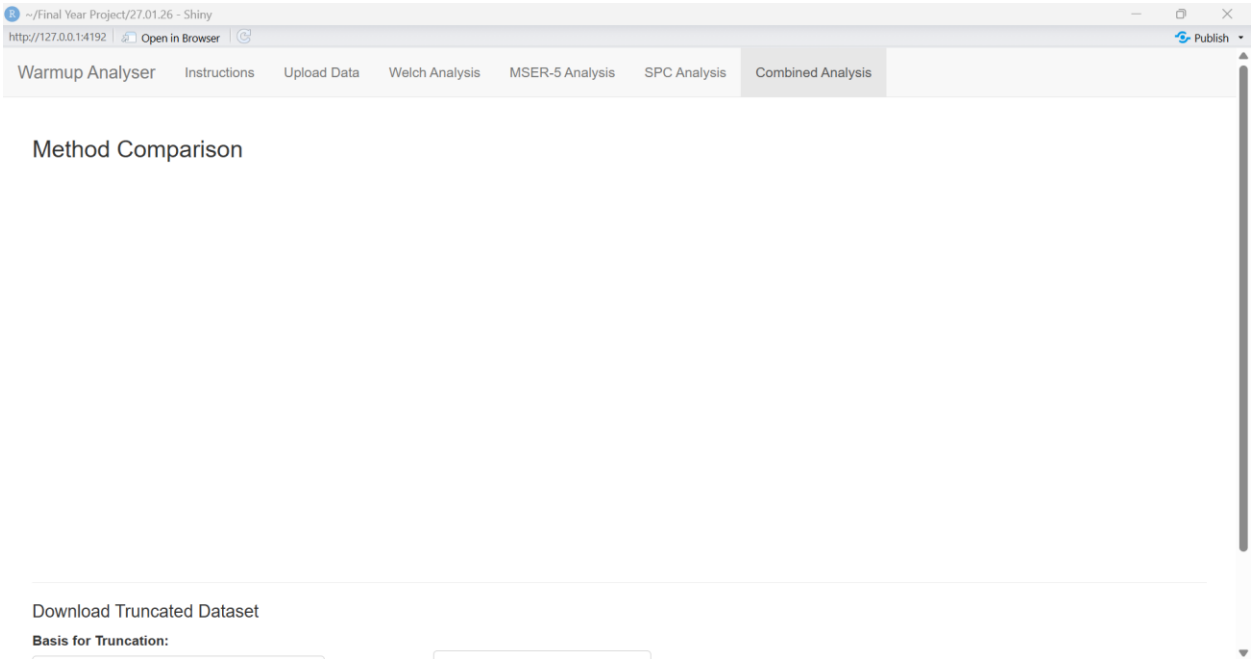


Figure 39: Combine Method Analysis Tab (No Data)

Appendix 7

```
 1. library(shiny)
 2. library(shinyjs)
 3. library(readr)
 4. library(readxl)
 5. library(ggplot2)
 6.
 7. options(shiny.maxRequestSize = 100 * 1024^2)
 8.
 9. #load methods
10. tryCatch({
11.   source("welch.R",local=TRUE)
12.   source("mser5.R",local=TRUE)
13.   source("SPC.R",local=TRUE)
14. })
15.
16. ui <- navbarPage(
17.   title="Warm-up Analyser",
18.   id   ="nav_menu",
19.   header=useShinyjs(),
20.
21.   tabPanel("Instructions",value="instructions_tab",
22.            fluidPage(
23.              h2("Warm-up Period Analyser"),
24.              p("Welcome to the Warm-up Analyser"),
25.              p("This tool is designed to assist in the identification of the warm-up period from
data acquired from Discrete Event Simulation"),
26.              tags$ul(
27.                tags$li("Upload simulation data in the 'Upload Data' tab"),
28.                tags$li("Individual method results can be viewed in their respective tabs"),
29.                    tags$li("Compare methods and download steady state dataset in the 'Combined
Analysis' tab")
30.              ),
31.
32.            )
33.   ),
34.
35.   tabPanel("Upload Data",value="upload_tab",
36.            fluidPage(
37.              h3("Upload Data"),
38.              div(id="upload_area",
39.                  fileInput("file_upload","Choose File",accept=c(".csv",".xlsx",".xls")),
```

```
                numericInput("timestep_value","Timestep (Hours):",value=1,min=0.01),
                checkboxInput("file_has_header","First row is column titles",TRUE),
                checkboxInput("file_has_timestep_column","First column is Timestep",TRUE)
              ),
              actionButton("scrap","Scrap & Restart",class="btn-danger")
            )
  ),

  tabPanel("Welch Analysis",value="welch_tab",
            fluidPage(
              h3("Welch's Method Visual Inspection"),
              numericInput("welch_window","Moving Average Window (w):",value=20,min=0),
              plotOutput("welch_plot"),
              hr(),
              actionButton("scrap","Scrap & Restart",class="btn-danger")
            )
  ),

  tabPanel("MSER-5 Analysis",value="mser_tab",
            fluidPage(
              h3("MSER-5 Truncation"),
              h4(textOutput("mser_result_text"),style="color:red; font-weight:bold;"),
              plotOutput("mser_plot"),
              hr(),
              actionButton("scrap","Scrap & Restart",class="btn-danger")
            )
  ),

  tabPanel("SPC Analysis",value="spc_tab",
            fluidPage(
              h3("SPC Control Chart"),
              h5(textOutput("spc_batch_text"),style="color:blue; font-weight:bold;"),
              h4(textOutput("spc_cutoff_text"),style="color:red; font-weight:bold;"),
              plotOutput("spc_chart"),
              hr(),
              actionButton("scrap","Scrap & Restart",class="btn-danger")
            )
  ),

  tabPanel("Combined Analysis",value="tab_summary",
            fluidPage(
              h3("Method Comparison"),
              plotOutput("summary_plot"),
```

```
 83.              hr(),
 84.              h4("Download Truncated Dataset"),
 85.              fluidRow(
 86.                    column(4,selectInput("trunc_method","Basis for Truncation:",choices=c("MSER-
5","SPC"))),
 87.                column(4,br(),downloadButton("download_truncated","Download Steady-State CSV"))
 88.              ),
 89.              hr(),
 90.              actionButton("scrap","Scrap & Restart",class="btn-danger")
 91.            )
 92.   )
 93. )
 94.
 95. server <- function(input,output,session) {
 96.
 97.   data_store <- reactiveValues(raw=NULL,clean=NULL,welch=NULL,mser=NULL,spc=NULL)
 98.
 99.   #Scrap & Restart Logic
100.   observeEvent(c(input$scrap),{
101.     #Clear all  data
102.     data_store$raw <- NULL
103.     data_store$clean <- NULL
104.     data_store$welch <- NULL
105.     data_store$mser <- NULL
106.     data_store$spc <- NULL
107.
108.     #Reset the file input UI
109.     shinyjs::reset("upload_area")
110.
111.     #Bring user back to the data upload tab
112.     updateTabsetPanel(session,"nav_menu",selected="upload_tab")
113.   },ignoreInit=TRUE)
114.
115.   #Data loading
116.   observeEvent(list(input$file_upload,input$welch_window),{
117.     req(input$file_upload)
118.     ext <- tolower(tools::file_ext(input$file_upload$name))
119.     df <- if(ext == "csv") read_csv(input$file_upload$datapath,col_names=input$file_has_header)
else read_excel(input$file_upload$datapath,col_names=input$file_has_header)
120.
121.     data_store$raw <- df
122.
123.     #check if the first column is a timestep
```

```
124.     clean_df <- if(input$file_has_timestep_column && ncol(df) > 1) df[,-1,drop=FALSE] else df
125.
126.     clean_df[] <- lapply(clean_df,function(x) as.numeric(as.character(x)))
127.     data_store$clean <- clean_df
128.
129.     #Dynamically count runs for SPC standard deviation adjustment
130.     number_of_runs <- ncol(clean_df)
131.     ensemble <- rowMeans(clean_df,na.rm=TRUE)
132.
133.     #Run calculations
134.     data_store$welch <- welch_apply(clean_df,window=input$welch_window)
135.     data_store$mser <- mser_apply(ensemble)
136.     data_store$spc <- spc_apply(ensemble,num_runs=number_of_runs)
137.   })
138.
139.   #Welch Plot
140.   output$welch_plot <- renderPlot({
141.     req(data_store$welch)
142.     df <- data_store$welch
143.     df$Time <- (0:(nrow(df)-1))*input$timestep_value
144.
145.     ggplot(df,aes(x=Time))+
146.       geom_line(aes(y=Welch_Mean,color="Raw Mean"),alpha=0.5,size=00.75,na.rm=TRUE)+
147.       geom_line(aes(y=Welch_MovingAvg,color="Welch Moving Avg"),size=0.75,na.rm=TRUE)+
148.       scale_color_manual(values=c("Raw Mean"="darkgray","Welch Moving Avg"="black"))+
149.       theme_minimal()+labs(title="Welch's Method",y="Averag")
150.   })
151.
152.   #MSER-5 Plot
153.   output$mser_result_text <- renderText({
154.     req(data_store$mser)
155.     paste("Recommended Cutoff:",data_store$mser$cutoff*input$timestep_value,"Hours")
156.   })
157.
158.   output$mser_plot <- renderPlot({
159.     req(data_store$mser)
160.     stats <- data_store$mser$stats_vector
161.      if(length(stats) == 0) return(ggplot()+theme_void()+ggtitle("Not enough data to calculate
MSER-5"))
162.
163.     df_mser <- data.frame(Batch_d=0:(length(stats)-1),Stat=stats)
164.     cutoff_d <- data_store$mser$cutoff/5
165.
```

```
    ggplot(df_mser,aes(x=Batch_d,y=stats))+
      geom_line(color="black",size=1)+
      geom_vline(xintercept=cutoff_d,color="red",size=1.5)+
        theme_minimal()+labs(title="MSER-5 Statistic Curve",x="Batches Truncated (d)",y="MSER Statistic Value")
  })

  #SPC Plot
  output$spc_batch_text <- renderText({
    req(data_store$spc)
                      if(data_store$spc$passed)    paste("Minimum    Batch    Size    Passing Tests:",data_store$spc$batch_size) else "Validation failed."
  })

  output$spc_cutoff_text <- renderText({
    req(data_store$spc)
    if(!data_store$spc$passed) return("No cutoff available.")
    ct <- data_store$spc$cutoff_time
    if(is.na(ct)) return("Recommended Cutoff: Process did not reach steady state within run.")
    paste("Recommended Cutoff:",ct * input$timestep_value,"Hours")
  })

  output$spc_chart <- renderPlot({
    req(data_store$spc)
    res <- data_store$spc
    if(!res$passed) return(ggplot()+theme_void()+ggtitle("Data failed validation"))

    df_spc <- data.frame(Idx=1:length(res$batch_means),Val=res$batch_means)

    p <- ggplot(df_spc,aes(x=Idx,y=Val))+
      geom_line(color="black")+geom_point(alpha=0.5)+
      geom_hline(yintercept=res$mean,color="red")+
      geom_hline(yintercept=c(res$u1,res$l1),color="purple")+
      geom_hline(yintercept=c(res$u2,res$l2),color="forestgreen")+
      geom_hline(yintercept=c(res$u3,res$l3),color="skyblue")+
        labs(title="SPC Control Chart",subtitle="Mean (Black), ±1σ (Purple), ±2σ (Green), ±3σ (Blue)",x="Batch No.",y="Batch Mean")+
      theme_minimal()

    if(!is.na(res$cutoff_batch)) {
      p <- p+geom_vline(xintercept=res$cutoff_batch,color="maroon",size=1.5)
    } else {
      p <- p+ggtitle("Strict SPC: Process NEVER fully stabilizes within limits")
```

```
206.     }
207.     p
208.   })
209.
210.   #Combined Plot
211.   output$summary_plot <- renderPlot({
212.     req(data_store$welch)
213.     df_w <- data_store$welch
214.     df_w$Time <- (0:(nrow(df_w)-1))*input$timestep_value
215.
216.     p <- ggplot(df_w,aes(x=Time,y=Welch_MovingAvg,color="Welch Smooth"))+
217.       geom_line(size=0.75,na.rm=TRUE)+
218.       theme_minimal()+
219.       labs(title="Method Comparison",x="Hours",y="Smoothed Mean")
220.
221.     #build the annotation label as we go
222.     label_lines <- c()
223.
224.     if(!is.null(data_store$mser)) {
225.       m_cut <- data_store$mser$cutoff * input$timestep_value
226.       p <- p+geom_vline(aes(xintercept=m_cut,color="MSER-5"),size=1)+
227.              annotate("text",x=m_cut,y=Inf,label="MSER-5",color="forestgreen",angle=90,vjust=-
0.5,hjust=1.1)
228.       label_lines <- c(label_lines, paste0("MSER-5:  ", m_cut, " hrs"))
229.     }
230.
231.     if(!is.null(data_store$spc) && data_store$spc$passed && !is.na(data_store$spc$cutoff_time))
{
232.       s_cut <- data_store$spc$cutoff_time * input$timestep_value
233.       p <- p+geom_vline(aes(xintercept=s_cut,color="SPC"),size=1)+
234.         annotate("text",x=s_cut,y=Inf,label="SPC",color="maroon",angle=90,vjust=1.2,hjust=1.1)
235.       label_lines <- c(label_lines, paste0("SPC:        ", s_cut, " hrs"))
236.     }
237.
238.             p   <-   p+scale_color_manual(name="Legend",values=c("Welch   Smooth"="black","MSER-
5"="forestgreen","SPC"="maroon"))
239.
240.     #method legend
241.       box_label <- paste(c("Recommended Warm-up:", label_lines), collapse="\n")
242.       p <- p+annotate("label",
243.                       x=Inf, y=Inf,
244.                       label=box_label,
245.                       hjust=1, vjust=7,
```

```
                      size=4,
                      fill="white",
                      colour="black",
                      label.size=0.5)


    p
  })
  #Data Export
  output$download_truncated <- downloadHandler(
    filename=function() { paste0("Cleaned ",input$trunc_method,".csv") },
    content=function(file) {
      req(data_store$raw)

      drop_until <- if(input$trunc_method == "MSER-5") {
        data_store$mser$cutoff
      } else {
        if(is.na(data_store$spc$cutoff_time)) 0 else data_store$spc$cutoff_time
      }

      if(is.null(drop_until) || drop_until >= nrow(data_store$raw) || drop_until <= 0) {
        write_csv(data_store$raw,file)
      } else {
        write_csv(data_store$raw[(drop_until+1):nrow(data_store$raw),],file)
      }
    }
  )
}

shinyApp(ui,server)

```

Figure 40: R Shiny Code (App)

Appendix 8

```r
welch_apply <- function(df, window = 20) {
  if (is.null(df) || nrow(df) == 0) return(NULL)

  #calculating raw means
  y_bar <- rowMeans(df, na.rm = TRUE)
  #determining no. of time steps
  m <- length(y_bar)
  #pulling window size from app.R
  w <- as.integer(window)
  #storage for welch's method
  ma <- rep(NA, m)

  for (i in 1:m) {
    #stopping smoothing as not enough data
    if (i > (m - w)) {
      ma[i] <- NA
    }
    #apply growing moving average
    else if (i <= w) {
      ma[i] <- mean(y_bar[1:(2 * i - 1)], na.rm = TRUE)
    }
    #standard welch's moving window
    else {
      ma[i] <- mean(y_bar[(i - w):(i + w)], na.rm = TRUE)
    }
  }

  #return original data with the raw and smoothed means
  #mean at each time step
  df$Welch_Mean <- y_bar

  #smoothed means
  df$Welch_MovingAvg <- ma
  return(df)
}
```

Figure 41: Welch.R Code

Appendix 9

```
# Applies MSER to batch means with batch size b = 5 & returns cutoff point
mser_apply <- function(data_vector, b = 5, max_frac = 0.75, min_batches_remaining = 10) {

  #input is a numeric vector and remove any missing values
  y <- as.numeric(data_vector)
  y <- y[!is.na(y)]
  n_total <- length(y)

  # Check if the series is long enough to support the required minimum number of batches
  if (n_total < (b * (min_batches_remaining + 2))) {
    return(list(cutoff = 0L, stats_vector = numeric(0)))
  }

  # Pre-batch the raw output into non-overlapping groups to reduce autocorrelation
  num_batches <- floor(n_total / b)
  y_use <- y[1:(num_batches * b)]

  # Reshape data and calculate the mean for each batch (MSER-5 variant)
  batch_matrix <- matrix(y_use, nrow = b, byrow = FALSE)
  batch_means <- colMeans(batch_matrix)

  m <- length(batch_means)

  #check enough batches exist after potential truncation
  if (m < (min_batches_remaining + 2)) return(list(cutoff = 0L, stats_vector = numeric(0)))

  #set the maximum allowable truncation point to prevent discarding too much data
  max_d_by_frac <- floor(max_frac * m) - 1
  max_d_by_min  <- m - min_batches_remaining - 1
  max_d <- min(max_d_by_frac, max_d_by_min)

  if (max_d < 0) return(list(cutoff = 0L, stats_vector = numeric(0)))

  mser_stats <- numeric(max_d + 1)

    #iterate through candidate truncation points to find where the remaining sample is most
homogeneous
  for (d in 0:max_d) {
    remaining <- batch_means[(d + 1):m]
    L <- length(remaining)
    mu <- mean(remaining)
```

```
    #calculate sum of squared deviations divided by the square of remaining observations
    sse <- sum((remaining - mu)^2)
    mser_stats[d + 1] <- sse / (L^2)
  }

  #identify the truncation point d that minimizes the MSER statistic (approximating MSE)
  best_d <- which.min(mser_stats) - 1
  #convert the batch-based truncation point back into the original row/time index
  cutoff_rows <- best_d * b

  return(list(
    cutoff = as.integer(cutoff_rows),
    stats_vector = mser_stats
  ))
}
```

Figure 42:MSER5.R Code

Appendix 10

```
library(nortest) #needed for anderson darling test

spc_apply <- function(data_vector, num_runs = 1) {
  y <- as.numeric(na.omit(data_vector))
  n_total <- length(y)
  batch_size <- 1
  passing <- FALSE

  res <- list(passed = FALSE, batch_size = 1, cutoff_time = 0)

  while (!passing && batch_size < (n_total / 20)) {
    num_batches <- floor(n_total / batch_size)
    y_use <- y[1:(num_batches * batch_size)]
    batch_matrix <- matrix(y_use, nrow = batch_size, byrow = FALSE)
    b_means <- colMeans(batch_matrix)
    n <- length(b_means)



    #Von Neumann and Anderson Darling tests
    rvn <- (sum(diff(b_means)^2)/(n-1)) / (sum((b_means-mean(b_means))^2)/n)
    vn_p <- 2*(1-pnorm(abs((rvn-2)/sqrt(4/n))))
    ad_p <- nortest::ad.test(b_means)$p.value

    if (vn_p > 0.05 && ad_p > 0.05) {
      passing <- TRUE

      # Limits from second half of batches
      second_half_data <- floor(n/2):n
      steady_means <- b_means[second_half_data]
      mu_est <- mean(steady_means)
      sigma_est <- sd(steady_means)

      #Control Limit Calculations
      u1 <- mu_est + sigma_est
      u2 <- mu_est + 2*sigma_est
      u3 <- mu_est + 3*sigma_est
      l1 <- mu_est - sigma_est
      l2 <- mu_est - 2*sigma_est
      l3 <- mu_est - 3*sigma_est

```

```
42.
43.       #evaluate SPC Rules
44.       is_out <- rep(FALSE, n) #create a tracker for all points
45.
46.       #1. A single point plots outside a 3-sigma control limit
47.       is_out[b_means > u3 | b_means < l3] <- TRUE
48.
49.       #2. two out of three consecutive points plot outside a 2-sigma limit
50.       out_2s_upper <- (b_means > u2)
51.       out_2s_lower <- (b_means < l2)
52.       if (n >= 3) {
53.         for (i in 3:n) {
54.           #check upper side and lower side separately
55.           if (sum(out_2s_upper[(i-2):i]) >= 2 || sum(out_2s_lower[(i-2):i]) >= 2) {
56.             is_out[(i-2):i] <- TRUE
57.           }
58.         }
59.       }
60.
61.       #3. Four out of five consecutive points plot outside a 1-sigma limit (SAME SIDE)
62.       out_1s_upper <- (b_means > u1)
63.       out_1s_lower <- (b_means < l1)
64.       if (n >= 5) {
65.         for (i in 5:n) {
66.           # Check upper side and lower side separately
67.           if (sum(out_1s_upper[(i-4):i]) >= 4 || sum(out_1s_lower[(i-4):i]) >= 4) {
68.             is_out[(i-4):i] <- TRUE
69.           }
70.         }
71.       }
72.
73.       #4. nine consecutive points plot on one side of the mean
74.       above_mu <- (b_means > mu_est)
75.       below_mu <- (b_means < mu_est)
76.       if (n >= 9) {
77.         for (i in 9:n) {
78.           if (all(above_mu[(i-8):i]) || all(below_mu[(i-8):i])) {
79.             is_out[(i-8):i] <- TRUE  #entire 9-point window out of control
80.           }
81.         }
82.       }
83.
84.
```

```
      #determining final warm-up period
      out_of_limits <- which(is_out)

      if (length(out_of_limits) == 0) {
        cutoff_batch <- 1
      } else {
        last_out_of_control <- max(out_of_limits)

        if (last_out_of_control == n) {
          cutoff_batch <- NA
        } else {
          cutoff_batch <- last_out_of_control + 1
        }
      }

      cutoff_t <- if(!is.na(cutoff_batch)) (cutoff_batch - 1) * batch_size else NA
      res <- list(
        passed = TRUE, batch_size = batch_size, batch_means = b_means,
        cutoff_batch = cutoff_batch, cutoff_time = cutoff_t,
        mean = mu_est,
        u1 = u1, u2 = u2, u3 = u3,
        l1 = l1, l2 = l2, l3 = l3
      )
    } else {
      #incremental batch size increase
      batch_size <- batch_size + 1
    }
  }
  return(res)
}
```

Figure 43:SPC.R Code

Appendix 11

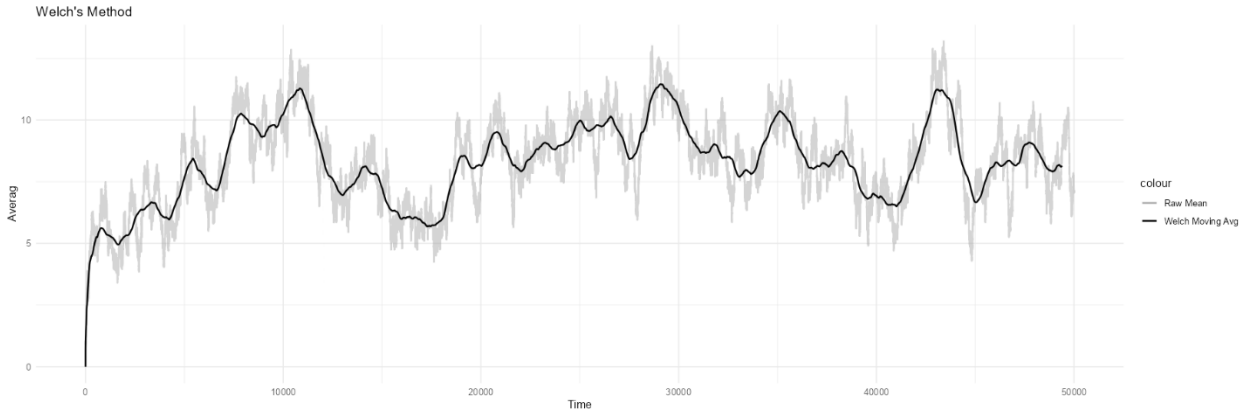


Figure 44: Example Welch's Method Application (*w*=600)

Appendix 12

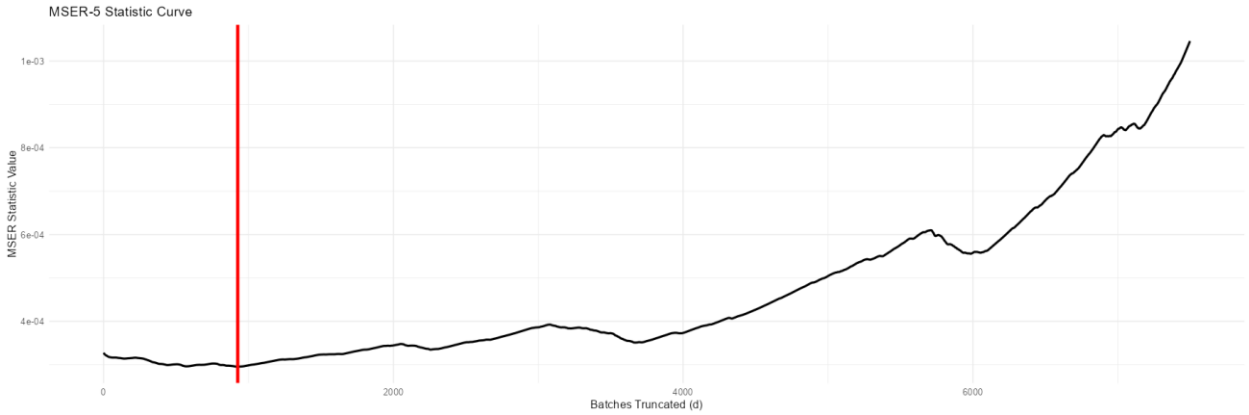


Figure 45: Example MSER-5 Application

Appendix 13

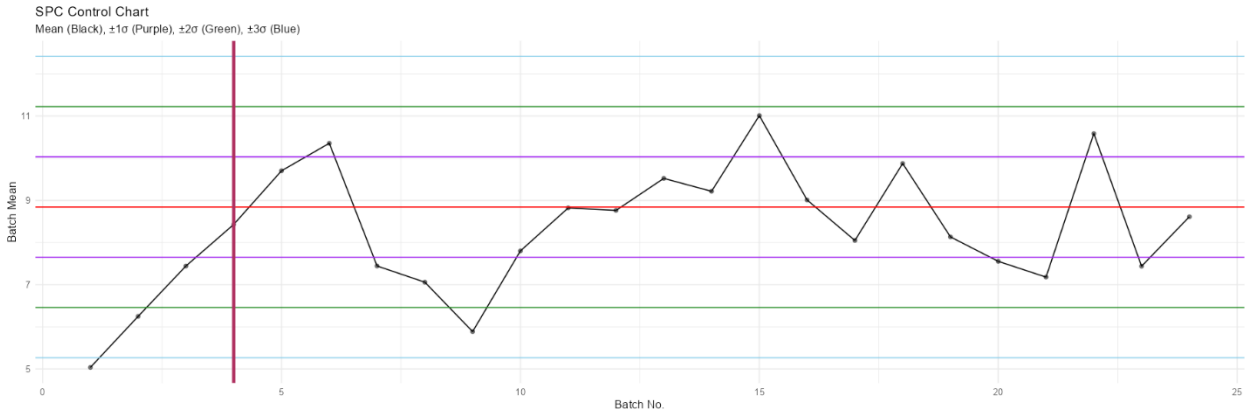


Figure 46: Example SPC Application

Appendix 14

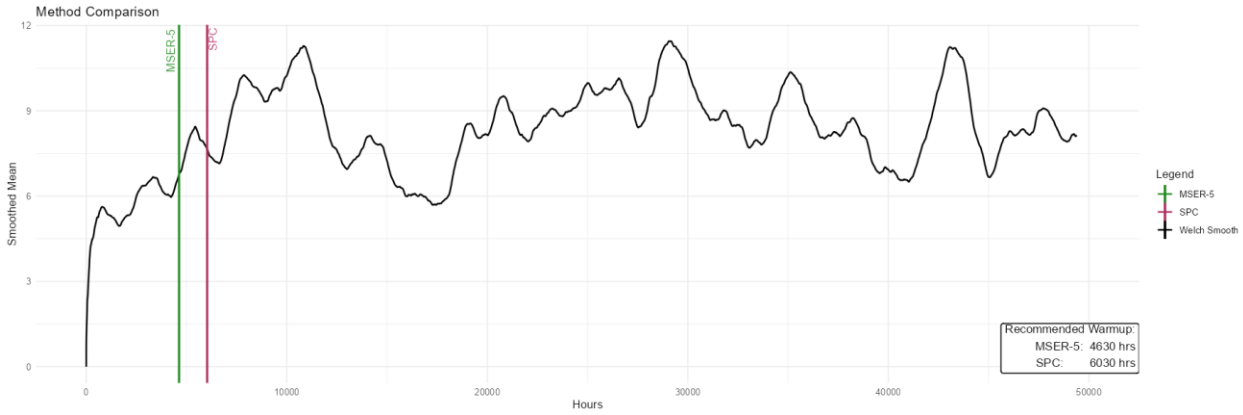


Figure 47: Example Combined Method Analysis Application