Journal of Statistical Software

July 2025, Volume 113, Issue 7. *doi: 10.18637/jss.v113.i07*

StatisticalProcessMonitoring.jl: A General Framework for Statistical Process Monitoring in Julia

Daniele Zago
Università degli Studi di Padova

Abstract

Statistical process monitoring (SPM) control charts are widely used for monitoring the stability of sequential processes. Currently, there is no open-source software which provides a general and extensible implementation of control charts. **StatisticalProcessMonitoring.jl** is a novel Julia package which aims at addressing this gap, offering support for monitoring various type of data, such as univariate and multivariate observations, partially-observed data streams, and profiles. The package introduces an extensible SPM framework, allowing users to seamlessly design control charts for structured data types using the existing implementation. By introducing a flexible implementation of control charts, **StatisticalProcessMonitoring.jl** provides fully-automated and efficient algorithms for determining control limits and tuning control chart hyperparameters. These algorithms can accommodate various commonly-used performance metrics based on the run length distribution. The package further leverages existing packages in the Julia ecosystem to offer users a range of optimization and plotting functionalities.

Keywords: Statistical process monitoring, control charts, Julia, Monte Carlo simulations.

1. Introduction

In statistical process monitoring (SPM), control charts are the main tool for testing the stability of sequential processes, with the aim of detecting departures from stable conditions. These methods find broad applicability across various domains such as, among others, industrial processes (Jin and Shi 1999; Huang, Zhou, and Shi 2002; Zhou, Huang, and Shi 2003), healthcare and disease surveillance (Steiner, Cook, Farewell, and Treasure 2000; Woodall 2006; Qiu and You 2022), monitoring of functions (termed "profiles" in the literature, see Kang and Albin 2000; Kim, Mahmoud, and Woodall 2003; Qiu, Zi, and Zou 2018), surveillance of images and videos (Yan, Paynabar, and Shi 2014; Grasso, Laguzza, Semeraro, and Colosimo 2016; Colosimo and Grasso 2018; Yan, Grasso, Paynabar, and Colosimo 2022), and detection of defects in manufactured 3D shapes (Colosimo, Semeraro, and Pacella 2008; Zang and Qiu 2018a,b; Zhao and del Castillo 2021; Scimone, Taormina, Colosimo, Grasso, Menafoglio, and Secchi 2022).

While novel control charts are constantly being developed, most SPM software packages focus on implementing traditional control charts for analyzing univariate or multivariate data. Therefore, they provide limited support for developing more sophisticated charting methodologies, such as multi-chart schemes (see, e.g., Gan 1995). Moreover, currently-available packages typically lack general methodologies for user-defined control charts, such as selecting the control limits and optimizing the chart hyperparameters. As a result, practitioners who wish to implement their own control charts often also need to implement their own routines to address these problems.

To the best of our knowledge, no other package implements a general SPM framework with a fully-automatic way of calculating control limits and optimizing control chart hyperparameters. Here, we present a comprehensive review of existing SPM packages across various programming languages and of the features they provide.

Most of the available packages for SPM are implemented in R (R Core Team 2025), a high-level language that is designed for statistical applications and is especially popular due to its ease of use and dynamic type system (Tratt 2009). The **qcc** package (Scrucca, Snow, and Bloomfield 2004) allows users to run Shewhart charts (Shewhart 1931), cumulative sum (CUSUM, Page 1954), and exponentially weighted moving average (EWMA, Roberts 1959) charts, with limited support for multivariate Shewhart charts. The **MSQC** package (Santos-Fernández 2012) specializes in conventional multivariate control charts, including multivariate Shewhart, multivariate cumulative sum (MCUSUM, Crosier 1988), and multivariate exponentially weighted moving average (MEWMA, Lowry, Woodall, Champ, and Rigdon 1992) charts. The **spc** package (Knoth 2024) focuses on calculating zero-state and steady-state in-control average run length and run length quantiles for the EWMA, CUSUM, and Shiryaev-Roberts control charts (Shiryaev 1961, 1963; Roberts 1966). These control charts are primarily used for monitoring the mean or variance of a normally-distributed univariate process. The package also provides average run length calculations for the MEWMA control chart when monitoring the process mean, as well as methods for determining optimal allowance constants and smoothing constants based on target mean shift values to detect. The **qicharts** and **qicharts2** packages (Anhoej 2021a,b) provide functions for run rules (Khoo 2003), Shewhart charts, and Pareto control charts (Juran 1951). The **edcc** package (Zhu and Park 2013) provides numerical methods for finding the parameters for the $\overline{X}$, EWMA, and CUSUM control charts under the optimal economic design framework (Duncan 1956; Lorenzen and Vance 1986).

Other, more specialized packages provide functionalities that may find use in specific applications. The **spcadjust** package (Gandy and Kvaløy 2013, 2017) addresses estimation error in control limit calculation for Shewhart, CUSUM, and EWMA control charts. The package also allows the control charts to be applied on the residuals of a linear regression, which can be used for monitoring profiles. The **funcharts** package (Capezza, Centofanti, Lepore, Menafoglio, Palumbo, and Vantini 2023a,b) specializes in control charts for multivariate functional data, using either Hotelling's T^2 or the squared prediction error to detect process changes. This package provides control charts for various scenarios involving functional data, including scalar responses with functional covariates (Capezza, Lepore, Menafoglio, Palumbo, and Vantini 2020), functional response variables with functional covariates (Centofanti, Lepore, Menafoglio, Palumbo, and Vantini 2021), and general functional data monitoring. The

surveillance package (Salmon, Schumacher, and Höhle 2016; Meyer, Held, and Höhle 2017) provides methods for change detection in public health surveillance using various models and tools for time series analysis, univariate and multivariate generalized linear models (GLM), and generalized additive models. Other packages such as **cpm** (Ross 2015), **strucchange** (Zeileis, Leisch, Hornik, and Kleiber 2002), **bcp** (Erdman and Emerson 2008), and **changepoint** (Killick and Eckley 2014; Killick, Haynes, and Eckley 2022) provide methods for process monitoring based on the change-point model (Hawkins, Qiu, and Kang 2003). Finally, the **dfphase1** package (Capizzi and Masarotto 2018) allows change detection in retrospective univariate and multivariate samples using nonparametric control charts.

Limited support of SPM methodologies is available in other programming languages: The **PySpc** package (Silva 2023) provides a Python (Van Rossum and Drake Jr 1995) implementation of classical univariate and multivariate control charts, while the **Pre-Screen** (Yi, Herdsman, and Morris 2019) toolbox for MATLAB (The Mathworks Inc. 2023) offers various control charts based on principal components. Within the Julia language, the **MultivariateAnomalies.jl** (Flach *et al.* 2017) provides some basic tools for multivariate SPM such as Hotelling's T^2 and k-nearest-neighbor-based outlier measures.

The remainder of this paper is organized as follows: Section 2 provides an introduction to SPM, followed by a discussion on common challenges encountered in control chart design. Section 3 presents a comprehensive examination of the **StatisticalProcessMonitoring.jl** package and the underlying design principles of its interface. Section 4 demonstrates some applications of the proposed package in settings such as joint monitoring of the mean and covariance matrix, monitoring changes in the residuals of autocorrelated data, detecting changes in surgical outcomes, and monitoring of nonparametric profiles. Finally, Section 5 concludes the paper by discussing planned future developments of the package.

2. Statistical process monitoring

2.1. Basic terminology

Control charts are the main tools used to assess process stability within the framework of SPM. According to Woodall (2000), control charts can be broadly separated in Phase I and Phase II control charts. Phase I control charts are used on historical data to remove any anomalous patterns and obtain a clean dataset $\mathcal{X}^{(0)} = \{\boldsymbol{X}_{-m+1}, \boldsymbol{X}_{-m+2}, \dots, \boldsymbol{X}_0\}$ which is representative of the process under stable conditions (in-control, IC). Here, $\boldsymbol{X}_i$ for $i > -m$ denotes the i-th observed sample from the sequential process. This cleaned dataset is then typically used to estimate certain IC parameters of the process. Phase II control charts instead focus on the use of control charts to monitor deviations from the IC state as new data $\boldsymbol{X}_1, \boldsymbol{X}_2, \dots,$ are sequentially collected over time. An alarm is then raised when the statistic detects a change in the process, indicating it has shifted to an out-of-control (OC) state. This paper is mainly concerned with the use of control charts in Phase II analysis, although the package herein introduced can be adapted to also handle Phase I analysis.

Phase II control charts typically involve the sequential calculation of a monitoring statistic C_t, $t = 1, 2, \dots,$ and an OC alarm is raised as soon as C_t falls outside control limits $(\mathrm{LCL}_t, \mathrm{UCL}_t)$, where $\mathrm{LCL}_t, \mathrm{UCL}_t \in \mathbb{R} \cup \{-\infty, \infty\}$. For simplicity, we will assume that $\mathrm{LCL}_t \leq 0$ and $\mathrm{UCL}_t \geq 0$ for all $t > 0$. To ensure these properties are satisfied, straightforward transformations of the monitoring statistic can be applied. Furthermore, one-directional control charts can be defined using decision intervals of the form $(-\infty, \mathrm{UCL}_t)$ or (LCL_t, ∞). Users are typically interested in the control chart's run length (RL),

$$\mathrm{RL} = \inf\left\{t > 0 : C_t > \mathrm{UCL}_t \text{ or } C_t < \mathrm{LCL}_t\right\},$$

which is a random variable representing the number of time points required for the monitoring procedure to signal an alarm.

The control limit(s) that define the decision interval are typically selected so as to constrain some IC properties of the control chart's run length to a nominal value. The most common way of determining the decision thresholds is by imposing

$$\mathrm{ARL}_{\mathrm{IC}} := \mathsf{E}_0[\mathrm{RL}] = A_0,$$

where $A_0 > 1$ and $\mathsf{E}_0[\cdot]$ represents expectation assuming that the process always remains IC. Other metrics such as the IC median run length ($\mathrm{MRL}_{\mathrm{IC}}$, see Waldmann 1986a,b; Gan 1993, 1994; Graham, Mukherjee, and Chakraborti 2017; Hu, Castagliola, Tang, and Zhong 2021; Qiao, Sun, Castagliola, and Hu 2022) or quantiles of the IC run length (Knoth 2015) may also be used in place of the $\mathrm{ARL}_{\mathrm{IC}}$.

2.2. Taxonomy of control charts

Following the exposition of Chakraborti, Van Der Laan, and Bakir (2001), control charts can broadly be classified in three main categories: Shewhart-type control charts, CUSUM-type control charts, and EWMA-type control charts.

Shewhart-type control charts are memoryless control charts, i.e., at each time $t > 0$ the monitoring statistic C_t is calculated using only the information about $\boldsymbol{X}_t$. Since historical information about $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{t-1})$ is discarded at time t, these control charts are typically reactive to large changes in the process parameters. On the other hand, persistent small parameter shifts are much harder to detect using Shewhart-type control charts.

In contrast, CUSUM- and EWMA-type control charts are memory-type control charts. They use the whole history $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t)$ to compute the current value C_t of the monitoring statistic. This enables them to detect persistent shifts of smaller magnitude more efficiently. Due to the continuous accumulation of information, memory-type control charts require a discounting mechanism to mitigate the dampening effect of the historical data on the process parameter shift (see Qiu 2013).

CUSUM-type control charts discount historical information by employing a restarting mechanism. This is typically implemented using a recursive relation of the monitoring statistic, such as

$$C_t = \max\left\{0, C_{t-1} + f(\boldsymbol{X}_t)\right\},$$

where f is a scalar function of the incoming observation $\boldsymbol{X}_t$. Once the monitoring statistic is set to zero, the values prior to the last restarting time ("spring length", see also Chatterjee and Qiu 2009; You and Qiu 2019) do not influence future values of C_t.

On the other hand, EWMA-type control charts discount past information by using an exponentially decaying weighting scheme for past observations. The classical EWMA statistic for a standardized univariate process X_t defines the value of the monitoring statistic, C_t, as

$$C_t = (1 - \lambda)C_{t-1} + \lambda X_t,$$

where $\lambda \in (0, 1)$. With this smoothing mechanism, at time t, a weight $w_j(t) = \lambda(1-\lambda)^j$ is assigned to observation X_{t-j} for $j = 0, \ldots, t-1$.

2.3. Nonparametric control charts

Traditional control charts assume independent and identically distributed (i.i.d.) continuous quality variables that follow a parametric distribution. This is the case for most of the existing packages discussed in Section 1. However, when these assumptions are violated, control charts designed under these assumptions suffer from serious limitations (Crosier 1988; Lowry *et al.* 1992). Real-world process observations often exhibit serial correlation (Capizzi and Masarotto 2009; Qiu and Xie 2021) and have multivariate distributions that are not easily described by parametric models (Yang and Qiu 2020). As a result, various methodologies have been developed to address these issues and expand the applicability of SPM (Chakraborti *et al.* 2001; Chakraborti and Graham 2019).

Nonparametric control charts based on ranks (Qiu and Hawkins 2001; Zou and Tsung 2010; Li, Pu, Tsung, and Xiang 2017) and data categorization (Qiu 2008; Li, Tsung, and Zou 2012; Wang, Li, and Su 2017) have been proposed for cases where parametric distributions are inadequate. These nonparametric charts can be applied without strong parametric assumptions. However, their effectiveness in change detection during online monitoring may be reduced by the loss of information compared to parametric charts (Xie and Qiu 2022). When process observations are autocorrelated, control charts can be developed using either parametric time series approaches (Capizzi and Masarotto 2008; Lee and Apley 2011; Prajapati and Singh 2012) or nonparametric decorrelation methods (Qiu and Xie 2021; Xue and Qiu 2021; Xie and Qiu 2023).

Note that, as pointed out by Zou and Tsung (2011), any control chart can be designed to be distribution-free by assuming the reference sample to be large enough to obtain accurate estimates of the IC distribution. Then, one can achieve the desired nominal IC run length distribution by means of simulations from either the estimated IC distribution or the historical data themselves. In the latter case, standard techniques such as the classical bootstrap (Efron and Tibshirani 1993) as well as bootstrap for autocorrelated data (Politis and Romano 1992; Bühlmann 2002) may be used.

2.4. Selecting the hyperparameters

Control charts used in SPM often require choosing the value of tuning parameters $\boldsymbol{\zeta} \in \mathcal{Z} \subseteq \mathbb{R}^d$. Common examples are the smoothing constant λ in the EWMA control chart and the allowance constant k in the CUSUM control chart. These parameters are crucial for efficiently detecting process changes.

The choice of tuning parameters is typically done to efficiently detect a specific magnitude of parameter change. This optimization problem is usually formulated as follows,

$$\begin{gathered} \frac{\partial \mathsf{E}_1[\mathrm{RL}]}{\partial \boldsymbol{\zeta}}\bigg|_{\boldsymbol{\zeta}=\boldsymbol{\zeta}^*} = \mathbf{0}, \\ \text{s.t. } \mathsf{E}_0[\mathrm{RL}] = A_0, \end{gathered} \tag{1}$$

where $\mathsf{E}_1[\cdot]$ indicates expectation under the desired OC scenario. While Equation 1 is formulated in terms of average run length, the median run length and other quantiles of the run length can be considered (Knoth 2015).

Optimally designing a control chart can be challenging as analytical solutions are typically only available for simple, univariate cases. In other cases, methods based on integral equations (Rigdon 1995a,b; Knoth 2017) may be employed. However, these methods depend on specific distributional assumptions on the sequential process and may not be applicable in general.

For complex control charts applied to general processes, methods based on Monte-Carlo simulations can be utilized to solve the optimal design problem in Equation 1. These methods involve estimating the OC average run length using a large number of simulated run lengths, under the constraint of the IC average run length. Once the OC average run length has been estimated, the control chart parameter can be optimized using approaches such as grid search (Qiu 2008) or classical numerical solvers (Capizzi and Masarotto 2003; Mahmoud and Zahran 2010).

In general, optimizing the constrained function in Equation 1 presents several challenges. Firstly, as a stochastic optimization problem, finding the optimal solution $\boldsymbol{\zeta}^*$ is generally more difficult than ordinary numerical optimization. Additionally, the appropriate value of the control limit to satisfy the IC constraint depends on the value of the hyperparameter. This significantly increases the computational cost of evaluating the objective function.

2.5. Multi-chart monitoring schemes

Multi-chart monitoring schemes consist of multiple control charts that are run simultaneously, each with its associated control limit(s). An alarm is triggered as soon as one of the monitoring statistics falls outside the decision boundaries.

Multi-chart monitoring schemes are useful when multiple parameters need to be monitored jointly, such as the mean and variance of a distribution (Gan 1995). Additionally, multiple control charts with varying sensitivities to parameter shifts can be combined for detecting both small and large parameter changes efficiently. Some examples include combining a Shewhart chart with an EWMA chart (Lucas and Saccucci 1990; Reynolds and Stoumbos 2008), combining a Shewhart chart with a CUSUM chart (Lucas 1982), or using multiple CUSUM charts with different allowance constants (Lorden and Eisenberger 1973; Sparks 2000; Zhao, Tsung, and Wang 2005).

The design of the control limits for multi-chart monitoring schemes is typically based on a joint run length criterion with a constraint on the run lengths of the individual control charts (Capizzi and Masarotto 2016). Specifically, if $J > 1$ control charts are run simultaneously, let RL_k indicate the run length of the k-th control chart, for $k = 1, \ldots, J$. Then, the control limits $\mathbf{LCL}_t = (\mathrm{LCL}_{t1}, \ldots, \mathrm{LCL}_{tJ})^\top$ and $\mathbf{UCL}_t = (\mathrm{UCL}_{t1}, \ldots, \mathrm{UCL}_{tJ})^\top$ can be determined so that

$$\mathsf{E}_0\left[\min(\mathrm{RL}_1, \mathrm{RL}_2, \ldots, \mathrm{RL}_J)\right] = A_0, \tag{2}$$

and

$$\mathsf{E}_0[\mathrm{RL}_1] = \mathsf{E}_0[\mathrm{RL}_2] = \ldots = \mathsf{E}_0[\mathrm{RL}_J]. \tag{3}$$

These equations ensure that the IC average run length of the joint monitoring scheme of the overall scheme is equal to the nominal value ARL_0, and that the expected run length for each control chart is the same. It is worth noting that Equations 2 and 3 can be based on other measures, such as run length quantiles, instead of the average run length. Furthermore, by introducing weighting schemes in Equation 3, it is possible account for the relative importance of each control chart.

Designing joint monitoring schemes typically involves the use of traditional algorithms like bisection search (see, for example, Qiu 2013). Nevertheless, this method may involve substantial computational overhead compared to the design of single control charts. Recent research (Capizzi and Masarotto 2016) has addressed this concern by introducing a novel approach for estimating control limits using stochastic approximations (see Chen 2002; Kushner and Yin 2003, for an introduction to stochastic optimization).

3. The StatisticalProcessMonitoring.jl package

This paper introduces a new package named **StatisticalProcessMonitoring.jl**, implemented in the Julia programming language (Bezanson, Edelman, Karpinski, and Shah 2017). The choice of Julia stems from the "two-language" problem encountered in interpreted, dynamically typed languages such as R. While these languages support fast function prototyping, there is often a drawback in terms of computational efficiency. Therefore, users often resort to quickly writing a proof-of-concept code in a high-level language and then rewriting it in a lower-level, strongly-typed language such as C++ (Stroustrup 2013) or Fortran (Ellis, Philips, and Lahey 1994) to achieve better performance.

Julia is a programming language designed specifically for scientific computing (Bezanson *et al.* 2017). It achieves computational speed comparable to lower-level languages by combining just-in-time compilation with a sophisticated type system, enabling efficient execution of numerical computations. The Julia language uses multiple dispatch for function implementation (Bezanson *et al.* 2017), where functions are composed of multiple methods with distinct type signatures. At runtime, the most specific method for a function call is selected by comparing argument and parameter types. Unlike overloading in C++, Julia resolves overloading dynamically using multiple dispatch rather than statically.

Julia's hierarchy of base and abstract types offer a high-level, expressive syntax that promotes concise and readable code. Its flexible and intuitive design makes it easier for users to understand and modify the implementation of control charts in the **StatisticalProcessMonitoring.jl** package. This allows users to easily customize and extend the functionalities according to their specific requirements. Finally, Julia has a rapidly expanding ecosystem with a wide range of packages and libraries dedicated to numerical algebra, optimization, and visualization tools.

The following sections present a comprehensive overview of the design philosophy of the **StatisticalProcessMonitoring.jl** package. The aim of this explanation is to clarify how the proposed framework enables very general control chart functionalities. Furthermore, we will discuss the main features of the **StatisticalProcessMonitoring.jl** package, which include automatic control limit estimation and hyperparameter optimization.

3.1. Overview of StatisticalProcessMonitoring.jl

Type hierarchy

The **StatisticalProcessMonitoring.jl** package introduces a flexible definition of a control chart, which aims at separating the basic building blocks commonly found in control charts. The main type introduced by the package is the `ControlChart` type, whose attributes determine the main properties required by a control chart:

```
mutable struct ControlChart{STAT, LIM, NOM, PH2} <:
    AbstractChart{STAT, LIM, NOM, PH2}
  stat::STAT
  limit::LIM
  nominal::NOM
  phase2::PH2
  t::Int
end
```

In Julia, the `<:` operator is used in the definition of a new type to specify its direct supertype. The `ControlChart` type contains five attributes: i) the monitoring statistic (Section 3.5); ii) the control limit (Section 3.2); iii) the nominal property of interest (Section 3.3); iv) the desired Phase II data simulator (Section 3.4); and v) the indicator of the current time point t. Note that attributes i)-iv) are specified as parametric types, meaning that their type is defined in the moment an object of type `ControlChart` is instantiated. This allows Julia's multiple dispatch functionality to compile functions into high-performance code.

The `ControlChart` type is defined as `mutable`, allowing its elements to be modified and updated by other functions. This is particularly useful for defining the generic update mechanism of a control chart at time t whenever a new data point $\boldsymbol{X}_{t+1}$ is observed. The `update_chart!` function is an example of such a mechanism, which modifies an object of type `ControlChart` by incrementing the time indicator and updating the monitoring statistic using the incoming observation,

```
function update_chart!(CH::AbstractChart, x)
  CH.t += 1
  update_statistic!(get_statistic(CH), x)
end
```

Note that, following Julia's function naming convention (Bezanson *et al.* 2017), the name of functions that modify any of their arguments ends with a "`!`". In the `update_chart!` function, the roles of `update_statistic!` and `get_statistic` are clearly understood.

Using this approach, a new monitoring statistic type simply requires an appropriate specialization of the `update_statistic!` function to be used as part of the `ControlChart` object. Moreover, the exact type of the incoming observation `x` is left unspecified. This argument can be, for example, a real number for univariate or a vector for multivariate control charts. In addition, as demonstrated later in Section 4.4, the argument `x` can also be of a more complicated data type. This design allows the package to handle monitoring of structured data types such as profiles, etc.

Example 3.1. As an example to illustrate the flexibility of the proposed framework, consider a univariate EWMA control chart and an univariate adaptive exponentially weighted moving average (AEWMA) control chart (Capizzi and Masarotto 2003). Assume for simplicity that the two control charts share the same control limit, nominal property of interest, and Phase II object. The only difference between the two control charts is the different monitoring statistic objects, respectively of type `EWMA` and `AEWMA`, used in the construction of the control chart. These types can be defined, for example, as follows:

```
mutable struct EWMA <: AbstractStatistic
  λ::Float64
  value::Float64
end

mutable struct AEWMA <: AbstractStatistic
  λ::Float64
  k::Float64
  value::Float64
end
```

The computation of the quantities necessary to update the value of the monitoring statistic is handled by defining suitable specializations of the `update_statistic!` function:

```
function update_statistic!(stat::EWMA, x::Real)
  stat.value = (1.0 - stat.λ) * stat.value + stat.λ * x
end

function update_statistic!(stat::AEWMA, x::Real)
  stat.value = stat.value + huber(x - stat.value, stat.λ, stat.k)
end
```

In the above code snippet, the `huber` function implements Huber's score function (Huber 1981),

$$\phi_{\text{hu}}(e, \lambda, k) = \begin{cases} e + (1-\lambda)k & \text{if } e < -k \\ \lambda e & \text{if } |e| \leq k \\ e - (1-\lambda)k & \text{if } e > -k. \end{cases}$$

It is important to note that, despite substantially changing the behavior of the resulting control chart, the use of different monitoring statistics leaves the rest of the interface of the `ControlChart` type unchanged. Therefore, users can define new monitoring statistics and immediately integrate them with the existing control limits, Phase II data samplers, and other features of the **StatisticalProcessMonitoring.jl** package discussed in the next sections.

In general, a complete specification of the five elements that compose an object of type `ControlChart` allows the user to define the behavior of a wide range of control charts, as will be documented in the following sections.

Implementation of multi-chart monitoring schemes

The `ControlChart` type is general enough to also accommodate multi-chart monitoring schemes such as those discussed in Section 2.5. Observe that a multi-chart scheme can simply be defined by instantiating an object of type `ControlChart` using a `Tuple` of $J > 1$ monitoring statistics, each with its own control limit. To facilitate this, we define a convenient type alias to represent a multi-chart:

```
const MultipleControlChart{S,L,N,P} = ControlChart{S,L,N,P} where
  {S<:Tuple,L<:Tuple,N,P}
```

It should be emphasized that the discussions regarding the nominal properties of the run length and Phase II data samplers, as presented in Section 3.3 and Section 3.4, are also applicable to the `MultipleControlChart` type. To handle joint criteria such as the one in Equations 2 and 3, several algorithms are available for determining the appropriate values of the control limits, as discussed in Section 3.2.2.

3.2. Control limits

Types of control limits

Control limits play a crucial role in SPM as they define the boundaries within which a process is considered to be in control. In general, control limits can be be either fixed or dynamic (time-varying), with the latter class allowing multiple definitions depending on the desired properties of the decision interval. The **StatisticalProcessMonitoring.jl** package includes various implementations of fixed and dynamic control limits.

Fixed control limits The simplest type of decision interval is defined using constant boundaries over time,

$$\mathrm{LCL}_t = \mathrm{LCL}, \text{ and } \mathrm{UCL}_t = \mathrm{UCL} \quad \text{for all } t = 1, 2, \ldots.$$

Fixed control limits are commonly used for Shewhart-type monitoring statistic or when the analytical expression of the variance of the monitoring statistic C_t is complicated. Examples that utilize fixed control limits include, among others, the CUSUM, MCUSUM, adaptive cumulative sum (ACUSUM, Yi and Qiu 2021), and adaptive cumulative score (ACUSCORE, Capizzi and Masarotto 2012) control charts.

In the **StatisticalProcessMonitoring.jl** package, fixed control limits are implemented as the `TwoSidedFixedLimit` and `OneSidedFixedLimit` types. The latter allows choosing between an upward decision interval $(-\infty, \mathrm{UCL})$ or a downward decision interval (LCL, ∞).

Deterministic time-varying control limits Among dynamic-type control limits, the simplest case is the deterministic time-varying control limit. These are defined as

$$\mathrm{LCL}_t = h \cdot g_l(t), \text{ and } \mathrm{UCL}_t = h \cdot g_u(t), \tag{4}$$

where the functions $g_l(t)$ and $g_u(t)$ typically have a similar functional form as the standard deviation of the control chart statistic at time t. For example, an EWMA control chart with time-varying control limits and fast initial response (Steiner 1999) for a standardized univariate process $(X_t)_{t>0}$ defines the charting statistic as

$$C_t = (1-\lambda)C_{t-1} + \lambda X_t, \quad C_0 = 0.$$

The functions g_l and g_u that determine the time-varying control limits are

$$g_u(t) = \left\{1 - \left(\frac{1}{2}\right)^{\frac{7+3t}{10}}\right\}\sqrt{\frac{\lambda\left[1-(1-\lambda)^{2t}\right]}{2-\lambda}}, \text{ and } g_l(t) = -g_u(t). \tag{5}$$

To implement the fast initial response control limits for the EWMA chart, we can use the `TwoSidedCurvedLimit` and `OneSidedCurvedLimit` types. To implement the control limits in Equations 4 and 5 with $h = 1$ and $\lambda = 0.2$, one can use the following code:

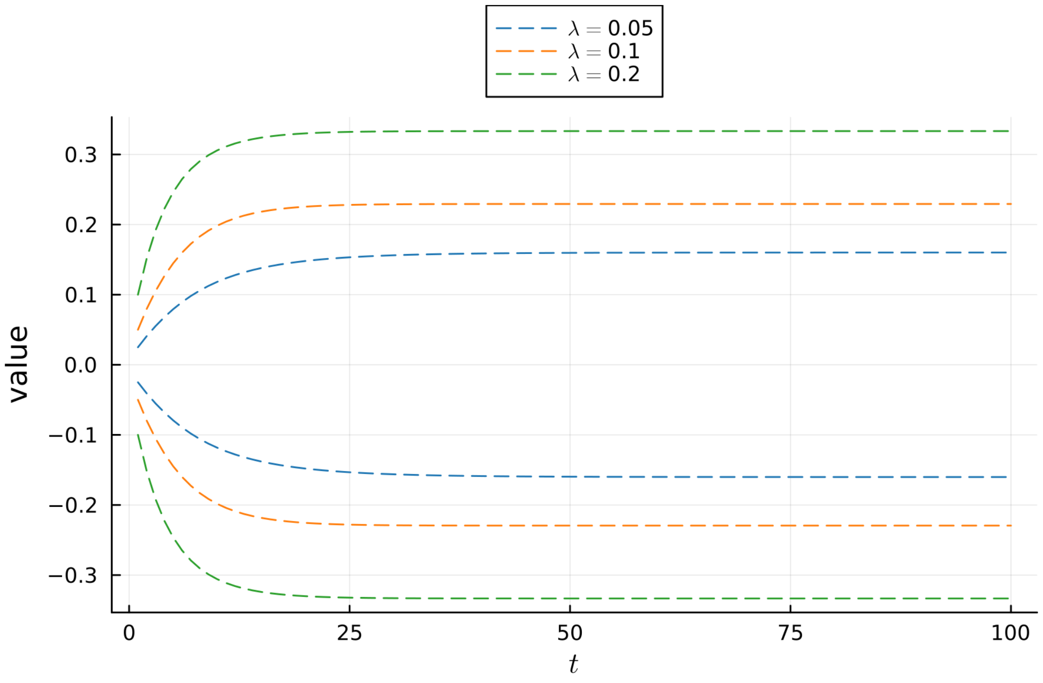

Figure 1: Two-sided control limits for the EWMA control chart with fast initial response using $\lambda = 0.05, 0.1$, and 0.2 and $h = 1$, as $t = 1, \ldots, 100$.

```
julia> λ = 0.2
julia> f(t) = (1 - 0.5^((7 + 3 * t) / 10)) * sqrt(λ/(2 - λ) *
    (1 - (1 - λ)^(2 * t)))
julia> LIM = TwoSidedCurvedLimit(1.0, f)
```

Figure 1 shows the deterministic time-varying control limits for a fast initial response EWMA control chart using various values of λ, when $h = 1$.

Time-varying control limits with constant false-alarm rate Another type of time-varying control limit is designed to achieve a constant false alarm rate for each $t = 1, 2, \ldots$. More formally, $(\mathrm{LCL}_t, \mathrm{UCL}_t)$ are defined such that, for all $t = 1, 2, \ldots$,

$$\Pr(C_t > \mathrm{UCL}_t \text{ or } C_t < \mathrm{LCL}_t \mid \mathrm{LCL}_s \leq C_s \leq \mathrm{UCL}_s \text{ for all } s < t\ ) = \alpha, \quad \alpha \in (0, 1).$$

The constant false alarm property ensures that the run length of the control chart approximately follows a Geometric(α) distribution. Therefore, $\mathrm{ARL}_{\mathrm{IC}} = 1/\alpha$ and $\mathrm{MRL}_{\mathrm{IC}} = \lceil -1/\log_2(1-\alpha) \rceil$. Control limits that satisfy the above property can be obtained using a sequential bootstrap resampling approach to approximate the IC distribution of the monitoring statistic at each time $t = 1, 2, \ldots$ (Shen, Zou, Jiang, and Tsung 2013; Shen, Tsui, Zou, and Woodall 2016; Yang, Zou, and Wang 2017).

These control limits are implemented in the package as the `TwoSidedBootstrapLimit` and `OneSidedBootstrapLimit` types. For both these types, a convenient constructor is used to initialize the control limits using the monitoring statistic. For example, recalling the `EWMA` statistic from Example 3.1, a `TwoSidedBootstrapLimit` using 10000 bootstrap replications can be initialized as

```
julia> STAT = EWMA(0.2, 0.0)
julia> LIM = TwoSidedBootstrapLimit(STAT, 10000)
```

For the `TwoSidedBootstrapLimit`, the control limits at each time $t = 1, 2, \ldots$, are determined by taking the $\alpha/2$- and $((1-\alpha)/2)$-level quantiles of the simulated IC distribution of the monitoring statistic at time t. On the other hand, the `OneSidedBootstrapLimit` uses either the $(1-\alpha)$-level quantile for the one-sided upward decision interval or the α-level quantile for the one-sided downward decision interval. Here, $\alpha = 1/\text{ARL}_0$, where ARL_0 is the desired nominal value of ARL_{IC}. On the other hand, if the desired nominal characteristic of interest is the MRL_{IC}, we can follow the same procedure using $\alpha = 1 - 2^{-1/\text{MRL}_0}$.

Control limit design

In contrast to bootstrap-based control limits, fixed and deterministic time-varying control limits require a design step to determine their appropriate values prior to the monitoring phase. One notable feature of the **StatisticalProcessMonitoring.jl** package is its ability to automatically estimate control limit values that satisfy the desired nominal IC property . This estimation is accomplished using Monte-Carlo methods to simulate new Phase II data, thus approximating the IC run length distribution of the control chart in a fully-automatic way. Simulation of Phase II observations can be achieved either by sampling from a distribution or by resampling data from a reference IC sample. To specify the simulation method, an object that is a subtype of `AbstractPhase2` can be used, as will be explained in Section 3.4.

In the following, we give a very brief overview of the most common methods for estimating control limits.

Bisection search The bisection search algorithm is the most common method for finding the control limit of a control chart (see, for example, Qiu 2013). Suppose we have a control limit of the form $(0, h)$. Extension of this method for symmetric two-sided decision intervals, as well as deterministic time-varying control limits such as Equation 4 are immediate.

The search begins by specifying an initial interval $[0, h_U]$ that must contain the parameter value of interest, h^*. To make sure that this property is satisfied, a very wide initial interval may be specified. Then, the bisection search proceeds as follows:

1. At each iteration $k = 1, 2, \ldots$, the current value of the control limit is set to $h^{(k)} = (h_L + h_U)/2$. If $|h^{(k)} - h^{(k-1)}| < \varepsilon_1$, for a small value of $\varepsilon_1 > 0$, the algorithm is terminated and $h^{(k)}$ is returned as the estimated control limit value. Otherwise, the IC run length distribution is determined based on a large number B of simulations. Typically, $B = 1000$ or 10000. The in-control characteristic of the run length G_0 is estimated for each control limit value as $\widehat{G_0}(h^{(k)})$. When $G_0 = \text{ARL}_{\text{IC}}$, we use the estimate
$$\widehat{\text{ARL}_{\text{IC}}}(h) = \frac{1}{B}\sum_{b=1}^{B} r_b,$$
where r_b represents the b-th simulated run length. For the p-level quantile of the run length, the estimate is taken as $r_{(N)}$, where $N = [B \cdot p]$ and $r_{(i)}$ indicates the i-th ordered value of the simulated run lengths.

2. If the estimate is included in the interval $[A_0-\varepsilon_2, A_0+\varepsilon_2]$, for a small value of $\varepsilon_2 > 0$, the algorithm is stopped and $h^{(k)}$ is returned as the estimate of the control limit. Otherwise, the interval boundaries are adjusted for the next iteration as

$$\begin{cases} h_L = h_L, h_U = h^{(k)} & \text{if } \widehat{G_0}(h^{(k)}) > A_0 \\ h_L = h^{(k)}, h_U = h_U & \text{if } \widehat{G_0}(h^{(k)}) < A_0. \end{cases}$$

   The algorithm then proceeds from step 1.

This algorithm is implemented in the `bisectionCL!` function and in the `bisectionCL` function, which has the same signature as the former but calculates the control limit without modifying the `ControlChart` argument.

Stochastic approximations This algorithm was introduced by Capizzi and Masarotto (2016) for multi-chart monitoring schemes. The stochastic approximations (SA) algorithm aims at finding the vector of control limits $\boldsymbol{h}^* = (h_1^*, \ldots, h_J^*)$ so that multi-chart designs such as the one defined by Equations 2 and 3 are satisfied. This is achieved by simulating, for all individual charts, one run length $\boldsymbol{r}_i = (\boldsymbol{r}_{i1}, \ldots, \boldsymbol{r}_{iJ})$ at a time and applying a gradient descent iteration given by

$$\boldsymbol{h}_{k+1} = \Psi\left(\boldsymbol{h}_k - \frac{1}{(k+1)^q} D\boldsymbol{s}(\boldsymbol{h}_k)\right). \tag{6}$$

Here, $\Psi(\boldsymbol{x}) = \big(\max(0, x_j)\big)_{j=1,\ldots,J}$ is the projection on the positive half-line, $D \in \mathbb{R}^{J\times J}$ is a diagonal gain matrix, $\boldsymbol{s}(\boldsymbol{h}) = \big(s_1(\boldsymbol{h}), \ldots, s_J(\boldsymbol{h})\big) \in \mathbb{R}^J$ is a score vector, and $q \in (0,1)$. For an in-depth discussion on estimating the optimal matrix D and selecting the value of q, see Capizzi and Masarotto (2016). The choice of score $\boldsymbol{s}(\boldsymbol{h})$ is based on the joint design criterion to satisfy. For a joint design based on the $\mathrm{ARL}_{\mathrm{IC}}$ (Equations 2 and 3), the score can be defined as

$$s_i(\boldsymbol{h}) = \frac{\min(r_{i1}, \ldots, r_{iJ}) - A_0}{A_0} + \frac{r_{ij} - \bar{r}_i}{A_0}, \quad j = 1, \ldots, J,$$

where $\bar{r}_i = (r_{i1} + \cdots + r_{iJ})/J$ is the average of the individual simulated run lengths. The first addend of $s_i(\boldsymbol{h})$ moves the current control limit value of every control chart towards a solution that satisfies the run length criterion for the multi-chart monitoring scheme. The second addend is used to enforce the constraint on the individual control charts. For a design based on the p-level quantile of the run length, a score based on the gradient descent iteration in Capizzi and Masarotto (2009) may be used.

In order to enhance the stability of the algorithm, the iterate averaging technique (Ruppert 1991; Polyak and Juditsky 1992) is applied to the control limit values obtained through the gradient descent iteration in Equation 6. Iterate averaging yields the following estimate for the control limit,

$$\bar{\boldsymbol{h}}_k = \frac{1}{k}\sum_{i=1}^{k} \boldsymbol{h}_i, \quad k = 1, 2, \ldots. \tag{7}$$

To determine when to stop the SA algorithm, we rely on the asymptotic distribution of the averaged estimate in Equation 7 (Ruppert 1991; Polyak and Juditsky 1992). For large values of k, approximately,

$$\mathsf{E}[\boldsymbol{s}(\bar{\boldsymbol{h}}_k)] \sim N\left(\boldsymbol{0}, \frac{1}{k}\mathsf{E}\left[\boldsymbol{s}(\boldsymbol{h}^*)\boldsymbol{s}(\boldsymbol{h}^*)^\top\right]\right).$$

Therefore, for $\gamma > 0$, the algorithm can be terminated whenever $|\mathsf{E}[\boldsymbol{s}(\bar{\boldsymbol{h}}_k)]| < \gamma$ using the following stopping rule,

$$\overline{N} = \inf\left\{N > N_{\min} : N > \left(\frac{z}{\gamma}\right)^2 \max_{j=1,\dots,J} \frac{1}{N}\sum_{k=1}^{N} s_{kj}^2\right\}.$$

Here, z is chosen such that $P(-z \leq N(0,1) \leq < z) = 1 - \alpha$ for a small value of $\alpha > 0$, and $N_{\min} > 0$ is used to prevent premature terminations of the algorithm. The SA algorithm is implemented in the `saCL!` and `saCL` functions.

The **StatisticalProcessMonitoring.jl** package provides several methods for determining control limits based on the two algorithms described above. Table 3 lists the available functions for designing control limits.

The documentation of each function provides a detailed explanation of the keyword arguments that can be specified. Accessing the documentation of a function can be done by typing the `?` character in the Julia console and writing the name of the function.

```
julia> ?
help?> saCL!
```

Accepted keyword arguments contain tuning parameters and, among others, tolerance constants.

All methods that calculate the control limits are also implemented using parallel processing. To execute the parallel version of each function, simply set the `parallel` keyword argument to `true`. Since parallelization is achieved using the `Threads.@threads` macro, Julia needs to be launched in multithread mode. As an example, the command `julia -t 4` can be used to start Julia using four threads.

In our experience, the optimal method for designing control limits depends on the specific control chart being used. In general, no single algorithm consistently outperforms the others in all cases. Therefore, providing a variety of control limit estimation methods allows users to select the most suitable algorithm for their particular case.

An example of control limit design for multi-chart monitoring schemes based on the median run length can be found in Section 4.1.

3.3. Nominal properties

Subtypes of the `NominalProperties` type are used to define constraints on the characteristics of the IC run length distribution. An object of type `ARL` specifies the nominal value of ARL_{IC}. The following code defines an object of type `ARL` with a nominal value of 200:

```
julia> NM = ARL(200)
```

Similarly, an object of type `QRL` specifies the nominal value of the p-level quantile of the IC run length, where $p \in (0,1)$. This requires specifying both the nominal value and p, for example as

```
julia> NM = QRL(200, 0.3)
```

Naturally, when $p = 0.5$, the `QRL` type specifies the nominal value of $\mathrm{MRL}_{\mathrm{IC}}$. Objects of type `ARL` and `QRL` are used to modify the behavior of the algorithms that estimate the control limits (Section 3.2.2) as well as the algorithms that tune the control chart hyperparameters (Section 3.5.2) to account for the specific run length characteristic of interest.

For the SA algorithm, when the nominal property is the p-level quantile of the run length, the control limit is determined using the gradient descent iteration described in Capizzi and Masarotto (2009).

3.4. Simulating new observations

Simulation of Phase II data is used for approximating the run length of a control chart, and is achieved by subtyping the abstract type `AbstractPhase2`. Subtyping this class requires the specialization of two methods:

```
new_data(::AbstractPhase2)
new_data!(::AbstractPhase2)
```

These methods specify how new observations should be generated. The **StatisticalProcess-Monitoring.jl** package provides implementations of the `Phase2Distribution` type, which is used to generate data by sampling from any distribution in the **Distributions.jl** package (Lin *et al.* 2019; Besançon *et al.* 2021). For example, the following object can be used to sample data from a standard normal distribution using the `new_data` and `new_data!` functions,

```
julia> using Distributions
julia> PH2 = Phase2Distribution(Normal(0,1))
```

In cases when the underlying data distribution is unknown, `Phase2` type allows resampling data from an IC reference dataset. A `Phase2` object contains an attribute which is a subtype of `AbstractSampling`, which specifies the data resampling mechanism, as well as the IC reference dataset, which can be a `Vector`, a `Matrix`, or a `DataFrame`.

Depending on the sampler object used in creating the `Phase2` object, data can be sampled using different techniques. To handle the most common practical scenarios, the `Bootstrap` type implements the classical bootstrap (Efron and Tibshirani 1993), the `BlockBootstrap` type allows circular block bootstrap (Politis and Romano 1992), and the `StationaryBootstrap` type implements stationary bootstrap (Politis and Romano 1994). Objects of type `BlockBootstrap` and `StationaryBootstrap` are constructed by specifying an integer, which represents, respectively, the block size and the expected block size.

For example, Phase II data can be resampled from a reference IC sample of 500 normally-distributed observations by applying the `new_data` and `new_data!` functions on the following object,

```
julia> PH2 = Phase2(Bootstrap(), randn(500))
```

To automate the simulation of run lengths, the **StatisticalProcessMonitoring.jl** package provides a convenient `run_sim` function. This function simulates an in-control run length for a `ControlChart` object using the `new_data!` function applied to its Phase II attribute.

3.5. Monitoring statistic

The monitoring statistic attributed is subtyped from the `AbstractStatistic` type. All monitoring statistics implement a specialization of the `update_statistic!` function, following the signature provided by the abstract type interface

```
update_statistic!(::AbstractStatistic, x)
```

Furthermore, the value of the monitoring statistic at time $t = 0, 1, 2, \dots$ is stored in the `value` attribute of the object. If a new monitoring statistic subtype is defined which stores its value in a different attribute, then the `get_value` function must be specialized for that particular type to reflect the change.

A list of monitoring statistics implemented in the **StatisticalProcessMonitoring.jl** package can be found in Appendix A.

Statistics with estimated parameters

Monitoring statistics can incorporate estimation of the process parameters such as mean, variance, and other quantities into their definition. However, in some specific cases it is possible to separate the monitoring statistic and the parameter estimation to promote code compartmentalization.

As an example to illustrate this concept, we consider an EWMA statistic applied to incoming univariate observations X_t,

$$C_t = (1 - \lambda)C_{t-1} + \lambda Z_t, \quad C_0 = 0, \tag{8}$$

where $Z_t = (X_t - \widehat{\mu}_0)/\widehat{\sigma}_0$ are the standardized observations, and $\widehat{\mu}_0$ and $\widehat{\sigma}_0$ are the estimates of the mean and standard deviation based on an IC reference sample.

Separation of the monitoring statistic from the parameter estimation can be achieved using the following strategy: First, the behavior of the monitoring statistic in Equation 8 when applied to Z_t is defined using a subtype of `AbstractStatistic`. Then, a subtype of the abstract class `ResidualStatistic` is used to estimate the parameters and calculate the residuals. The `ResidualStatistic` is a subclass of `AbstractStatistic`, whose `update_statistic!` function is specialized as follows:

```
function update_statistic!(S::ResidualStatistic, x)
  return update_statistic!(get_statistic(S), residual!(x, S))
end
```

Subtyping a `ResidualStatistic` requires the specialization of the `residual!` function to define how the residuals are calculated. The code below illustrates how standardized residuals may be monitored in location-scale families of distributions using the `LocationScaleStatistic` type,

```
@with_kw mutable struct LocationScaleStatistic{S, M, P} <: ResidualStatistic
  stat::S
  μ::M = 0.0
  Ω::P = 1.0
end
residual!(x, S::LocationScaleStatistic) = S.Ω * (x .- S.μ)
```

In the above code, the `@with_kw` macro is available by importing the **Parameters.jl** package and allows types to be defined with default values for some of their arguments. By providing an object of type `EWMA` as the `stat` attribute, and setting μ and Ω to be the mean and inverse square root of the variance calculated on a reference initial sample, the resulting monitoring statistic defined by the `LocationScaleStatistic` object will be equivalent to the one defined in Equation 8.

This approach promotes code reusability and compartmentalization, allowing different monitoring statistics to be applied to location-scale families. It also facilitates the implementation of various residual-based control charts, such as regression-adjusted control charts (see Grigg and Farewell 2004, for a review), control charts for residuals of time series models (Wiel 1996; Lu and Reynolds 1999; English, Lee, Martin, and Tilmon 2000; Apley and Lee 2008), and control charts for monitoring principal components (Ranger and Alt 1996; Scranton, Runger, Keats, and Montgomery 1996; Riaz, Zaman, Mehmood, Abbas, and Abujiya 2021). It is also worth noting that self-starting control charts, in the sense of Hawkins (1987), can be easily implemented following this strategy by updating the parameter estimates alongside the value of the monitoring statistic. This can be easily achieved by specializing the `update_statistic!(::ResidualStatistic, x)` function to update the estimates of the parameters.

Hyperparameter tuning

To solve the optimization problem stated in Equation 1, a large number of OC run lengths are simulated to evaluate the detection performance. For different tuning parameter values, the control limit(s) that satisfy the IC constraint are found using one of the algorithms mentioned in Section 3.2.2. To allow the optimization of the hyperparameters of a monitoring statistic, the `set_design!` function needs to be specialized.

```
set_design!(::AbstractStatistic, ::AbstractVector)
```

The `set_design!` function allows for a unified interface for optimization of the hyperparameters. In **StatisticalProcessMonitoring.jl**, hyperparameter optimization is available via:

1. A multidimensional grid search method based on Algorithm 2 in Qiu (2008), which works by successively refining a grid of points until convergence to a minimum.

2. Any numerical solver that is available as part of the **NLopt.jl** package (Johnson 2007). This package is a wrapper to a powerful C++ library implementing efficient algorithms for global and local optimization. A recommended method when using this option is the bound optimization by quadratic approximation (BOBYQA) algorithm (Powell 2009). This algorithm iteratively computes and optimizes a local quadratic approximation of the objective function.

A high-level interface is available for optimization of the hyperparameters by means of the `optimize_design!` function. The signature of the `optimize_design!` function is defined as follows:

```
optimize_design!(CH::ControlChart,
  rlsim_oc::Function,
  settings::OptSettings = OptSettings(CH);
  optimizer::Symbol = :Grid,
  solver::Symbol = :Bootstrap,
  hmax::Float64 = 100.0,
  kw...
  )
```

In addition to the `ControlChart` object, the function requires the following arguments: A `rlsim_oc` function, which simulates a run length under the target OC scenario; and an optional `settings` object of type `OptSettings`, which controls various aspects of the hyperparameter optimization, such as the bounds on the hyperparameters and the convergence criteria.

Keyword arguments to the `optimize_design!` function include an optional `Symbol` named `optimizer`, which indicates the algorithm to be used for hyperparameter optimization. Currently, supported symbols are `:Grid`, for grid search, as well as any valid optimizer in the **NLopt.jl** package.

In addition, the keyword argument `solver` specifies the root-finding algorithm for control limit estimation. If `solver` is set to `:Bisection`, the maximum value for the control limit can be specified using the `hmax` keyword argument. See Appendix A for a list of available root-finding algorithms. Additional keyword arguments are passed to the root-finding algorithm responsible for estimating the control limit.

To minimize the loading time of **StatisticalProcessMonitoring.jl**, the functions that interface the hyperparameter optimization with the **NLopt.jl** package are implemented as an extension (Bezanson *et al.* 2017). This extension is loaded only if the `using NLopt` command is run, thus reducing the package loading time when hyperparameter tuning is not required.

4. Examples

In this section, four examples are provided to illustrate the use of the **StatisticalProcessMonitoring.jl** package. Before running the examples, we have to make sure that all necessary packages are loaded. These packages contain utility functions that are useful for tasks such as generating random numbers, handling data matrices, plotting, and more.

```
julia> using StatisticalProcessMonitoring, CategoricalArrays, CSV,
   DataFrames, Distributions, LaTeXStrings, LinearAlgebra, NLopt,
   Parameters, Plots, Random
```

4.1. Jointly monitoring the mean and covariance using a multi-chart scheme

Here, we provide an example of multiple control charts being run simultaneously. We consider sequential monitoring of the data from Example 7.7 of Qiu (2013).

In the example, three quality variables resulting from a production process are monitored for changes. Under IC conditions, the quality variables are are assumed to follow the $N_3(\mathbf{0}, \Sigma_0)$

distribution, where

$$\Sigma_0 = \begin{pmatrix} 1.0 & 0.8 & 0.5 \\ 0.8 & 1.0 & 0.8 \\ 0.5 & 0.8 & 1.0 \end{pmatrix}.$$

The Phase II data have been downloaded from the author's website and are also provided in the `example77.csv` file, which is part of the replication material. To illustrate the flexibility of the **StatisticalProcessMonitoring.jl** package, we consider a multi-chart scheme to monitor both the mean and the variance-covariance matrix of the process. The multi-chart scheme consists of three MCUSUM charts (Crosier 1988) for monitoring the mean of the process and a multivariate exponentially weighted moving covariance matrix (MEWMC) control chart (Hawkins and Maboudou-Tchao 2008) for monitoring changes in the process dispersion.

First, we set the seed for replicability.

```
julia> seed = 54397858713
julia> Random.seed!(seed)
```

Then, we initialize the distribution of the observations under IC conditions.

```
julia> μ = [0, 0, 0]
julia> Σ = [1.0 0.8 0.5; 0.8 1.0 0.8; 0.5 0.8 1.0]
julia> DIST = MultivariateNormal(μ, Σ)
```

We then define three CUSUM charts with allowance constants, respectively, equal to 0.25, 0.5, and 1. Furthermore, we define a MEWMC control chart with smoothing constant equal to 0.2.

```
julia> p = length(μ)
julia> STATS = (MCUSUM(k = 0.25, p = p), MCUSUM(k = 0.5, p = p),
   MCUSUM(k = 1.0, p = p), MEWMC(λ = 0.2, p = p))
```

To account for the mean and covariance of the distribution, the four control charts are applied to the standardized observations,

$$\boldsymbol{Z}_t = \Sigma_0^{-1/2}\boldsymbol{X}_t.$$

As discussed in Section 3.5.1, the `LocationScaleStatistic` type provides a convenient interface for standardizing observations in location-scale families. As discussed in Section 3.1.2, multi-chart monitoring schemes are implemented by constructing a `Tuple` of monitoring statistics.

```
julia> Σ_sqm1 = inv(sqrt(Σ))
julia> EST_STATS = Tuple(LocationScaleStatistic(s, μ, Σ_sqm1) for
   s in STATS)
```

Each control chart, for $j = 1, \ldots, 4$, will signal an alarm whenever its value crosses a corresponding upper control limit UCL_j. Similarly to the monitoring statistic, for multi-charts the control limit is also a `Tuple` of individual control limit objects.

```
julia> LIMS = Tuple(OneSidedFixedLimit(1.0, true) for _ in 1:4)
```

The values of $(\text{UCL}_1, \ldots, \text{UCL}_4)$ are found so that the MRL_{IC} of the monitoring scheme is equal to 200, and the MRL_{IC}'s of all individual control charts are equal to each other.

```
julia> NM = QRL(200, 0.5)
```

To find the appropriate values of the control limits, Phase II observations are generated from the IC process distribution.

```
julia> PH2 = Phase2Distribution(DIST)
```

After creating the `ControlChart` object, a bisection approach is used to find the four control limits that satisfy the specified constraints.

```
julia> CH = ControlChart(EST_STATS, LIMS, NM, PH2)
julia> h = bootstrapCL!(CH)
julia> print(h)

(h = [14.568353665174286, 8.627690357206408, 4.677555568948659,
2.360499495019121], iter = 8, status = "Convergence")
```

The joint monitoring scheme is then applied to the Phase II data. To do so, we first load the data contained in the example77.csv file. The first three data columns contain the observations of $\boldsymbol{X}_t$ to be monitored using the multi-chart.

```
julia> data = CSV.read("data/example77.csv", DataFrame)
julia> xnew = Matrix(data)[:, 1:3]
```

The multi-chart is then applied to the data and the results are plotted.

```
julia> proc = apply_chart(CH, xnew)
julia> plt = plot_series(proc, dpi = 300, label = "", xlab = L"t",
   ylab = L"C_t", subtitles = ["MCUSUM k = 0.25", "MCUSUM k = 0.5",
   "MCUSUM k = 1", "MEWMC λ = 0.2"])
```

Figure 2 shows the application of the multi-chart monitoring scheme to the Phase II data. The MEWMC control chart signals a change in the dispersion of the observations starting from the 14th observation. Two of the control charts that monitor the mean of the observations also detect a change later on in the monitoring phase. However, it should be noted that the detected mean change might be a spurious signal. As pointed out by Hawkins and Deng (2009), changes in the process dispersion may erroneously cause the control charts that monitor the mean to trigger an alarm.

4.2. Residual-based monitoring of autocorrelated data

In this section, we demonstrate how the `ResidualStatistic` and `AbstractPhase2` interfaces can be extended to accommodate custom data types. Specifically, we focus on monitoring the residuals of an autoregressive AR(1) model (see, e.g., Shumway and Stoffer 2017),

$$y_t = \phi y_{t-1} + \varepsilon_t, \quad \varepsilon_t \sim N(0, 1),$$

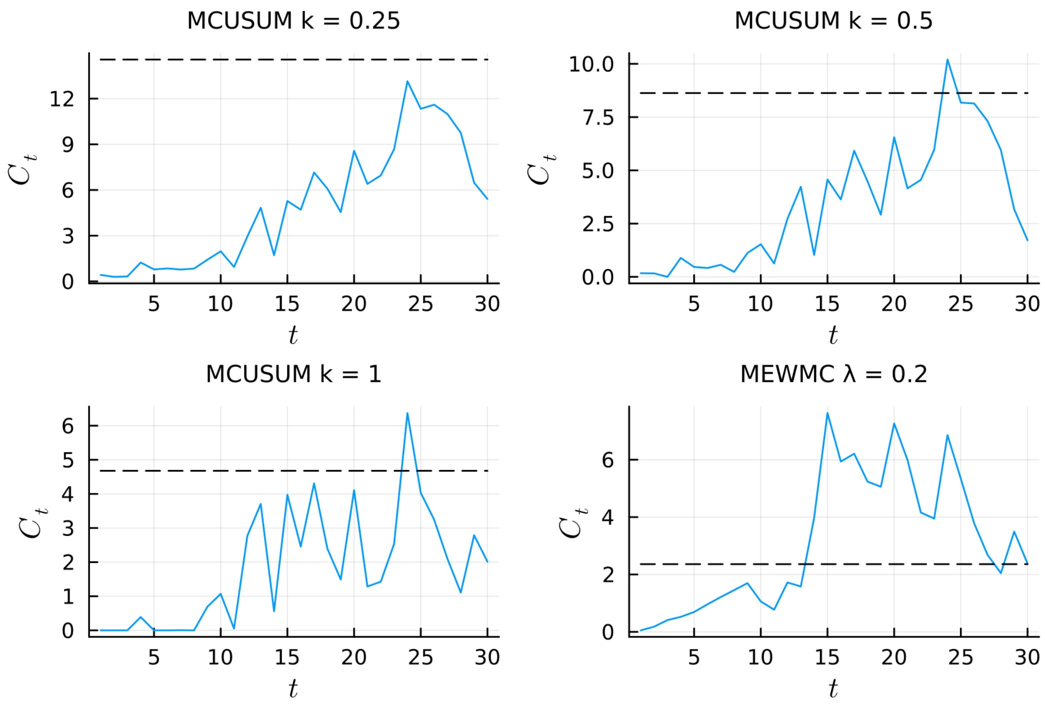

Figure 2: Joint monitoring of mean and covariance for the Phase II data using four control charts. The joint monitoring scheme signals an alarm whenever one of the control charts crosses its corresponding upper control limit.

using an EWMA control chart. To work on the residuals of the AR(1) model, a subtype of the `ResidualStatistic` abstract class will be used. In order to write a specialization of the `residual!` and `new_data!` functions, these must be explicitly imported:

```
julia> import StatisticalProcessMonitoring.residual!,
   StatisticalProcessMonitoring.new_data!
```

We thus define a new type called `AR1Statistic`, which will be used to apply the statistic to the residuals of the AR(1) model.

```
mutable struct AR1Statistic{S} <: ResidualStatistic
  stat::S
  phi::Float64
  ym1::Float64
end

function residual!(x, S::AR1Statistic)
  yhat = x - S.phi * S.ym1
  S.ym1 = x
  return yhat
end
```

To sample observations an AR(1) process, we can define a new type called `Phase2AR1`.

```
@with_kw mutable struct Phase2AR1 <:
  StatisticalProcessMonitoring.AbstractPhase2
  phi::Float64
  y::Float64 = 0.0
  init::Bool = false
end
```

The initial observation y_0 required to sample y_1 is initialized using the stationary distribution $N(0, 1/(1+\phi^2))$ when the `new_data!` function is first called on the `Phase2AR1` object:

```
function new_data!(PH2::Phase2AR1)
  if !PH2.init
    PH2.y = randn() / sqrt(1 - PH2.phi^2)
    PH2.init = true
  end
  yhat = PH2.phi * PH2.y + randn()
  PH2.y = yhat
  return yhat
end
```

The `new_data!` function samples a new observation from the AR(1) process by updating the internal attribute `y` of the `Phase2AR1` object with the last sampled observation.

We now consider an AR(1) model with $\phi = 0.5$. Using the `Phase2AR1` object defined above, we set up the object for simulating run lengths.

```
julia> seed = 4398354798
julia> Random.seed!(seed)
julia> phi = 0.5
julia> PH2 = Phase2AR1(phi = phi)
```

We then define an EWMA control chart applied to the residuals of the AR(1) model, using the `AR1Statistic` object defined previously.

```
julia> STAT = AR1Statistic(EWMA(λ = 0.1), phi, 0.0)
```

The control chart uses a two-sided control limit of the form $(\mathrm{LCL}_t, \mathrm{UCL}_t) = (\mathrm{LCL}, \mathrm{UCL})$ for all $t = 1, 2, \ldots$.

```
julia> LIM = TwoSidedFixedLimit(1.0)
```

Furthermore, we set the $\mathrm{ARL}_{\mathrm{IC}}$ to 500 and create the `ControlChart` object.

```
julia> NOM = ARL(500)
julia> CH = ControlChart(STAT, LIM, NOM, PH2)
```

The smoothing constant of the EWMA control chart is optimized against an anticipated persistent mean shift of $\delta = 2$ To simulate the run length of a process undergoing the location shift, the function `run_sim_oc` provided in the **StatisticalProcessMonitoring.jl** package is used.

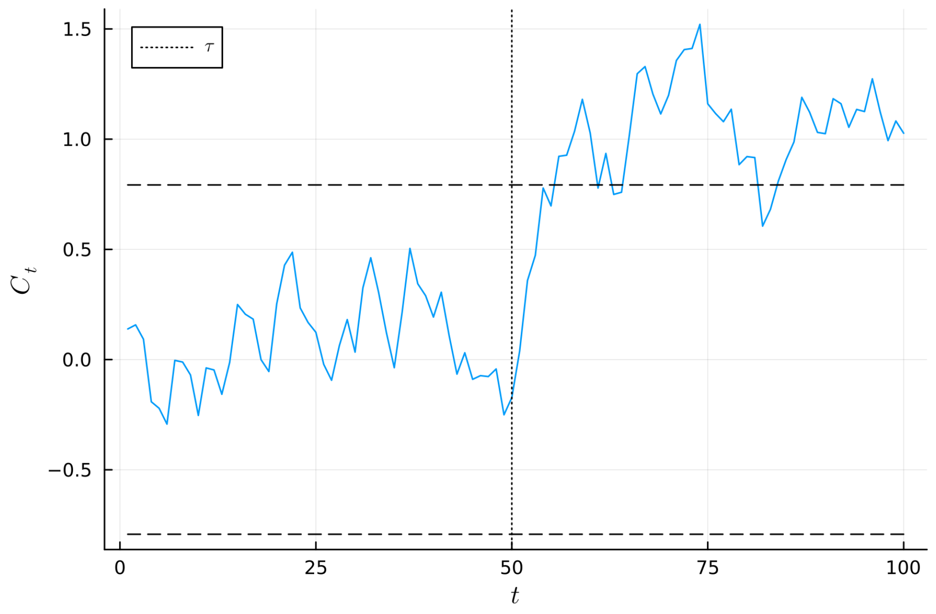

Figure 3: EWMA control chart applied to the 100 residuals of an AR(1) model. The dotted vertical line indicates the change point τ after which observations are shifted by $\delta = 2$. The dashed lines indicate the estimated control limits such that $\text{ARL}_{\text{IC}} = 500$.

```
julia> delta = 2.0
julia> rlsim_oc = x -> run_sim_oc(x, shift = delta)
```

Optimization of the smoothing constant is done using the `optimize_design!` function, which provides a convenient interface for hyperparameter tuning. We initialize an `OptSettings` object to customize the print messages of the optimization algorithm and also impose constraints on the value of the smoothing constant $\lambda \in (0, 1)$.

```
julia> settings = OptSettings(verbose = false, minpar = [0.001],
         maxpar = [0.99])
```

Then, we use the BOBYQA algorithm to find the optimal smoothing constant for the anticipated process shift.

```
julia> opt = optimize_design!(CH, rlsim_oc, settings,
   optimizer = :LN_BOBYQA)
julia> print(opt)
```

```
1-element Vector{Float64}:
 0.13830568163003895
```

During the smoothing constant optimization, the control limit that satisfies the ARL_{IC} constraint is also estimated. Therefore, there is no need to run a preliminary control limit estimation procedure.

We consider 100 Phase II observations from the AR(1) process with $\phi = 0.5$. Of these Phase II data, the first 50 observations are generated under the IC process, while the last 50 undergo a mean shift of $\delta = 2$.

```
julia> n = 100
julia> tau = 50
julia> DGP = Phase2AR1(phi = phi)
julia> y = [new_data!(DGP) for _ in 1:n]
julia> y[(tau+1):n] .+= delta
```

The control chart is then applied to the data and the results are plotted.

```
julia> proc = apply_chart(CH, y)
julia> plt = plot_series(proc, dpi = 300, label = "", xlab = L"t",
   ylab = L"C_t")
julia> vline!(plt, [tau], label = "tau", linestyle = :dot, colour = "black")
```

As seen by Figure 3, the control chart successfully detects a change in the mean of the process shortly after the change-point has occurred.

4.3. Monitoring surgical outcomes using a risk-adjusted CUSUM chart

In this example, we demonstrate the use of a risk-adjusted CUSUM chart using the **StatisticalProcessMonitoring.jl** package. The data comes from a center for cardiac surgery in the United Kingdom and is contained in the `cardiacsurgery` variable from the R package **spcadjust**. The data is also available in the cardiacsurgery.csv file which is found in the replication material.

```
julia> dat = CSV.read("data/cardiacsurgery.csv", DataFrame)
julia> dat.surgeon = categorical(dat.surgeon)
```

The dataset contains information for 5595 recorded surgeries, including the date, surgeon (anonymized), an indicator of whether the patient died during follow-up, the time until patient death (if applicable), and the pre-surgery Parsonnet score. The Parsonnet score is a widely-used scale for predicting the pre-surgical individual risk of death, and is calculated based on a combination of factors such as age and pre-existing medical conditions.

Following the approach of Gandy and Kvaløy (2017), we divide the data into Phase I, for model estimation, and Phase II, for prospective monitoring. Phase I data consists of the first two years of observations, comprising 1769 cases.

```
julia> dat_ic = dat[dat.date .<= 730, :]
```

The goal of this analysis is to monitor potential increases in post-operative patient mortality after adjusting for observed covariates (Steiner *et al.* 2000). To do so, we estimate the post-operative mortality rate on the Phase I data using a logistic regression model with the Parsonnet score as a covariate. Additionally, we include a random intercept to account for each surgeon's individual ability. The mixed-effect logistic model is estimated using the **MixedModels.jl** Julia package (Bates *et al.* 2023).

```
julia> using MixedModels
julia> mod = fit(MixedModel, @formula(status ~ Parsonnet + (1|surgeon)),
   dat_ic, Bernoulli())
julia> print(mod)
```

```
Generalized Linear Mixed Model fit by maximum likelihood (nAGQ = 1)
  status ~ 1 + Parsonnet + (1 | surgeon)
  Distribution: Bernoulli{Float64}
  Link: LogitLink()

   logLik    deviance     AIC       AICc        BIC
  -388.8235   777.6471   783.6471   783.6607   800.0816

Variance components:
           Column   Variance Std.Dev.
surgeon (Intercept)  0.037837 0.194518

 Number of obs: 1769; levels of grouping factors: 6

Fixed-effects parameters:
─────────────────────────────────────────────────────
                  Coef.  Std. Error       z  Pr(>|z|)
─────────────────────────────────────────────────────
(Intercept)  -3.65655    0.17509     -20.88    <1e-96
Parsonnet     0.0818093  0.00723527   11.31    <1e-28
─────────────────────────────────────────────────────
```

For Phase II process monitoring we consider data from the following year, consisting of 779 observations:

```
julia> dat_oc = dat[730 .< dat.date .<= 1095, :]
```

We use a risk-adjusted CUSUM control chart to monitor the incoming observations $(\boldsymbol{X}_i^\top, Y_i)$, where $\boldsymbol{X}_i$ contains the Parsonnet score of the patient and the indicator of the surgeon conducting the surgery, and Y_i is the binary outcome variable. For detecting a change in $\Pr(Y_i = 1 \mid \boldsymbol{X}_i = \boldsymbol{x}_i)$ we use a risk-adjusted CUSUM control chart (Steiner *et al.* 2000; Gandy and Kvaløy 2017). The monitoring statistic is defined as

$$S_t = \max\left\{0, S_{t-1} + R_t\right\}, \quad S_0 = 0,$$

where

$$R_t = \log\left\{\frac{\exp(\Delta + \boldsymbol{x}_t^\top \boldsymbol{\beta})^{Y_t}/(1+\exp(\Delta + \boldsymbol{x}_t^\top \boldsymbol{\beta}))}{\exp(\boldsymbol{x}_t^\top \boldsymbol{\beta})^{Y_i}/(1+\exp(\boldsymbol{x}_t^\top \boldsymbol{\beta}))}\right\} = Y_t\Delta \cdot \log\left\{\frac{1+\exp(\boldsymbol{x}_t^\top \boldsymbol{\beta})}{1+\exp(\Delta + \boldsymbol{x}_t^\top \boldsymbol{\beta})}\right\}. \quad (9)$$

The control chart will signal an alarm whenever $S_t > \text{UCL}$. In Equation 9, $\boldsymbol{\beta}$ is substituted with the estimate $\widehat{\boldsymbol{\beta}}$ based on the Phase I data. The risk-adjusted CUSUM chart is implemented as the `RiskAdjustedCUSUM` type, which requires the specification of the value of Δ, the estimated logistic regression model, and the name of the binary outcome variable.

In this example, we show an application for monitoring increases of mortality rates, which are typically more relevant to detect than decreases. We set up the risk-adjusted CUSUM chart with $\Delta = 0.75$, which roughly corresponds to a doubling of mortality rate for a baseline estimated rate of 5.18%.

```
julia> Random.seed!(239184367)
julia> STAT = RiskAdjustedCUSUM(Δ = 0.75, model = mod, response = :status)
```

For efficient detection of various magnitudes of increase in mortality rate, multiple control charts with different Δ values may also be simultaneously run. We set the IC average run length to 1000 and initialize the control limit as an upper-decision interval.

```
julia> NOM = ARL(1000)
julia> LIM = OneSidedFixedLimit(1.0, true)
```

In-control run lengths for estimating the control limit are simulated by resampling the Phase I data using bootstrap:

```
julia> PH2 = Phase2(Bootstrap(), dat_ic)
```

After creating the `ControlChart` object, the appropriate value of the control limit is found using the SA algorithm:

```
julia> CH = ControlChart(STAT, LIM, NOM, PH2)
julia> h = saCL!(CH, verbose = true, gamma = 0.05)
Running SA ...
Running adaptive gain ...
Estimated gain D = 0.4117860274331409
Running optimization ...
i: 0/50000      h: 2.77856      hm: 0.0 stop: 0
i: 1000/50000   h: 2.86375      hm: 2.9541      stop: 3457
i: 2000/50000   h: 2.94813      hm: 2.95045     stop: 3435
i: 3000/50000   h: 2.92253      hm: 2.95877     stop: 3420
i: 3359/50000   Convergence!
```

The output of the algorithm indicates that convergence was reached before the default maximum number of iterations.

```
julia> print(h)
(h = 2.9568877844997226, iter = 3359, status = "Convergence")
```

Figure 4 shows the risk-adjusted CUSUM control chart applied to the Phase II data. The control chart does not signal any increase in the mortality rate of patients for the considered prospective data.

```
julia> proc = apply_chart(CH, dat_oc)
julia> plt = plot_series(proc, dpi = 300, label = "", xlab = L"t",
   ylab = L"C_t")
```

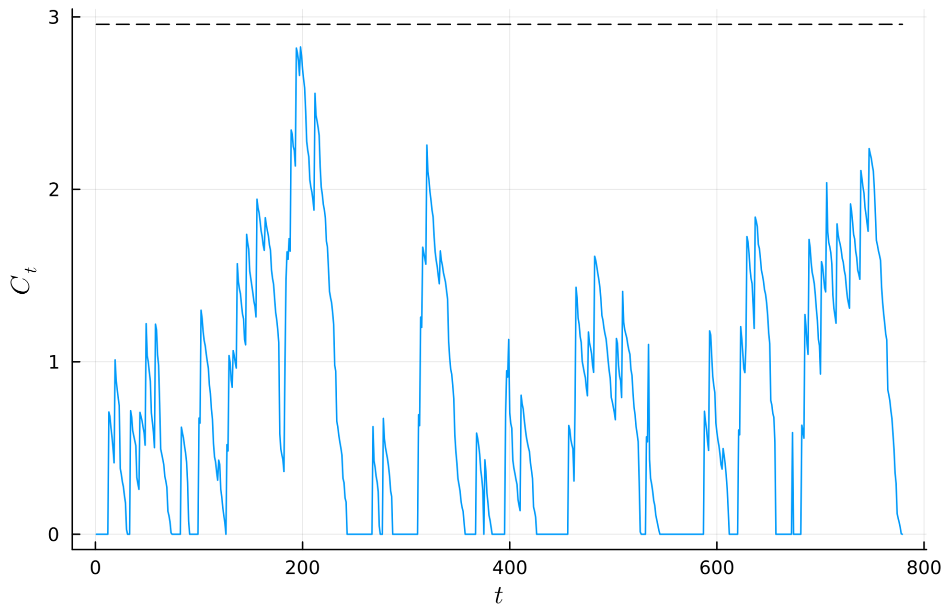

Figure 4: Risk-adjusted CUSUM control chart with $\Delta = 0.75$ applied to the Phase II observations in the dataset. The dashed line indicates the control limit such that $\mathrm{ARL}_{\mathrm{IC}} = 1000$.

4.4. Detecting changes in profiles

As mentioned in Section 3.4, the `Phase2` type is general enough to work with structured data types. In this example, we consider the problem of monitoring profiles. A pair of predictors and responses $(\boldsymbol{X}_t, \boldsymbol{Y}_t)$ with $\boldsymbol{X}_t, \boldsymbol{Y}_t \in \mathbb{R}^{n_t}$ are observed at each time point $t > 0$. Such functional data are typically characterized by a relationship such as

$$Y_{tj} = f(x_{tj}) + \varepsilon_{ij}, \quad j = 1, 2, \ldots, n_t, \tag{10}$$

where $f(x_{tj})$ is the unknown function (profile) relating x_{tj} to Y_{tj}, $\varepsilon_{ij} \overset{\text{iid}}{\sim} F_\varepsilon$, $\mathsf{E}[\varepsilon_{ij}] = 0$, and $\mathrm{Var}[\varepsilon_{ij}] = \sigma > 0$ for all t and j. Naturally, more complicated settings could be considered, for instance by assuming multiple cross-correlated profiles (Zhang, Yan, Lee, and Shi 2018) as well as autocorrelation of the profiles (Qiu *et al.* 2018; Liu, Du, Zang, Zhang, and Wang 2023). Typically, one is interested in determining changes in the function f as well as changes in the variance $\mathrm{Var}[\varepsilon_i]$.

For this example, we assume for simplicity, that $\boldsymbol{x_t}$ are equi-spaced in [0.5, 10] with increments of 0.5 for all $t = -m+1, \ldots, 0, 1, 2, \ldots$.

```
julia> n = 500
julia> nj = 20
julia> x_grid = collect(0.5:0.5:10)
julia> xs = Matrix{Float64}(undef, n, nj)
julia> for i in 1:n; xs[i, :] = x_grid; end
```

After importing the necessary packages and setting the seed for replicability, we consider an initial sample of 500 IC profiles from the functional process (10) using the standard normal distribution for F_ε.

```
julia> Random.seed!(41289355)
julia> ys = sin.(xs) .+ randn(n, nj)
```

StatisticalProcessMonitoring.jl implements a `FunctionalObservation` type that can be used to convenient represent profile data. This data structure contains the sets of predictors $\boldsymbol{x}_t$ and responses $\boldsymbol{y}_t$, as seen from the source code implementation:

```
@with_kw struct FunctionalObservation{A, B}
  x::Vector{A}
  y::Vector{B}
  @assert length(x) == length(y)
end
```

Leveraging Julia's type system, the package provides an alias for representing functional data as a collection of functional observations.

```
const FunctionalData{A,B} = Vector{FunctionalObservation{A,B}} where {A,B}
```

Using the available `FunctionalData` structure and assuming independence of the pairs $(\boldsymbol{X}_t, \boldsymbol{Y}_t)$ and $(\boldsymbol{X}_s, \boldsymbol{Y}_s)$ for any t and s, a suitable Phase II object to simulate run lengths for a functional control chart to monitor the stability of Equation 10 can easily be defined as follows:

```
julia> dat = FunctionalData(xs, ys)
julia> PH2 = Phase2(Bootstrap(), dat)
```

Here, we show an example of profile monitoring using the nonparametric exponentially weighted moving average (NEWMA) control chart (Zou, Tsung, and Wang 2008; Yang *et al.* 2017). The estimate $\widehat{f}$ of the IC profile f is obtained using the locally estimated scatterplot smoothing (LOESS) estimator (Savitzky and Golay 1964; Cleveland 1979), which is implemented in the **Loess.jl** Julia package (Noack, Jones, Arslan, and Fields 2013).

```
julia> using Loess
julia> g = loess(vec(xs), vec(ys), span = 0.3)
julia> plt = plot(minimum(xs):0.01:maximum(xs), (x) -> predict(g, x),
   linewidth = 1.75, label = L"\hat{f}(x)", xlab = L"x", ylab = L"y",
   dpi = 300)
julia> scatter!(plt, vec(xs), vec(ys), markersize = 1, label = "",
   colour = "black")
```

Figure 5 shows the reference sample alongside the nonparametric estimate of the profile using the LOESS estimator. A bandwidth of 0.3 appears to provide a reasonable estimate of the curve without overfitting the data.

The NEWMA control chart is then defined as

$$C_t = \boldsymbol{E}_t^\top \boldsymbol{E}_t, \quad t = 1, 2, \ldots, \quad C_0 = \boldsymbol{0},$$

where $\boldsymbol{E}_t$ is recursively updated using the following EWMA-type smoother,

$$\boldsymbol{E}_t = (1 - \lambda)\boldsymbol{E}_{t-1} + \lambda \boldsymbol{U}_t, \quad t = 1, 2, \ldots. \tag{11}$$

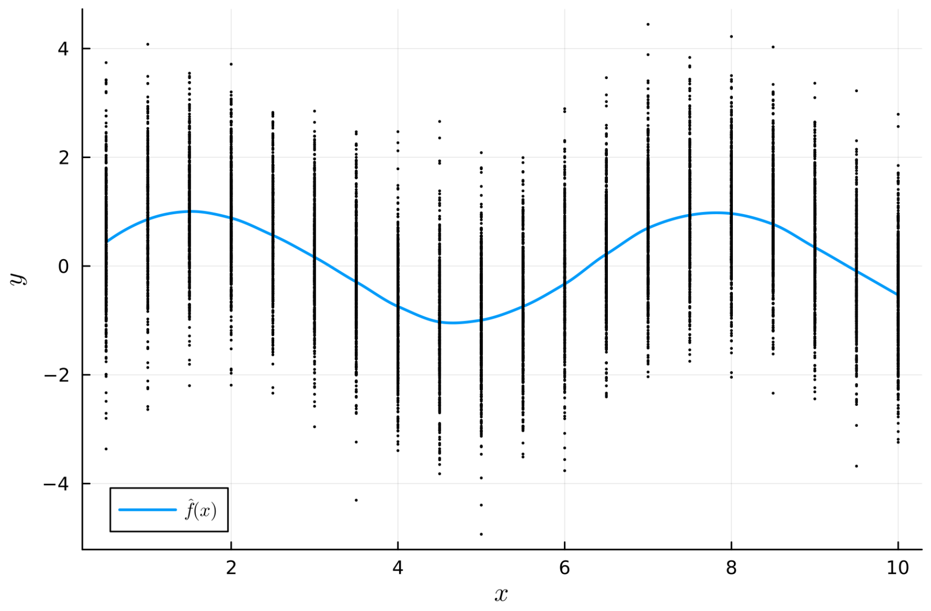

Figure 5: In-control data and estimated profile using the LOESS estimator.

In Equation 11, $\lambda \in (0,1)$ is a smoothing constant and

$$\boldsymbol{U}_t = \begin{pmatrix} \boldsymbol{Z}_t \\ \Phi^{-1}\left(\widehat{F}_0(\widehat{\sigma}_t)\right) \end{pmatrix},$$

where $Z_{tj} = \left(y_{tj} - \widehat{f}(x_{tj})\right)/\sigma_0$ for all $j = 1, \dots, n_t$ is the vector of the standardized residuals from the estimated IC profile $\widehat{f}$, $\widehat{\sigma}_t$ is the estimate of the standard deviation of the residuals $\boldsymbol{Z}_t$, Φ is the cumulative distribution function of the standard Normal distribution, and $\widehat{F}_0$ is an estimate of the IC cumulative distribution function F of σ_t. In the original formulation of the control chart, the authors use a parametric approximation to estimate F. In this example, we estimate F using the empirical cumulative distribution function of $(\widehat{\sigma}_{-m+1}, \dots, \widehat{\sigma}_0)$.

To monitor incoming Phase II observations, we define a NEWMA chart with smoothing parameter $\lambda = 0.2$, consistently with the value used by Zou *et al.* (2008).

```
julia> STAT = NEWMA(0.2, g, dat)
```

We also use an upper decision interval $(-\infty, \mathrm{UCL}_t)$ with constant false-alarm rate (Yang *et al.* 2017), which is sequentially estimated using 1000 bootstrap simulations.

```
julia> LIM = OneSidedBootstrapLimit(STAT, true, 1000)
```

The nominal in-control average run length of the control chart is then set to 500.

```
julia> NM = ARL(500)
```

The `ControlChart` object is then initialized. Since the control limit is dynamically estimated using sequential bootstrap, no preliminary control limit calculation is required before applying the control chart for monitoring new profiles.

Description	Profile
In-control	$Y_{tj} = \sin x_{tj} + \varepsilon_{ij}$
Profile shift	$Y_{tj} = \sin x_{tj} + 2\cos x_{tj} + \varepsilon_{ij}$
Variance shift	$Y_{tj} = \sin x_{tj} + 2\varepsilon_{ij}$.

Table 1: Data-generating processes considered for the Phase II observations in the profile monitoring example.

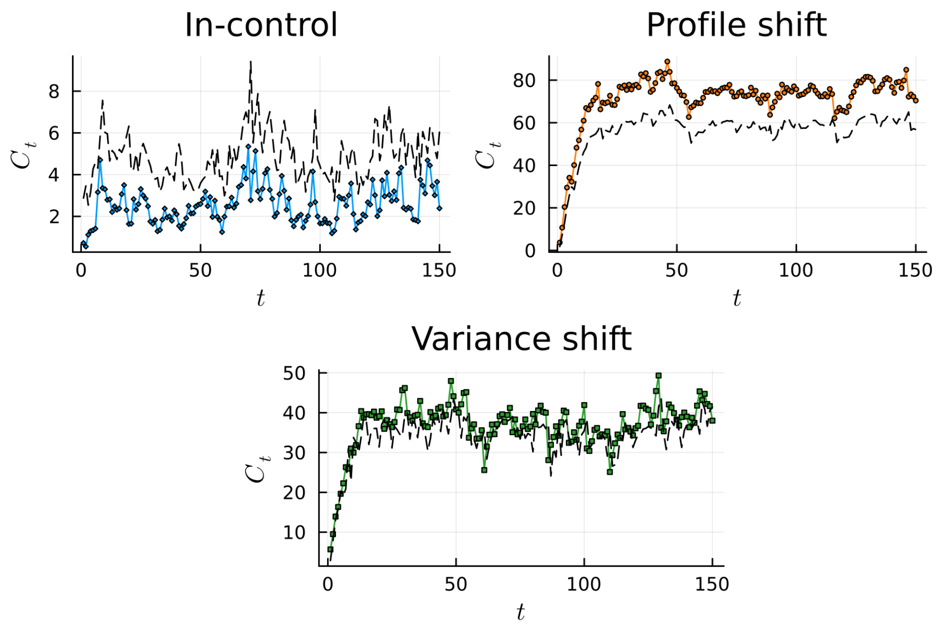

Figure 6: NEWMA control chart applied to the Phase II profiles. The dashed line in each plot indicates the estimated control limit such that $\text{ARL}_{\text{IC}} = 500$ and the conditional false alarm rate is constant.

```
julia> CH = ControlChart(STAT, LIM, NM, PH2)
```

The NEWMA control chart is then applied to three sequences of Phase II data of length 150. The three sequences are sampled from each of the data-generating processes listed in Table 1.

```
julia> n2 = 150
julia> xs_oc = xs[1:n2, :]
julia> ys_ic = sin.(xs_oc) .+ randn(n2, nj)
julia> ys_oc = sin.(xs_oc) + 2*cos.(xs_oc) .+ randn(n2, nj)
julia> ys_oc2 = sin.(xs_oc) .+ 2*randn(n2, nj)
```

The matrices containing the Phase II observations are then converted to `FunctionalData` objects to apply the control chart.

```
julia> dat_ic = FunctionalData(xs_oc, ys_ic)
julia> dat_oc = FunctionalData(xs_oc, ys_oc)
julia> dat_oc2 = FunctionalData(xs_oc, ys_oc2)
```

Then, the control chart is run on the three data sequences.

```
julia> proc_ic = apply_chart(CH, dat_ic)
julia> proc_oc = apply_chart(CH, dat_oc)
julia> proc_oc2 = apply_chart(CH, dat_oc2)
```

The results of applying the NEWMA control chart to the Phase II sequences are displayed in Figure 6. The plots show the values of the NEWMA control chart in the three cases, with the estimated control limits as a black dashed line. The control chart correctly detects the change in both the profile shift and variance shift. When the process is IC, no change is detected by the control chart.

5. Conclusion

StatisticalProcessMonitoring.jl is a Julia package which provides a general framework for statistical process monitoring. In addition to implementing various control charts, the main features of the package include algorithms for automatic estimation of the control limits, and optimization of the tuning parameters of a control chart. The package interface allows users to define a wide range of control charts for univariate, multivariate, and structured data types.

The package is available on the JuliaHub General registry (`https://juliahub.com/ui/Packages/General/StatisticalProcessMonitoring`) and can be installed from within Julia by executing the following code.

```
julia> using Pkg
julia> Pkg.add("StatisticalProcessMonitoring")
julia> using StatisticalProcessMonitoring
```

The package is released under the GNU GPL license version 3 and is undergoing active development, which can be followed at the Github page `https://github.com/DedZago/StatisticalProcessMonitoring.jl`. Future developments for the package include:

- Supporting the adjustment of control limits for parameter estimation using resampling methods such as the AR-sieve bootstrap (Capizzi and Masarotto 2009) and the guaranteed in-control performance methodology (Gandy and Kvaløy 2013).
- Implementing additional Phase II monitoring statistics, as well as monitoring statistics based on the change-point model (Hawkins *et al.* 2003).
- Implementing parametric and nonparametric Phase I control charts.

Computational details

The results in this paper were obtained using Julia 1.10.0 running on Ubuntu 22.04.1. All packages and dependencies are listed alongside their version in the `Manifest.toml` file, which is found in the replication material. Julia itself can be downloaded from `https://julialang.org/downloads/`, whereas all packages used are available from the Julia ecosystem and can be installed using the built-in package manager `Pkg`. The exact version of the packages used in the above simulations in the paper are listed in the `Manifest.toml` file. To access the built-in package manager in interactive mode, the `]` key can be pressed in the Julia REPL.

References

Alt FA (1984). "Multivariate Quality Control." In NL Johnson, S Kotz, CR Read (eds.), *The Encyclopedia of Statistical Sciences*, volume 6, pp. 110–122. John Wiley & Sons, New York.

Anhoej J (2021a). "**qicharts**: Quality Improvement Charts." `doi:10.32614/CRAN.package.qicharts`. R package version 0.5.8.

Anhoej J (2021b). "**qicharts2**: Quality Improvement Charts." `doi:10.32614/CRAN.package.qicharts2`. R package version 0.7.5.

Apley DW, Lee HC (2008). "Robustness Comparison of Exponentially Weighted Moving-Average Charts on Autocorrelated Data and on Residuals." *Journal of Quality Technology*, **40**(4), 428–447. `doi:10.1080/00224065.2008.11917747`.

Bates D, Alday P, Kleinschmidt D, Calderón JBS, Zhan L, Noack A, Bouchet-Valat M, Arslan A, Kelman T, Baldassari A, Ehinger B, Karrasch D, Saba E, Quinn J, Hatherly M, Piibeleht M, Mogensen PK, Babayan S, Holy T, Gagnon YL, Nazarathy Y (2023). "JuliaStats/**MixedModels.jl**: V4.22.3." `doi:10.5281/zenodo.10268806`.

Besançon M, Papamarkou T, Anthoff D, Arslan A, Byrne S, Lin D, Pearson J (2021). "**Distributions.jl**: Definition and Modeling of Probability Distributions in the JuliaStats Ecosystem." *Journal of Statistical Software*, **98**, 1–30. `doi:10.18637/jss.v098.i16`.

Bezanson J, Edelman A, Karpinski S, Shah VB (2017). "Julia: A Fresh Approach to Numerical Computing." *SIAM Review*, **59**(1), 65–98. `doi:10.1137/141000671`.

Bühlmann P (2002). "Bootstraps for Time Series." *Statistical Science*, **17**(1), 52–72. `doi:10.1214/ss/1023798998`.

Capezza C, Centofanti F, Lepore A, Menafoglio A, Palumbo B, Vantini S (2023a). "**funcharts**: Control Charts for Multivariate Functional Data in R." *Journal of Quality Technology*, **55**(5), 566–583. `doi:10.1080/00224065.2023.2219012`.

Capezza C, Centofanti F, Lepore A, Menafoglio A, Palumbo B, Vantini S (2023b). ***funcharts**: Functional Control Charts*. `doi:10.32614/CRAN.package.funcharts`. R package version 1.6.0.

Capezza C, Lepore A, Menafoglio A, Palumbo B, Vantini S (2020). "Control Charts for Monitoring Ship Operating Conditions and CO2 Emissions Based on Scalar-on-Function Regression." *Applied Stochastic Models in Business and Industry*, **36**(3), 477–500. `doi:10.1002/asmb.2507`.

Capizzi G, Masarotto G (2003). "An Adaptive Exponentially Weighted Moving Average Control Chart." *Technometrics*, **45**(3), 199–207. `doi:10.1198/004017003000000023`.

Capizzi G, Masarotto G (2008). "Practical Design of Generalized Likelihood Ratio Control Charts for Autocorrelated Data." *Technometrics*, **50**(3), 357–370. doi:10.1198/004017008000000280.

Capizzi G, Masarotto G (2009). "Bootstrap-Based Design of Residual Control Charts." *IIE Transactions*, **41**(4), 275–286. doi:10.1080/07408170802120059.

Capizzi G, Masarotto G (2012). "An Enhanced Control Chart for Start-Up Processes and Short Runs." *Quality Technology & Quantitative Management*, **9**(2), 189–202. doi:10.1080/16843703.2012.11673285.

Capizzi G, Masarotto G (2016). "Efficient Control Chart Calibration by Simulated Stochastic Approximation." *IIE Transactions*, **48**(1), 57–65. doi:10.1080/0740817x.2015.1055392.

Capizzi G, Masarotto G (2018). "Phase I Distribution-Free Analysis with the R Package **dfphase1**." In S Knoth, W Schmid (eds.), *Frontiers in Statistical Quality Control 12*, pp. 3–19. Springer-Verlag. doi:10.1007/978-3-319-75295-2_1.

Centofanti F, Lepore A, Menafoglio A, Palumbo B, Vantini S (2021). "Functional Regression Control Chart." *Technometrics*, **63**(3), 281–294. doi:10.1080/00401706.2020.1753581.

Chakraborti S, Graham M (2019). "Nonparametric (Distribution-Free) Control Charts: An Updated Overview and Some Results." *Quality Engineering*, **31**(4), 523–544. doi:10.1080/08982112.2018.1549330.

Chakraborti S, Van Der Laan P, Bakir ST (2001). "Nonparametric Control Charts: An Overview and Some Results." *Journal of Quality Technology*, **33**(3), 304–315. doi:10.1080/00224065.2001.11980081.

Champ CW, Woodall WH, Mohsen HA (1991). "A Generalized Quality Control Procedure." *Statistics & Probability Letters*, **11**(3), 211–218. doi:10.1016/0167-7152(91)90145-h.

Chatterjee S, Qiu P (2009). "Distribution-Free Cumulative Sum Control Charts Using Bootstrap-Based Control Limits." *The Annals of Applied Statistics*, **3**(1), 349–369. doi:10.1214/08-aoas197.

Chen HF (2002). *Stochastic Approximation and Its Applications*. Springer-Verlag, New York.

Cleveland WS (1979). "Robust Locally Weighted Regression and Smoothing Scatterplots." *Journal of the American Statistical Association*, **74**(368), 829–836. doi:10.1080/01621459.1979.10481038.

Colosimo BM, Grasso M (2018). "Spatially Weighted PCA for Monitoring Video Image Data with Application to Additive Manufacturing." *Journal of Quality Technology*, **50**(4), 391–417. doi:10.1080/00224065.2018.1507563.

Colosimo BM, Semeraro Q, Pacella M (2008). "Statistical Process Control for Geometric Specifications: On the Monitoring of Roundness Profiles." *Journal of Quality Technology*, **40**(1), 1–18. doi:10.1080/00224065.2008.11917709.

Crosier RB (1988). "Multivariate Generalizations of Cumulative Sum Quality-Control Schemes." *Technometrics*, **30**(3), 291–303. doi:10.2307/1270083.

Dai Y, Luo Y, Li Z, Wang Z (2011). "A New Adaptive CUSUM Control Chart for Detecting the Multivariate Process Mean." *Quality and Reliability Engineering International*, **27**(7), 877–884. doi:10.1002/qre.1177.

Duncan AJ (1956). "The Economic Design of X Charts Used to Maintain Current Control of a Process." *Journal of the American Statistical Association*, **51**(274), 228–242. doi:10.1080/01621459.1956.10501322.

Efron B, Tibshirani RJ (1993). *An Introduction to the Bootstrap*. Chapman and Hall/CRC, New York.

Ellis TMR, Philips IR, Lahey TM (1994). *Fortran 90 Programming*. Addison-Wesley, Wokingham.

English JR, Lee SC, Martin TW, Tilmon C (2000). "Detecting Changes in Autoregressive Processes with X¯and EWMA Charts." *IIE Transactions*, **32**(12), 1103–1113. doi:10.1080/07408170008967465.

Erdman C, Emerson JW (2008). "**bcp**: An R Package for Performing a Bayesian Analysis of Change Point Problems." *Journal of Statistical Software*, **23**, 1–13. doi:10.18637/jss.v023.i03.

Flach M, Gans F, Brenning A, Denzler J, Reichstein M, Rodner E, Bathiany S, Bodesheim P, Guanche Y, Sippel S, Mahecha MD (2017). "Multivariate Anomaly Detection for Earth Observations: A Comparison of Algorithms and Feature Extraction Techniques." *Earth System Dynamics*, **8**(3), 677–696. doi:10.5194/esd-8-677-2017.

Gan FF (1993). "An Optimal Design of EWMA Control Charts Based on Median Run Length." *Journal of Statistical Computation and Simulation*, **45**(3-4), 169–184. doi:10.1080/00949659308811479.

Gan FF (1994). "An Optimal Design of Cumulative Sum Control Chart Based on Median Run Length." *Communications in Statistics – Simulation and Computation*, **23**(2), 485–503. doi:10.1080/03610919408813183.

Gan FF (1995). "Joint Monitoring of Process Mean and Variance Using Exponentially Weighted Moving Average Control Charts." *Technometrics*, **37**(4), 446–453. doi:10.2307/1269736.

Gandy A, Kvaløy JT (2013). "Guaranteed Conditional Performance of Control Charts via Bootstrap Methods." *Scandinavian Journal of Statistics*, **40**(4), 647–668. doi:10.1002/sjos.12006.

Gandy A, Kvaløy JT (2017). "**spcadjust**: Functions for Calibrating Control Charts." *The R Journal*, **9**(1), 458–476. doi:10.32614/rj-2017-014.

Graham MA, Mukherjee A, Chakraborti S (2017). "Design and Implementation Issues for a Class of Distribution-Free Phase II EWMA Exceedance Control Charts." *International Journal of Production Research*, **55**(8), 2397–2430. doi:10.1080/00207543.2016.1249428.

Grasso M, Laguzza V, Semeraro Q, Colosimo BM (2016). "In-Process Monitoring of Selective Laser Melting: Spatial Detection of Defects Via Image Data Analysis." *Journal of Manufacturing Science and Engineering*, **139**(5). doi:10.1115/1.4034715.

Grigg O, Farewell V (2004). "An Overview of Risk-Adjusted Charts." *Journal of the Royal Statistical Society A*, **167**(3), 523–539. doi:10.1111/j.1467-985x.2004.0apm2.x.

Hawkins DM (1987). "Self-Starting Cusum Charts for Location and Scale." *Journal of the Royal Statistical Society D*, **36**(4), 299–316. doi:10.2307/2348827.

Hawkins DM, Deng Q (2009). "Combined Charts for Mean and Variance Information." *Journal of Quality Technology*, **41**(4), 415–425. doi:10.1080/00224065.2009.11917795.

Hawkins DM, Maboudou-Tchao EM (2008). "Multivariate Exponentially Weighted Moving Covariance Matrix." *Technometrics*, **50**(2), 155–166. doi:10.1198/004017008000000163.

Hawkins DM, Qiu P, Kang CW (2003). "The Changepoint Model for Statistical Process Control." *Journal of Quality Technology*, **35**(4), 355–366. doi:10.1080/00224065.2003.11980233.

Hu X, Castagliola P, Tang A, Zhong J (2021). "Guaranteed Conditional Performance of the Median Run Length Based EWMA X¯ Chart with Unknown Process Parameters." *Communications in Statistics - Simulation and Computation*, **50**(12), 4280–4299. doi:10.1080/03610918.2019.1642485.

Huang Q, Zhou S, Shi J (2002). "Diagnosis of Multi-Operational Machining Processes Through Variation Propagation Analysis." *Robotics and Computer-Integrated Manufacturing*, **18**(3), 233–239. doi:10.1016/s0736-5845(02)00014-5.

Huber PJ (1981). *Robust Statistics*. John Wiley & Sons. doi:10.1002/0471725250.

Huwang L, Yeh AB, Wu CW (2007). "Monitoring Multivariate Process Variability for Individual Observations." *Journal of Quality Technology*, **39**(3), 258–278. doi:10.1080/00224065.2007.11917692.

Jin J, Shi J (1999). "State Space Modeling of Sheet Metal Assembly for Dimensional Control." *Journal of Manufacturing Science and Engineering*, **121**(4), 756–762. doi:10.1115/1.2833137.

Johnson SG (2007). "The **NLopt** Nonlinear-Optimization Package." URL https://github.com/stevengj/nlopt.

Juran JM (1951). "The Economics of Quality." In JM Juran (ed.), *Quality Control Handbook*, pp. 1–41. McGraw-Hill, New Tork.

Kang L, Albin SL (2000). "On-Line Monitoring When the Process Yields a Linear Profile." *Journal of Quality Technology*, **32**(4), 418–426. doi:10.1080/00224065.2000.11980027.

Khoo MBC (2003). "Design of Runs Rules Schemes." *Quality Engineering*, **16**(1), 27–43. doi:10.1081/qen-120020769.

Killick R, Eckley IA (2014). "**changepoint**: An R Package for Changepoint Analysis." *Journal of Statistical Software*, **58**, 1–19. doi:10.18637/jss.v058.i03.

Killick R, Haynes K, Eckley IA (2022). ***changepoint**: An R Package for Changepoint Analysis.* doi:10.32614/CRAN.package.changepoint. R package version 2.3.

Kim K, Mahmoud MA, Woodall WH (2003). "On the Monitoring of Linear Profiles." *Journal of Quality Technology*, **35**(3), 317–328. doi:10.1080/00224065.2003.11980225.

Knoth S (2015). "Run Length Quantiles of EWMA Control Charts Monitoring Normal Mean or/and Variance." *International Journal of Production Research*, **53**(15), 4629–4647. doi:10.1080/00207543.2015.1005253.

Knoth S (2017). "ARL Numerics for MEWMA Charts." *Journal of Quality Technology*, **49**(1), 78–89. doi:10.1080/00224065.2017.11918186.

Knoth S (2024). ***spc**: Statistical Process Control – Calculation of ARL and Other Control Chart Performance Measures.* doi:10.32614/CRAN.package.spc. R package version 0.7.1.

Kushner H, Yin GG (2003). *Stochastic Approximation and Recursive Algorithms and Applications.* 2nd edition. Springer-Verlag, New York.

Lee HC, Apley DW (2011). "Improved Design of Robust Exponentially Weighted Moving Average Control Charts for Autocorrelated Processes." *Quality and Reliability Engineering International*, **27**(3), 337–352. doi:10.1002/qre.1126.

Li J, Tsung F, Zou C (2012). "Directional Control Schemes for Multivariate Categorical Processes." *Journal of Quality Technology*, **44**(2), 136–154. doi:10.1080/00224065.2012.11917889.

Li W, Pu X, Tsung F, Xiang D (2017). "A Robust Self-Starting Spatial Rank Multivariate EWMA Chart Based on Forward Variable Selection." *Computers and Industrial Engineering*, **103**(C), 116–130. doi:10.1016/j.cie.2016.11.024.

Lin D, White JM, Byrne S, Bates D, Noack A, Pearson J, Arslan A, Squire K, Anthoff D, Papamarkou T, Besançon M, Drugowitsch J, Schauer M, other contributors (2019). "JuliaStats/**Distributions.jl**: A Julia Package for Probability Distributions and Associated Functions." doi:10.5281/zenodo.2647458.

Liu K, Mei Y, Shi J (2015). "An Adaptive Sampling Strategy for Online High-Dimensional Process Monitoring." *Technometrics*, **57**(3), 305–319. doi:10.1080/00401706.2014.947005.

Liu P, Du J, Zang Y, Zhang C, Wang K (2023). "In-Profile Monitoring for Cluster-Correlated Data in Advanced Manufacturing System." *Journal of Quality Technology*, **55**(2), 195–219. doi:10.1080/00224065.2022.2106912.

Lorden G, Eisenberger I (1973). "Detection of Failure Rate Increases." *Technometrics*, **15**(1), 167–175. doi:10.2307/1266833.

Lorenzen TJ, Vance LC (1986). "The Economic Design of Control Charts: A Unified Approach." *Technometrics*, **28**(1), 3–10. doi:10.2307/1269598.

Lowry CA, Woodall WH, Champ CW, Rigdon SE (1992). "A Multivariate Exponentially Weighted Moving Average Control Chart." *Technometrics*, **34**(1), 46–53. doi:10.2307/1269551.

Lu CW, Reynolds MR (1999). "EWMA Control Charts for Monitoring the Mean of Autocorrelated Processes." *Journal of Quality Technology*, **31**(2), 166–188. doi:10.1080/00224065.1999.11979913.

Lucas JM (1982). "Combined Shewhart-CUSUM Quality Control Schemes." *Journal of Quality Technology*, **14**(2), 51–59. doi:10.1080/00224065.1982.11978790.

Lucas JM, Saccucci MS (1990). "Exponentially Weighted Moving Average Control Schemes: Properties and Enhancements." *Technometrics*, **32**(1), 1–12. doi:10.2307/1269835.

Mahmoud MA, Zahran AR (2010). "A Multivariate Adaptive Exponentially Weighted Moving Average Control Chart." *Communications in Statistics – Theory and Methods*, **39**(4), 606–625. doi:10.1080/03610920902755813.

Meyer S, Held L, Höhle M (2017). "Spatio-Temporal Analysis of Epidemic Phenomena Using the R Package **surveillance**." *Journal of Statistical Software*, **77**(11), 1–55. doi:10.18637/jss.v077.i11.

Noack A, Jones DC, Arslan A, Fields E (2013). "JuliaStats/**Loess.jl**: V0.6.3." URL https://github.com/JuliaStats/Loess.jl.

Page ES (1954). "Continuous Inspection Schemes." *Biometrika*, **41**(1/2), 100. doi:10.2307/2333009.

Politis DN, Romano JP (1992). "A Circular Block-Resampling Procedure for Stationary Data." In R LePage, L Billard (eds.), *Exploring the Limits of Bootstrap*, pp. 263–270. John Wiley & Sons, New York.

Politis DN, Romano JP (1994). "The Stationary Bootstrap." *Journal of the American Statistical Association*, **89**(428), 1303–1313. doi:10.2307/2290993.

Polyak B, Juditsky A (1992). "Acceleration of Stochastic Approximation by Averaging." *SIAM Journal on Control and Optimization*, **20**(4), 838–855. doi:10.1137/0330046.

Powell M (2009). "The BOBYQA Algorithm for Bound Constrained Optimization without Derivatives." *Technical Report DAMTP 2009/NA06*, Department of Applied Mathematics and Theoretical Physics, Centre for Mathematical Sciences, University of Cambridge.

Prajapati DR, Singh S (2012). "Control Charts for Monitoring the Autocorrelated Process Parameters: A Literature Review." *International Journal of Productivity and Quality Management*, **10**(2), 207. doi:10.1504/ijpqm.2012.048298.

Qiao Y, Sun J, Castagliola P, Hu X (2022). "Optimal Design of One-Sided Exponential EWMA Charts Based on Median Run Length and Expected Median Run Length." *Communications in Statistics – Theory and Methods*, **51**(9), 2887–2907. doi:10.1080/03610926.2020.1782937.

Qiu P (2008). "Distribution-Free Multivariate Process Control Based on Log-Linear Modeling." *IIE Transactions*, **40**(7), 664–677. doi:10.1080/07408170701744843.

Qiu P (2013). *Introduction to Statistical Process Control*. CRC Press, Boca Raton.

Qiu P, Hawkins D (2001). "A Rank-Based Multivariate CUSUM Procedure." *Technometrics*, **43**(2), 120–132. doi:10.1198/004017001750386242.

Qiu P, Xie X (2021). "Transparent Sequential Learning for Statistical Process Control of Serially Correlated Data." *Technometrics*, **64**(4), 487–501. doi:10.1080/00401706.2021.1929493.

Qiu P, You L (2022). "Dynamic Disease Screening by Joint Modelling of Survival and Longitudinal Data." *Journal of the Royal Statistical Society C*, **71**(5), 1158–1180. doi:10.1111/rssc.12573.

Qiu P, Zi X, Zou C (2018). "Nonparametric Dynamic Curve Monitoring." *Technometrics*, **60**(3), 386–397. doi:10.1080/00401706.2017.1361340.

Ranger GC, Alt FB (1996). "Choosing Principal Components for Multivariate Statistical Process Control." *Communications in Statistics – Theory and Methods*, **25**(5), 909–922. doi:10.1080/03610929608831739.

R Core Team (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. doi:10.32614/R.manuals. URL https://www.R-project.org/.

Reynolds MR, Stoumbos ZG (2008). "Combinations of Multivariate Shewhart and MEWMA Control Charts for Monitoring the Mean Vector and Covariance Matrix." *Journal of Quality Technology*, **40**(4), 381–393. doi:10.1080/00224065.2008.11917744.

Riaz M, Zaman B, Mehmood R, Abbas N, Abujiya M (2021). "Advanced Multivariate Cumulative Sum Control Charts Based on Principal Component Method with Application." *Quality and Reliability Engineering International*, **37**(6), 2760–2789. doi:10.1002/qre.2889.

Rigdon SE (1995a). "A Double-Integral Equation for the Average Run Length of a Multivariate Exponentially Weighted Moving Average Control Chart." *Statistics & Probability Letters*, **24**(4), 365–373. doi:10.1016/0167-7152(94)00196-f.

Rigdon SE (1995b). "An Integral Equation for the In-Control Average Run Length of a Multivariate Exponentially Weighted Moving Average Control Chart." *Journal of Statistical Computation and Simulation*, **52**(4), 351–365. doi:10.1080/00949659508811685.

Roberts SW (1959). "Control Chart Tests Based on Geometric Moving Averages." *Technometrics*, **1**(3), 239–250. doi:10.1080/00401706.1959.10489860.

Roberts SW (1966). "A Comparison of Some Control Chart Procedures." *Technometrics*, **8**(3), 411–430. doi:10.1080/00401706.1966.10490374.

Ross GJ (2015). "Parametric and Nonparametric Sequential Change Detection in R: The **cpm** Package." *Journal of Statistical Software*, **66**(3), 1–20. doi:10.18637/jss.v066.i03.

Ruppert D (1991). "Stochastic Approximation." In BK Ghosh, PK Sen (eds.), *Handbook of Sequential Analysis*, pp. 503–529. Marcel Dekker, New York.

Salmon M, Schumacher D, Höhle M (2016). "Monitoring Count Time Series in R: Aberration Detection in Public Health Surveillance." *Journal of Statistical Software*, **70**, 1–35. doi:10.18637/jss.v070.i10.

Santos-Fernández E (2012). *Multivariate Statistical Quality Control Using R*, volume 14. Springer-Verlag. doi:10.1007/978-1-4614-5453-3.

Savitzky A, Golay MJE (1964). "Smoothing and Differentiation of Data by Simplified Least Squares Procedures." *Analytical Chemistry*, **36**(8), 1627–1639. doi:10.1021/ac60214a047.

Scimone R, Taormina T, Colosimo BM, Grasso M, Menafoglio A, Secchi P (2022). "Statistical Modeling and Monitoring of Geometrical Deviations in Complex Shapes with Application to Additive Manufacturing." *Technometrics*, **64**(4), 437–456. doi:10.1080/00401706.2021.1961870.

Scranton R, Runger GC, Keats JB, Montgomery DC (1996). "Efficient Shift Detection Using Multivariate Exponentially-Weighted Moving Average Control Charts and Principal Components." *Quality and Reliability Engineering International*, **12**(3), 165–171. doi:10.1002/(sici)1099-1638(199605)12:3<165::aid-qre990>3.0.co;2-q.

Scrucca L, Snow G, Bloomfield P (2004). "**qcc**: An R Package for Quality Control Charting and Statistical Process Control." *R News*, **4/1**, 11–17.

Shen X, Tsui KL, Zou C, Woodall WH (2016). "Self-Starting Monitoring Scheme for Poisson Count Data with Varying Population Sizes." *Technometrics*, **58**(4), 460–471. doi:10.1080/00401706.2015.1075423.

Shen X, Zou C, Jiang W, Tsung F (2013). "Monitoring Poisson Count Data with Probability Control Limits When Sample Sizes Are Time Varying." *Naval Research Logistics*, **60**(8), 625–636. doi:10.1002/nav.21557.

Shewhart WA (1931). *Economic Control of Quality Of Manufactured Product*. D. Van Nostrand Company, New York.

Shiryaev AN (1961). "The Problem of the Most Rapid Detection of a Disturbance in a Stationary Process." In *Soviet Mathematics Doklady*, volume 2, p. 103.

Shiryaev AN (1963). "On Optimum Methods in Quickest Detection Problems." *Theory of Probability & Its Applications*, **8**(1), 22–46. doi:10.1137/1108002.

Shu L, Jiang W, Tsui KL (2008). "A Weighted CUSUM Chart for Detecting Patterned Mean Shifts." *Journal of Quality Technology*, **40**(2), 194–213. doi:10.1080/00224065.2008.11917725.

Shumway RH, Stoffer DS (2017). *Time Series Analysis and Its Applications: With R Examples*. 4th edition. Springer-Verlag, New York.

Silva C (2023). "**PySpc**: Statistical Process Control Charts Library for Humans." URL https://github.com/carlosqsilva/pyspc.

Sparks RS (2000). "CUSUM Charts for Signalling Varying Location Shifts." *Journal of Quality Technology*, **32**(2), 157–171. doi:10.1080/00224065.2000.11979987.

Steiner SH (1999). "EWMA Control Charts with Time-Varying Control Limits and Fast Initial Response." *Journal of Quality Technology*, **31**(1), 75–86. doi:10.1080/00224065.1999.11979899.

Steiner SH, Cook RJ, Farewell VT, Treasure T (2000). "Monitoring Surgical Performance Using Risk-Adjusted Cumulative Sum Charts." *Biostatistics*, **1**(4), 441–452. doi:10.1093/biostatistics/1.4.441.

Stroustrup B (2013). *The C++ Programming Language*. 4th edition. Addison-Wesley Professional, Westport.

The Mathworks Inc (2023). "MATLAB Version: 23.2.0 (R2023b)." The MathWorks Inc.

Tratt L (2009). "Dynamically Typed Languages." In *Advances in Computers*, volume 77, pp. 149–184. Elsevier. doi:10.1016/s0065-2458(09)01205-4.

Van Rossum G, Drake Jr FL (1995). *Python Reference Manual*. Centrum voor Wiskunde en Informatica Amsterdam.

Waldmann KH (1986a). "Bounds for the Distribution of the Run Length of Geometric Moving Average Charts." *Journal of the Royal Statistical Society C*, **35**(2), 151–158. doi:10.2307/2347265.

Waldmann KH (1986b). "Bounds for the Distribution of the Run Length of One-Sided and Two-Sided CUSUM Quality Control Schemes." *Technometrics*, **28**(1), 61–67. doi:10.2307/1269604.

Wang J, Li J, Su Q (2017). "Multivariate Ordinal Categorical Process Control Based on Log-Linear Modeling." *Journal of Quality Technology*, **49**(2), 108–122. doi:10.1080/00224065.2017.11917983.

Wiel SAV (1996). "Monitoring Processes That Wander Using Integrated Moving Average Models." *Technometrics*, **38**(2), 139–151. doi:10.2307/1270407.

Woodall WH (2000). "Controversies and Contradictions in Statistical Process Control." *Journal of Quality Technology*, **32**(4), 341–350. doi:10.1080/00224065.2000.11980013.

Woodall WH (2006). "The Use of Control Charts in Health-Care and Public-Health Surveillance." *Journal of Quality Technology*, **38**(2), 89–104. doi:10.1080/00224065.2006.11918593.

Xian X, Zhang C, Bonk S, Liu K (2019). "Online Monitoring of Big Data Streams: A Rank-Based Sampling Algorithm by Data Augmentation." *Journal of Quality Technology*, **53**(2), 135–153. doi:10.1080/00224065.2019.1681924.

Xie X, Qiu P (2022). "Robust Monitoring of Multivariate Processes with Short-Ranged Serial Data Correlation." *Quality and Reliability Engineering International*, **38**(8), 4196–4209. doi:10.1002/qre.3199.

Xie X, Qiu P (2023). "A General Framework for Robust Monitoring of Multivariate Correlated Processes." *Technometrics*, pp. 1–22. doi:10.1080/00401706.2023.2224411.

Xue L, Qiu P (2021). "A Nonparametric CUSUM Chart for Monitoring Multivariate Serially Correlated Processes." *Journal of Quality Technology*, **53**(4), 396–409. doi:10.1080/00224065.2020.1778430.

Yan H, Grasso M, Paynabar K, Colosimo BM (2022). "Real-Time Detection of Clustered Events in Video-Imaging Data with Applications to Additive Manufacturing." *IISE Transactions*, **54**(5), 464–480. doi:10.1080/24725854.2021.1882013.

Yan H, Paynabar K, Shi J (2014). "Image-Based Process Monitoring Using Low-Rank Tensor Decomposition." *IEEE Transactions on Automation Science and Engineering*, **12**(1), 216–227. doi:10.1109/tase.2014.2327029.

Yang K, Qiu P (2020). "Online Sequential Monitoring of Spatio-Temporal Disease Incidence Rates." *IISE Transactions*, **52**(11), 1218–1233. doi:10.1080/24725854.2019.1696496.

Yang W, Zou C, Wang Z (2017). "Nonparametric Profile Monitoring Using Dynamic Probability Control Limits." *Quality and Reliability Engineering International*, **33**(5), 1131–1142. doi:10.1002/qre.2104.

Yi F, Qiu P (2021). "An Adaptive CUSUM Chart for Drift Detection." *Quality and Reliability Engineering International*, **38**(2), 887–894. doi:10.1002/qre.3020.

Yi G, Herdsman C, Morris J (2019). "A MATLAB Toolbox for Data Pre-Processing and Multivariate Statistical Process Control." *Chemometrics and Intelligent Laboratory Systems*, **194**, 103863. doi:10.1016/j.chemolab.2019.103863.

You L, Qiu P (2019). "Fast Computing for Dynamic Screening Systems When Analyzing Correlated Data." *Journal of Statistical Computation and Simulation*, **89**(3), 379–394. doi:10.1080/00949655.2018.1552273.

Zang Y, Qiu P (2018a). "Phase I Monitoring of Spatial Surface Data from 3D Printing." *Technometrics*, **60**(2), 169–180. doi:10.1080/00401706.2017.1321585.

Zang Y, Qiu P (2018b). "Phase II Monitoring of Free-Form Surfaces: An Application to 3D Printing." *Journal of Quality Technology*, **50**(4), 379–390. doi:10.1080/00224065.2018.1508274.

Zeileis A, Leisch F, Hornik K, Kleiber C (2002). "**strucchange**: An R Package for Testing for Structural Change in Linear Regression Models." *Journal of Statistical Software*, **7**, 1–38. doi:10.18637/jss.v007.i02.

Zhang C, Yan H, Lee S, Shi J (2018). "Multiple Profiles Sensor-Based Monitoring and Anomaly Detection." *Journal of Quality Technology*, **50**(4), 344–362. doi:10.1080/00224065.2018.1508275.

Zhao X, del Castillo E (2021). "An Intrinsic Geometrical Approach for Statistical Process Control of Surface and Manifold Data." *Technometrics*, **63**(3), 295–312. doi:10.1080/00401706.2020.1772114.

Zhao Y, Tsung F, Wang Z (2005). "Dual CUSUM Control Schemes for Detecting a Range of Mean Shifts." *IIE Transactions*, **37**(11), 1047–1057. doi:10.1080/07408170500232321.

Zhou S, Huang Q, Shi J (2003). "State Space Modeling of Dimensional Variation Propagation in Multistage Machining Process Using Differential Motion Vectors." *IEEE Transactions on Robotics and Automation*, **19**(2), 296–309. doi:10.1109/tra.2003.808852.

Zhu W, Park C (2013). "**edcc**: An R Package for the Economic Design of the Control Chart." *Journal of Statistical Software*, **52**, 1–24. doi:10.18637/jss.v052.i09.

Zou C, Tsung F (2010). "Likelihood Ratio-Based Distribution-Free EWMA Control Charts." *Journal of Quality Technology*, **42**(2), 174–196. doi:10.1080/00224065.2010.11917815.

Zou C, Tsung F (2011). "A Multivariate Sign EWMA Control Chart." *Technometrics*, **53**(1), 84–97. doi:10.1198/tech.2010.09095.

Zou C, Tsung F, Wang Z (2008). "Monitoring Profiles Based on Nonparametric Regression Methods." *Technometrics*, **50**(4), 512–526. doi:10.1198/004017008000000433.

A. Available methods

Monitoring statistic	Type name	Hyperparameters	Reference paper
Mean (univariate)			
Shewhart	`Shewhart`	–	Shewhart (1931)
CUSUM	`CUSUM`	$k \in \mathbb{R}^+$	Page (1954)
EWMA	`EWMA`	$\lambda \in (0,1)$	Roberts (1959)
One-sided EWMA	`OneSidedEWMA`	$\lambda \in (0,1)$	Champ, Woodall, and Mohsen (1991)
Adaptive EWMA	`AEWMA`	$\lambda \in (0,1), k \in \mathbb{R}^+$	Capizzi and Masarotto (2003)
Weighted CUSUM	`WCUSUM`	$\lambda \in (0,1)$, $k \in \mathbb{R}^+$	Shu, Jiang, and Tsui (2008)
Mean (multivariate)			
MShewhart	`MShewhart`	–	Shewhart (1931)
MEWMA	`DiagMEWMA`	$\boldsymbol{\lambda} \in (0,1)^p$	Lowry *et al.* (1992)
MCUSUM	`MCUSUM`	$k \in \mathbb{R}^+$	Crosier (1988)
Adaptive MEWMA	`MAEWMA`	$\lambda \in (0,1)$, $k \in \mathbb{R}^+$	Mahmoud and Zahran (2010)
Adaptive MCUSUM	`AMCUSUM`	$\lambda \in (0,1)$	Dai, Luo, Li, and Wang (2011)
LLCUSUM	`LLCUSUM`	$k \in \mathbb{R}^+$	Qiu (2008)
LLD	`LLD`	$\lambda \in (0,1)$	Li *et al.* (2012)
MOC	`MOC`	$\lambda \in (0,1)$	Wang *et al.* (2017)
Variance/covariance			
GLR-based statistic	`ALT`	–	Alt (1984)
MEWMS	`MEWMS`	$\lambda \in (0,1)$	Huwang, Yeh, and Wu (2007)
MEWMC	`MEWMC`	$\lambda \in (0,1)$	Hawkins and Maboudou-Tchao (2008)
Partially-observed data			
R-SADA	`RSADA`	$k \in \mathbb{R}^+$, $\mu_{\min} \in \mathbb{R}^+$	Xian, Zhang, Bonk, and Liu (2019)
TRAS	`TRAS`	$k \in \mathbb{R}^+$, $\mu_{\min} \in \mathbb{R}^+$, $\Delta \in \mathbb{R}^+$, $r \in \{1, \ldots, q\}$	Liu, Mei, and Shi (2015)
Profile monitoring			
NEWMA	`NEWMA`	$\lambda \in (0,1)$	Zou *et al.* (2008)
Risk-adjusted CUSUM	`RiskAdjustedCUSUM`	$\Delta \in \mathbb{R}$	Steiner *et al.* (2000)

Table 2: List of available monitoring statistics in the **StatisticalProcessMonitoring.jl** package. Here, p is the number of quality variables under monitoring. For partially-observed data, q is the number of observed variables.

Functions	Symbol	Description	Single charts	Multi-charts
`bisectionCL`, `bisectionCL!`	`:Bisection`	The standard bisection algorithm (see Qiu 2013). Requires an initial interval to search for the control limit.	✓	
`saCL`, `saCL!`	`:SA`	Algorithm based on stochastic approximations. See Capizzi and Masarotto (2016) for a complete description of the tuning parameters.	✓	✓
`combinedCL`, `combinedCL!`	`:Combined`	The standard bisection algorithm combined with a preliminary low-accuracy application of the SA algorithm to automatically select the initial interval.	✓	
`bootstrapCL`, `bootstrapCL!`	`:Bootstrap`	Approximates the distribution of the monitoring statistic at each time $t = 1, 2, \ldots, T$ for $T > 0$, and then applies bisection search. Does not require an initial interval to begin the search.	✓	✓

Table 3: List of available algorithms for designing fixed and deterministic time-varying control limits in the **StatisticalProcessMonitoring.jl** package.

Affiliation:

Daniele Zago
Università degli Studi di Padova
Department of Statistics
Via Cesare Battisti, 241-243
35121 Padova, Italy
E-mail: `daniele.zago.1@phd.unipd.it`
URL: `https://dedzago.github.io/`

Journal of Statistical Software `https://www.jstatsoft.org/`
published by the Foundation for Open Access Statistics `https://www.foastat.org/`

July 2025, Volume 113, Issue 7 *Submitted:* 2023-12-02
`doi:10.18637/jss.v113.i07` *Accepted:* 2024-06-04