Operations Research Perspectives 6 (2019) 100095

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

Regression based scenario generation: Applications for performance management

Sovan Mitra[*,a], Sungmook Lim[b], Andreas Karathanasopoulos[c]

[a] University of Liverpool, Liverpool L69 7ZX, United Kingdom
[b] Dongguk Business School, Dongguk University,-Seoul, 30, Pildong-ro 1-gil, Jung-gu, Seoul 04620, Republic of Korea
[c] University of Dubai, Academic City Emirates Road, Dubai, United Arab Emirates

ARTICLE INFO

Keywords:
Simple linear regression
Performance management
Scenario generation
Stochastic programming
Forecasting

ABSTRACT

Regression analysis is a common tool in performance management and measurement in industry. Many firms wish to optimise their performance using Stochastic Programming but to the best of our knowledge there exists no scenario generation method for regression models. In this paper we propose a new scenario generation method for linear regression used in performance management. Our scenario generation method is able to produce more representative scenarios by utilising the data driven properties of linear regression models and cluster based resampling. Secondly, our scenario generation method is more robust to model 'overfitting' by utilising a multiple of linear regression functions, hence our scenarios are more reliable. Finally, our scenario generation method enables parsimonious incorporation of decision analysis, such as worst case scenarios, hence our scenario generation facilitates decision making. This paper will also be of interest to industry professionals.

1. Introduction

Firms are frequently required to undertake decisions under uncertainty, in a range of sectors such as finance [14], power production [15] and agriculture planning [5]). Stochastic Programming provides a powerful method for modelling and optimising decisions under uncertainty. Stochastic Programming has been proven to provide optimal solutions to decisions under uncertainty and significant cost savings, for example $450–$1000 million [33].

A key aspect of Stochastic Programming is scenario generation: discretisation of the random process into scenarios. It can be shown that scenario generation is particularly beneficial if a model is significantly affected by changes in value of the random variables [5]. Most importantly, the scenario generation method fundamentally determines the quality of the modelling, since crude scenario generation will not sufficiently model the random process.

Stochastic programming can be applied to a range of applications, such as optimising a firm's performance (see for example [26,35,37,42]). Performance management is an increasingly important tool used in industry [16], and is frequently used as a method to measure and improve performance. For example in [4] performance is measured using the balanced scorecard methodology. Performance management can be applied to wide range of areas in an organisation, from measuring the performance of a specific product or service, measuring the quality of a business process to measuring the overall performance of a department.

Performance management is a particularly pertinent area to Management Science because performance management is concerned with decision making issues to improve organisational effectiveness. Moreover, it has been demonstrated that performance management leads to improved financial performance (such sales growth), higher motivation amongst the workforce (due to improved measurement of performance), and greater flexibility in responding to changes in the business environment. Consequently, Management Science has alot to offer in terms of improving performance management techniques.

One frequently used method in performance analysis is simple linear regression (SLR), see for example [2,19,31,38]. Although many firms and analysts are aware of the disadvantages of SLR, it is frequently applied in performance analysis for a wide range of reasons. This is because SLR has many attractive features for performance analysis, such as applicability to large datasets, tractable implementation and little training is required to utilise it compared to other quantitative methods. Moreover, alternative quantitative methods do not necessarily provide better analysis results.

Whilst SLR is a common performance analysis tool and stochastic programming is frequently used to optimise performance, to the best of our knowledge there is no scenario generation method for regression analysis. Although many scenario generation methods exist, such as

* Corresponding author.
E-mail address: sovan.mitra@liverpool.ac.uk (S. Mitra).

https://doi.org/10.1016/j.orp.2018.100095
Received 15 May 2018; Received in revised form 20 July 2018; Accepted 16 December 2018
Available online 28 December 2018

moment matching (to be discussed in the proceeding sections), these are not directly applicable to regression analysis. Moreover, such scenario generation methods are not necessarily desirable for regression models, or performance management applications.

In this paper we propose a new scenario generation method that can be applied to SLR for performance management applications. Our method is able to provide more representative scenarios, rather than depending on unrealistic error distributions, and more reliable scenarios by resampling the data. We are also able to incorporate decision analysis information, such as worst case scenarios, by conveniently adjusting our scenario generation process.

This paper is organised as follows: in the next section we review scenario generation and survey the current literature on scenario generation. In the next section we explain our regression based scenario generation method, providing the method and discussing the advantages of the method. In the next section we conduct numerical experiments to demonstrate our scenario generation method and compare our method against a standard scenario generation, presenting and analysing our results. We finally end with a conclusion.

2. Introduction to scenario generation and literature review

In this section we introduce scenario generation and review the current literature on scenario generation methods.

2.1. Introduction to stochastic programming scenario generation

Firms are frequently interested in improving their performance, in a variety of areas of their business. In particular, the aim is to optimise over a set Θ, whose elements represent feasible decisions that a decision maker can undertake to optimise some outcome. Our goal is to optimise our objective or cost function $f(.)$ over the set Θ. To define Θ more specifically, we have [13]

$$\Theta: =\{\mathbf{s} \in \mathbb{R}^n | \mathbf{s} \in \mathbf{S}, g_i(\mathbf{s}) \geq 0, i = 1, 2, ..., m\},$$

where $g_i(\mathbf{s})$ represents constraints on the decision $\mathbf{s}$, $\forall\ i = 1, 2, ..., m$. We also note in passing that typically we have $\mathbf{s} \in \mathbf{S} \subset \mathbb{R}^n$.

Let us now define a probability space $\{\Omega, \mathcal{F}, \mathbb{P}\}$, where Ω denotes the sample space, $\mathcal{F}$ denotes a collection of events in Ω with probability measure $\mathbb{P}$. We have a filtered probability space $\{\Omega, \mathcal{F}, \{\mathcal{F}_t\}_{t\geq 1}, \mathbb{P}\}$, where the set $\{\mathcal{F}_t\}$ denotes the set of information that is available to the observer up to time t and we have

$$\mathcal{F}_u \subseteq \mathcal{F}_t \subseteq \mathcal{F}_T, \forall\ u, t \text{ with } u < t < T.$$

The set $\{\mathcal{F}_t\}_{t\geq 1}$ is also known as a filtration. We have a time index $t \in \{1, 2, ..., T\}$, where T is the final time period, and the index is also known as the time stage of the scenario tree (to be defined later). We define ω as a scenario, or a state of the world, which is a finite element of the set $\omega \in \Omega$ [5]. We can consider Ω as the set of all outcomes, and ω has the associated probability measure $\mathbb{P}$.

In stochastic optimisation the objective function and the constraints are function of the decision variable $\mathbf{s}$ and the scenarios ω [13], that is $f(\mathbf{s}, \omega)$ and $g_i(\mathbf{s}, \omega)$ respectively (for a review see [43]). If we assume (without loss of generality) that our optimisation is a minimisation problem then the optimisation is formulated as [13]

$$\min_{\mathbf{s}\in\mathbf{S}} f(\mathbf{s}, \omega),$$
$$\text{s.t. } g_i(\mathbf{s}, \omega) \geq 0, \forall\ i = 1, 2, ..., m.$$

Stochastic programming is a flexible and widely applicable method that can take into account high degrees of uncertainty. As a modelling method stochastic programming has a number of important advantages over alternatives. Firstly, one can include constraints without losing tractability in solutions. Secondly, stochastic programming can be applied to a wide range of stochastic processes and this is not always possible in other modelling methods.

A stochastic program is defined as [13]

$$\min_{\mathbf{s}\in\mathbf{S}} \int f(\mathbf{s}, \omega)\mathbb{P}(d\omega). \tag{1}$$

The integral in Eq. (1) typically cannot be solved as it is intractable, moreover, the intractability increases in the presence of constraints and these must be incorporated to have realistic models. In order to solve the integral one may apply numerical analysis methods (such as quadratures) however they may not be able to satisfactorily solve the integral, especially for non-trivial probability measures $\mathbb{P}$. An alternative solution is to minimise (as suggested in [13])

$$\min_{\mathbf{s}\in\mathbf{S}} f(\mathbf{s}, \bar{\omega}), \quad \text{where } \bar{\omega} = \mathbb{E}^{\mathbb{P}}[\omega].$$

However, such a simplification can lead to erroneous solutions (see [13] for a discussion).

In order to feasibly solve stochastic programs, the Eq. (1) is minimised as

$$\min_{\mathbf{s}\in\mathbf{S}} \left\{ \sum_{i=1}^{i=N_1} p_1^i f(\mathbf{s}_1^i, \omega_1^i) + \sum_{i=1}^{i=N_2} p_2^i f(\mathbf{s}_2^i, \omega_2^i) + ... + \sum_{i=1}^{i=N_T} p_T^i f(\mathbf{s}_T^i, \omega_T^i) \right\}, \tag{2}$$

where N_t is the total number of scenarios at time index t; ω_t^i is scenario i at time index t; p_t^i is the branch probability of scenario ω_t^i at time index t in the scenario tree; $f(\mathbf{s}_t^i, \omega_t^i)$ is the objective function (as before). The $\mathbf{s} \in \{\mathbf{s}_t^i\}$ is the set of decisions at each time stage $t \in \{1, 2, ..., T\}$. The decision variable $\mathbf{s}_t$ are adapted to the filtration $\mathcal{F}_{t-1}$ to reflect the non-anticipative condition of stochastic programming. The non-anticipative condition means that decisions are made without anticipating future outcomes. The stochastic programming formulation (2) can be easily minimised by computation, including in the presence of constraints.

In order to solve Eq. (2) we require a method to produce $\omega_t^i, \forall i, t$ and this is known as scenario generation. Scenario generation involves discretising a random process into a set of discrete outcomes ω_t^i. To visualise scenarios one can draw a scenario tree (see [5] for an example), where the root node represents 'today'. The times $t = 1, 2, ..., T$ are also known as time stages of the scenario tree. Each branch of the scenario tree also has a probability p_t^i, with the standard probability constraint

$$0 < p_t^i < 1, \forall\ i, t.$$

One can also see from Eq. () that the quality of the solution depends significantly upon the scenario generation method applied, hence scenario generation is vital to stochastic programming.

The purpose of scenario generation is to find a good approximation of the original distribution, with discrete scenario values $\omega_t^i\ \forall i, t$. This is essentially a goodness of fit test, for example if we apply the Kolmogorov–Smirnoff test then we would minimise

$$\sup_y |F_t^g(y) - F_t^l(y)|,$$

where $F_t^g(.)$ is the cumulative distribution function associated with the scenarios at time t and $F_t^l(.)$ is the original distribution at time t. The scenario generation process is a non-trivial method because some approximation methods perform well for small N_t but perform poorly as N_t increases. In fact, Kaut and Wallace [23] mention that some scenario generation methods approximate distributions perfectly if $N_t \to \infty$, that is

$$|F_t^g(y) - F_t^l(y)| \to 0, \text{ as } N_t \to \infty, \forall\ y,$$

but perform poorly for small N_t.

In addition to approximating the original distribution, the purpose of scenario generation is to facilitate *decision analysis*. In other words, rather than using a probability distribution for decision analysis, many decision makers prefer to produce and analyse scenarios as they can be easier to apply to decision making in real world situations. Hence

scenario analysis and scenario generation are popular in industry. In such scenario analyses, one is not just interested in approximating a distribution but also generating scenarios of interest to a decision maker. For example, a particular case in point in industry is incorporating 'worst case scenarios', or the effect of low probability but high impact scenarios [43] in modelling and optimisation methods.

In addition to obtaining scenarios of interest from a probability distribution, decision makers are also interested in creating scenarios that can incorporate their expert opinions or judgements. This is desired for a number of reasons. Firstly, there is sometimes insufficient representative data to ensure the scenario generation process is unbiased, hence decision makers want to incorporate their own judgement or opinions. For example, in financial scenario generation a data sample of the past 5 years would not be representative of the next 5 years (5 years prior to the Global Financial Crisis most stock markets were generally increasing but after the financial crisis started most markets declined).

Secondly, decision makers have become increasingly sceptical of any quantitative models since the start of the Global Financial Crisis, as financial models have been partly blamed for the crisis and were unable to fully take into account all the risks and variables in their applications. Consequently, decision makers want to incorporate their own expert opinions and judgements, rather than relying solely on quantitative models.

2.2. Scenario generation methods

In this section we review the main scenario generation methods in stochastic programming. Firstly, the most common type of scenario generation method is Monte Carlo sampling based scenario generation, where different Monte Carlo based sampling methods give different scenario generation methods [1]; this is one of the most popular scenario generation methods due to its analytical and computational simplicity. Examples of Monte Carlo based scenario generation are [22] and the famous Russell–Yasuda Kasai model [8].

Monte Carlo based scenario generation essentially involves random sampling of a probability distribution, so that each sample produces a scenario ω_t^i. In Monte Carlo based scenario generation, we typically assume

$$p_t^i = \frac{1}{N_t}, \forall\ i = 1, .,N_t,\ t = 1, 2,..., T.$$

In other words, all scenarios ω_t^i have equal probabilities at a given time stage t. The simplest Monte Carlo based scenario generation method is inverse transform sampling based scenario generation. In inverse transform sampling we firstly obtain N_t random samples from the Uniform distribution over the unit interval $U(0, 1)$, that is

$$\xi_1, \xi_2,...,\xi_{N_t}, \quad \text{where } \xi_{(.)} \sim U(0, 1).$$

To produce scenarios ω_t^i at time stage t in the scenario tree, we require the inverse cumulative distribution function associated with the scenarios ω_t^i a time stage t, that is $F_t^{-1}(.)$ [27]. The scenario (or equivalently the sample value) is given by

$$\omega_t^i = F_t^{-1}(\xi_i),\ \forall\ i = 1,\ ...,N_t,\ t = 1,\ ...,T.$$

Consequently, each cumulative distribution function value corresponds to a unique scenario value ω_t^i and sufficient random sampling would reproduce the original probability distribution.

Another Monte Carlo based scenario generation method is stratified sampling based scenario generation. Stratified sampling is a modification of inverse transform sampling, however in stratified sampling we stratify or segment the sampling of the probability distribution. In stratified sampling we firstly segment the unit interval [0,1] such that

$$0 < I_1 < I_2 \cdots < I_n \leq 1.$$

As in standard inverse transform sampling, we require a random sample ξ but instead of $\xi \sim U(0, 1)$, we firstly obtain a random sample from the first interval, that is $\xi_1 \sim U(0, I_1)$. We then obtain a random sample from the next consecutive interval, that is $\xi_2 \sim U(I_1, I_2)$, and so on until all intervals have been sampled and then we return to the first interval. The scenario values ω_t^i are obtained as before, by using the inverse cumulative distribution function for time stage t:

$$\omega_t^i = F_t^{-1}(\xi_i).$$

However, in stratified sampling we are giving more importance to particular intervals $[I_i, I_j]$ and so increasing the probability of particular samples being obtained in $F_t^{-1}(\xi_{(.)})$. Hence stratified sampling will result in more scenarios (or equivalently samples) from the section of the probability distribution of interest.

A popular stratified sampling method is Latin Hypercube sampling [32]. This involves dividing the unit interval [0,1] into $n + 1$ equal length subintervals, that is

$$I_i = \frac{1}{n+1}, \quad \text{where } 0 < I_1 < I_2 \cdots < I_n \leq 1, \quad \text{and } i = \{1, 2,...,n\}.$$

As before, we require N_t random samples $\xi_1, \xi_2, ...,\xi_{N_t}$ from the uniform distribution, however ξ_1 will be taken from the first subinterval $U(0, I_1)$, ξ_2 will be taken from the subinterval $U(I_1, I_2)$ etc. The scenario value ω_t^i is obtained by using the inverse cumulative distribution function for time stage t as before, $\omega_t^i = F_t^{-1}(\xi_i)$. In sampling each interval at each iteration one obtains a more representative statistical sample compared to inverse transform based sampling, when the number of samples is small.

The second main group of scenario generation methods is statistical property matching. Each sub-tree within a scenario tree consists of a set of p_t^i probabilities with scenario values ω_t^i. Consequently, each scenario subtree will have some statistical properties, such as expectation $\mathbb{E}[.]$, or moments. Therefore a viable scenario generation process is to match the statistical properties of the scenario tree at time t to some target probability distribution's properties.

A popular property matching scenario generation method is the moment matching method proposed by Hoyland and Wallace in [21]. In this method, the scenarios in a given subtree originate from some stochastic process $Z(t)$, with known kth moments at time t

$$\mathbb{E}^{\mathbb{P}}[Z^k(t)] = \int_{-\infty}^{\infty} Z^k(t) d\mathbb{P},\ \forall\ k \in \mathbb{N}^+,\ t \in 1,\ 2,\ ...,T.$$

Furthermore the moments for the scenarios in the subtree at time stage t are

$$\sum_{i=a}^{i=b} p_t^i (\omega_t^i)^k,\ \forall\ k \in \mathbb{N}^+,\ t \in 1, 2,..., T,$$

where $i = a,\ ...,b$ are the scenarios in the subtree, at time t. The moment matching scenario generation process can therefore be expressed as

$$\min_{p_t^i, \omega_t^i} \ Q\left(\left|\left| \sum_{i=a}^{i=b} p_t^i (\omega_t^i)^k - \mathbb{E}^{\mathbb{P}}[Z^k(t)] \right|\right|\right),$$

where $Q(.)$ is typically some distance measure, such as a Euclidean distance measure.

The moment matching method is therefore an optimisation problem in itself, where one optimises the choice of p_t^i, ω_t^i, $\forall i,\ t$, to match the moments $\mathbb{E}^{\mathbb{P}}[Z^k(t)]$. Consequently, this scenario generation must be implemented using an optimisation program, whereby the program's constraints, objectives or decision variables can be assigned to p_t^i and ω_t^i. Examples of statistical property matching scenario generation are given in [18,30,40].

The property matching scenario generation method has a number of advantages. Firstly, it does not require a full specification of the target probability distribution. For example, for the hypergeometric distribution the probability mass function is given by

$$\mathbb{P}(Z = c) = \frac{\binom{d}{c}\binom{q-d}{a-c}}{\binom{q}{a}},$$

where $\binom{\cdot}{\cdot}$ is the binomial coefficient, q is the population size, d is the number in success states in the population, a is the number of draws and c is the number of observed successes. Therefore to fully specify this distribution we would require estimates of the three parameters d, q and a. However, with moment matching scenario generation, we may only need to specify two moments of the distribution.

The fact that we do not require a full specification of the distribution is desirable for a number of reasons. However, one may be able to determine one or two moments of a distribution with high accuracy (such as the mean etc.). Consequently, property matching scenario generation is more robust to distribution misspecification.

Secondly, property matching scenario generation can easily incorporate model specific information in the scenario trees, unlike other methods. For example, one may wish to generate scenarios from a distribution with heavier tails, and this can be easily incorporated by adjusting the moments of the distribution for moment matching by a weighting factor γ_t^i. The scenario generation method would now be:

$$\min_{p_t^i, \omega_t^i} Q\left(\left|\left| \sum_{i=a}^{i=b} p_t^i (\gamma_t^i \omega_t^i)^k - \mathbb{E}^{\mathbb{P}}[Z^k(t)] \right|\right|\right). \tag{3}$$

Alternative scenario generation methods, such as Monte Carlo based scenario generation, would not be able to adjust their scenario generation method as easily as moment matching. One would require modifying the entire function $F_t^{-1}(.)$ for the heavy tail of the distribution, which is non-trivial task.

Thirdly, property matching can perform better at generating scenarios over small N_t, that is smaller sized scenario trees, compared to other methods. This is because property matching based scenario generation is not as dependent on N_t compared to other methods to produce representative scenarios of the distribution. Finally, property matching scenario generation makes it possible to easily implement correlations compared to other scenario generation methods.

The main disadvantage of property matching based scenario generation is that implementing the method requires an optimisation program for p_t^i, ω_t^i, $\forall i, t$. The optimisation is typically nonlinear [21] and so a non-trivial task. In fact in many instances feasible solutions may not exist for p_t^i, ω_t^i, $\forall i, t$, or the quality of the solutions will heavily depend on the optimisation method. Also, there may not exist a unique scenario tree and therefore one will require a criteria for choosing between scenario trees.

The last main group of scenario generation methods is scenario generation by simulation. In simulation based scenario generation we are typically concerned with creating scenario tree paths. For example, one would produce

$$\{\omega_1^1, \omega_2^1, \omega_3^1, \omega_4^1, \ldots, \omega_T^1\} \in \Omega,$$

that is a set of scenarios over consecutive time stages, from one sample path. For the next sample path one would then produce

$$\{\omega_1^2, \omega_2^2, \omega_3^2, \omega_4^2, \ldots, \omega_T^2\} \in \Omega,$$

that is a set of scenarios over consecutive time stages, but over different scenario numbers. The scenario tree is therefore created by one sample path at a time.

The sample paths are generated by simulating a stochastic process, hence simulating different stochastic processes leads to different simulation based scenario generation methods. In particular, we simulate the stochastic component of some stochastic process, for example a Wiener process $W(t)$. This is especially useful in industry and financial applications because Wiener processes model many variables of interest, for example the model of stock prices [17].

In order to explain the scenario generation process by simulation, for the benefit of exposition we will assume the stochastic process is a stochastic differential equation. Stochastic differential equations are frequently used in finance to model a range of variables, for example interest rates, GDP and inflation. An example of scenario generation from stochastic differential equations can be found in [10].

A stochastic differential equation in $V(t)$ is a differential equation of the form

$$dV(t) = \mu(V, t)dt + \sigma(V, t)dW,$$

where

$$W_{t+u} - W_t \sim \mathcal{N}(0, u^2),$$

where $\mathcal{N}(\alpha, \kappa^2)$ denotes the Normal distribution with mean α and variance κ^2, $\mu(V, t)$ is known as the drift and $\sigma(V, t)$ denotes the volatility. One example of a stochastic differential equation is the Brownian motion process

$$dV(t) = \mu(t)dt + \sigma(t)dW.$$

In simulation we generate values at discrete points in time $0 < t_1 < t_2 < \cdots < t_n$, that is $V(t_1), V(t_2), \ldots, V(t_n)$, we therefore require values $W(t_1), W(t_2), \ldots, W(t_n)$. One can simulate values of $W(t_i)$ by using the following simulation process [17]:

$$W(t_{i+1}) = W(t_i) + (\sqrt{t_{i+1} - t_i})R_{i+1}, \quad \text{for } i = 0, 1, 2, \ldots n-1,$$

where R_{i+1} is a random sample drawn from the standard Normal distribution $\mathcal{N}(0, 1)$. The resulting simulation process is therefore

$$V(t_{i+1}) = V(t_i) + \mu(t_{i+1} - t_i) + \sigma(\sqrt{t_{i+1} - t_i})R_{i+1},$$
$$\text{for } i = 0, 1, 2, \ldots n-1.$$

One can therefore obtain the scenario tree path $\{\omega_1^1, \omega_2^1, \omega_3^1, \ldots, \omega_n^1\}$ by calculating $V(t_{i+1})$ for $i = 0, \ldots, n-1$ and we have $\omega_i^1 = V(t_i)$. To obtain the second scenario tree path $\{\omega_1^2, \omega_2^2, \omega_3^2, \ldots, \omega_n^2\}$ we would restart the simulation at $V(t_1)$ and obtain the scenarios such that $\omega_i^2 = V(t_i)$.

Another popular simulated stochastic differential equation in industry is the Geometric Brownian motion. This is because many real world phenoma and variables are modelled by Geometric Brownian motion and Samuelson is credited with introducing it to economic applications [39]:

$$dV(t)/V(t) = \mu dt + \sigma dW.$$

The scenarios ω_i^t are obtained in a similar process to the approach taken for Brownian motion. We therefore simulate the stochastic process to obtain a sample path and assign the scenario values ω_i^t to them.

In addition to simulating stochastic differential equations, another set of commonly simulated equations for scenario generation are econometric equations. An econometric model for the dependent variable $V(t)$ is generally of the form

$$V(t) = \beta_1 Z_1(t) + \beta_2 Z_2(t) + \cdots + \beta_n Z_n(t) + \epsilon(t),$$

where β_i is a constant for $i = 1, 2, \ldots, n$ and $Z_i(t)$ are independent variables. The variable $\epsilon(t)$ is a noise term, for example

$$\epsilon(t) \sim \mathcal{N}(0, 1).$$

The most commonly known econometric model is the GARCH model [6], which models volatility of economic time series and is one of the most successful econometric models. The econometric model is simulated by firstly calibrating the model parameters β_i and $Z_i(t)$, $\forall\, i = 1, 2, \ldots, n$. We then obtain $V(t)$, at each time stage t, by randomly sampling the distribution of $\epsilon(t)$. Consequently we obtain a sample path and this provides a scenario tree path. We repeat the process to obtain additional sample paths which also provide the other scenario tree paths.

The advantage of simulation based scenario generation is that we can generate scenarios from practically any stochastic process. This is a significant advantage because there exist many analytically intractable

processes where other scenario generation methods become infeasible. For simulation based scenario generation we randomly sample only from the noise term's distribution, which is generally easy to implement, and so simulation based scenario generation is possible for most processes.

The second advantage of simulation based scenario generation is that it enables one to capture the behaviour of complex processes through the sample paths that give the scenario tree paths. For example, an Ornstein–Uhlenbeck process is defined by

$$dV(t) = \alpha(\mu - V(t))dt + \sigma dW,$$

where α is a constant. Both processes have similar probability distributions and so would provide similar scenarios under Monte Carlo based scenario generation. However, both processes have different sample paths, therefore simulation based scenario generation would produce different scenarios. The key disadvantage of simulation based scenario generation is that it typically requires running multiple sample paths and time periods in order to obtain meaningful scenario values. This can consequently lead to an excessive number of scenarios, which can increase the computation time of the stochastic optimisation.

For the benefit of completeness we mention that other scenario generation processes exist and that a comprehensive of all scenario generation methods would be beyond the scope of this paper. We mention that the reader may also be interested in firstly surveying scenario generation methods consistent with no-arbitrage theory [25]. Klaassen [24] investigated the conditions required for financial scenario generation to conform to no arbitrage conditions and states that in an N-node scenario tree with n assets, spanning time period T, that no arbitrage opportunities exist if

$$\exists\ p^i > 0,\ \forall\ i = 1, 2, ..., N,$$

such that

$$V_j(0) = e^{-rT} \sum_{i=1}^{i=N} p^i \omega^{(i,j)},\ \forall\ j = 1, 2, ..., n,$$

where r is the riskless rate, $\omega^{(i,\ j)}$ is the scenario value in branch i for asset j, and $V_j(0)$ is the initial price of asset j at time 0. Other scenario generation methods of interest include boostrapping which is used for small samples of data (a detailed review of is given in [29]) and is computationally intensive; this has been used in scenario generation in [7,9]. Another scenario generation method is scenario reduction (which reduces the given scenario tree size); see [12,20]) for examples.

3. Regression based scenario generation

In this section we explain our scenario generation method, generating scenarios from linear regression in performance measurement applications. We firstly introduce linear regression, the motivation for scenario generation in linear regression and then discuss our method.

3.1. Introduction to linear regression and performance measurement

Performance management and measurement are increasingly important tools used in industry [16,28,36]. It is frequently used as a method to measure and improve performance. For example in [34] performance management systems are analysed, such as the organisational processes, monitoring methods and methods of learning. In [4] performance is assessed in terms of the balanced scorecard methodology, and combined with a fuzzy set concept to measure supply chain performance.

Whilst a range of methods exist for measuring performance, which can be qualitative or quantitative, firms have increasingly adopted quantitative methods. This is because quantitative methods tend to include more objectivity and they are amenable to substantial analysis and insight. Alternative performance measurement methods (such as qualitative methods) rely too heavily on subjective opinions and they do not facilitate further analysis (such as forecasting). Hence firms have adopted quantitative measures.

One particularly popular quantitative tool in performance measurement is simple linear regression (SLR), see for instance [31,41] for examples. For a given dataset $\{(x_1, y_1), (x_2, y_2), ..., (x_t, y_t)\}$ for $t \in 1, 2, ..., T$, one can model the dependent variable Y_t as

$$Y_t = \beta_0 + \beta_1 x_t + \epsilon_t,$$

where β_0 and β_1 are model parameters and ϵ_t denotes the error or noise term. We also assume that the error term has $\mathbb{E}[\epsilon_t] = 0$, $Var(\epsilon_t) = \sigma^2\ \forall t$, and that ϵ_t is identically and independently distributed, so that

$$\epsilon_1, \epsilon_2, ..., \epsilon_t \sim \mathcal{N}(0, \sigma^2),\ \forall\ t \in 1, 2, ..., T.$$

In order to calibrate the SLR model parameters, specifically the regression coefficients β_0 and β_1, we apply the Ordinary Least Squares method. Let us define

$$H = \sum_{t=1}^{t=T} [y_t - (\beta_0 + \beta_1 x_t)]^2,$$

and the ordinary least squares method calibrates the model parameters such that

$$\frac{\partial H}{\partial \beta_0} = 0,$$
$$\frac{\partial H}{\partial \beta_1} = 0.$$

Although many firms and users are fully aware of the significant disadvantages of SLR, particularly for the non-linear and noisy data that occurs in performance measurement applications, SLR is widely and frequently used. In other words, if we quantify the error in the model δ as

$$\delta = \Lambda(\|Y_t - y_t\|),$$

where $\Lambda(.)$ is some distance measure, then δ can be extremely large, especially for some values of t.

One of the reasons SLR is extremely popular is that it is very easy to implement with a range of datasets and one can obtain useful insights from the data (although many assumptions must be applied). For example, large datasets can be easily analysed on any Excel package, with little training and knowledge required to obtain the results. Secondly, it is one of the parsimonious quantitative methods that can be used for performance analysis. The SLR method can be taught and understood to managers and non-specialists without any significant mathematical training. In fact many users cite that alternative quantitative methods are no better. Consequently, this makes SLR a popular analysis tool.

3.2. Scenario generation method

The SLR model is typically used to forecast some variable related to performance, for example sales, income, market share etc. In particular, managers are especially keen to forecast future performance in relation to some variable. This enables firms to plan into the future (eg increase the capacity of their business), forecast costs, to determine whether performance is line with future expectation or whether corrective actions need to be taken.

In order to achieve a forecast, one can extrapolate the regression line, however this would not be informative because performance measurement data can be highly non-linear. Consequently, the SLR regression line tends not to provide a good forecast of the future. Additionally, most organisations engage in scenario analysis for forecasting future values; rather than using one value as a future value firms like to consider a range of values or scenarios. This is because it is unrealistic that one value will closely approximate a single future outcome. Moreover, scenarios are required for stochastic optimisation

methods such as stochastic programming.

To generate or forecast scenarios from SLR at time period T', where $T' > T$, one could consider the error distribution $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$, and apply some Monte Carlo based (or other) scenario generation method to the distribution of $Y_{T'}$. In other words the random sample or scenario from $Y_{T'}$ would be given by

$$Y_{T'} = \beta_0 + \beta_1 x_{T'} + \tilde{\epsilon}_{T'},$$

where $\tilde{\epsilon}_{T'}$ is a random sample from $\mathcal{N}(0, \sigma^2)$. The key disadvantage of this method is that it is well known that the error distribution $\epsilon_t \sim \mathcal{N}(0, \sigma^2)$ is frequently a poor model of future values. Consequently, sampling from the error distribution does not necessarily provide a viable scenario generation method.

We now explain our scenario generation process in more detail and for the benefit of exposition we restrict our scenario generation process to a single period. To generate scenarios from our regression model for forecasting, we take a different approach. Given that SLR is a data driven model, that is the SLR model's results and parameters are completely determined by the data, scenario generation from the error distribution alone is not a meaningful scenario generation process. This is because we are assuming that the SLR and the input data will be representative of the future, however historic data is not always a good representation of future outcomes and the forecasting capability of one SLR is limited. Therefore to generate scenarios we start with the premise of varying the data inputs into the SLR model.

We firstly partition our data into clusters by undertaking cluster analysis. Clustering is a method of partitioning data into mutually exclusive subsets, which are called clusters. Let the entire set of data ζ be partitioned by n subsets ϕ_i, for $i = 1, 2, ..., n$, so that

$$\begin{aligned} \zeta &= \phi_1 \bigcup \phi_2 \bigcup ..... \bigcup \phi_n, \\ \phi_i \bigcap \phi_j &= \varnothing, \forall\, i, j \in \{1, 2, ..., n\} \text{ s.t. } i \neq j, \end{aligned}$$

then we say we have ϕ_i clusters in ζ. Generally, we can consider a cluster as a group of objects that are more similar to one another than to members of other clusters.

To ensure our sampling includes important segments of the data we firstly cluster the data. Otherwise the scenario generation process will be based on unrepresentative data. This also enables us to generate more SLRs and generate more realistic scenarios values. In performance based data we expect clusters of data to exist because it is common for performances to occur in clusters, rather than data being randomly distributed.

The fact that performance management data tends to occur in clusters provides a significant advantage to our scenario generation method. This is because we can easily determine the number of clusters n by inspecting the data. Normally for a computer program to determine n is a non-trivial issue because data generally does not easily segment into clear clusters. Moreover, the number of clusters is generally not a unique solution but can be a range of viable numbers. Furthermore, if one can determine n then the cluster method (and therefore our scenario generation method) can produce better results.

In order to assign every datapoint to a particular cluster ϕ_i, we require some metric to assign them. Typically this is based on assigning each datapoint to a cluster such that we minimise the distance of the datapoint to the nearest cluster. If we have a data set belonging to one cluster ϕ_1, that is $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_k\} \in \phi_1$ where $\mathbf{v}_i \in \mathbb{R}^n$, $\forall\, i = 1, 2, ..., k$, then the centroid $\mathbf{C}_i \in \mathbb{R}^n$ is given by

$$\mathbf{C}_i = \frac{\mathbf{v}_1 + \mathbf{v}_2 + \cdots + \mathbf{v}_k}{k}. \tag{4}$$

A centroid is the central value of each cluster.

To cluster data we apply the following algorithm. Let there exist $\phi_1, ..., \phi_m$ clusters with m centroids $\{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_m\}$, and T datapoints $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_T\}$. To commence the algorithm we must initialise $\{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_m\}$ because we have not assigned any datapoints $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_T\}$ to $\phi_1, ..., \phi_m$ yet, hence the centroids have no value. Once $\{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_m\}$ values are assigned, for each $j \in 1, 2, ..., m$ we assign $\mathbf{v}_j$ to cluster ϕ_i, such that we minimise

$$\min_{j \in 1,..,T} \Lambda(\|\mathbf{v}_j - \mathbf{C}_i\|)\ \forall\ i \in 1, 2, ., m, \tag{5}$$

where $\Lambda(.)$ is some distance measure.

Once all the datapoints $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_T\}$ are assigned to a cluster ϕ_i we now re-calculate the centroid values $\{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_m\}$ using Eq. (4) (as their values would have changed with the new datapoint memberships). We then again re-assign $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_T\}$ by applying Eq. (5) and check if the set membership of $\phi_i\ \forall i$ has significantly changed since the previous assignment. If the assignment has significantly changed we repeat the algorithm until there is no significant change in cluster membership (or some other stopping criteria is met). The grouping of datapoints into clusters is therefore an iterative process.

For the function $\Lambda(.)$ a number of possible distance measure functions are available. We apply the Euclidean distance measure, that is for data $\mathbf{v}_j \in \mathbb{R}^N$ and centroids $\mathbf{C}_i \in \mathbb{R}^N$, the Euclidean distance is given by

$$\Lambda(\|\mathbf{v}_j - \mathbf{C}_i\|) = \sqrt{(v_{j1} - C_{i1})^2 + (v_{j2} - C_{i2})^2 + \cdots + (v_{jN} - C_{iN})^2}, \tag{6}$$

where $\mathbf{v}_j = \{v_{j1}, v_{j2}, ..., v_{jN}\}$ and $\mathbf{C}_i = \{C_{i1}, C_{i2}, ..., C_{iN}\}$. Although alternative distance measures are possible (such as the Mahalanobis distance or Manhattan distance measures) such measures do not necessarily improve clustering performance. For example, it has been found that the Mahalanobis distance measure is less robust to noisy data.

For SLR the dataset $\mathbf{v}_j$ is restricted to $\mathbf{v}_j \in \mathbb{R}^2$, that is cartesian coordinates $\{(x_1, y_1), (x_2, y_2), ..., (x_T, y_T)\}$. We stratify the data in each cluster into n strata, by values y_i. One can choose any n but $n = 2$ is generally sufficient. We then produce n SLRs Y_t^i for $i \in 1, ..., n$, where Y_t^i is obtained by random cluster sampling of data from strata i in each cluster. Assuming we create two SLRs Y_t^1 and Y_t^2 then we have:

$$Y_t^1 = \beta_0^1 + \beta_1^1 x_t + \epsilon_t^1,$$

and

$$Y_t^2 = \beta_0^2 + \beta_1^2 x_t + \epsilon_t^2.$$

To produce scenarios at time T' we sample from the distributions associated with $Y_{T'}^1$ and $Y_{T'}^2$.

We mention in passing that our scenario generation method differs from Beraldi and Bruni [3]. In [3] scenario reduction is proposed as a scenario generation method, whereby scenarios are firstly produced, and then clustered together. A reduced set of scenarios are produced by taking a representative scenario value from each cluster. In our scenario generation method we produce the scenarios by sampling from the distributions associated with $Y_{T'}^1$ and $Y_{T'}^2$. Whilst the data for $Y_{T'}^1$ and $Y_{T'}^2$ may have originated from clustered data, we do not use the clustering process itself to produce scenarios or implement scenario reduction.

Our scenario generation method has a number of advantages. Firstly, our scenario generation method produces more representative scenarios compared to using a single SLR, fitted to the original data. Linear regression is a data-driven method, therefore the regression will only provide meaningful scenarios if the data inputs are also meaningful. However, fitting a regression line to a single set of (performance measurement) data is unlikely to produce a representative set of data, as data frequently changes with samples. To ensure we take a representative data sample, we group the data into clusters, so that we have identified the representative segments of the population. This is because performance measurement data (as well as many other types of data) exist in clusters. We then use cluster based sampling to obtain more than one sample of data, rather than relying on one sample of data for modelling.

Secondly, our scenario generation method is more robust to data, that is it is less affected by overfitting to data. Overfitting occurs when a SLR's forecast underperforms because the regression line is too reliant on historic data for forecasting purposes. Whilst fitting to data is advantageous if the data is an accurate representation of the future, for many applications (in particular performance management applications) the data frequently changes with each sample. Hence overfitting is a significant disadvantage in scenario generation for SLR.

To achieve more robust scenario generation and overcome overfitting, our scenario generation method incorporates the robustness principle of scenarios in [11]. In [11] scenario robustness is defined as resistance to variation in input samples used for scenario generation. In our method we achieve robust scenario generation by engaging in random sampling of the original dataset, and we use a set of SLRs Y_t^i (rather than a single SLR) to generate our scenarios. This ensures the input data sample is more varied and our generated scenarios are less likely to be affected to changes in input data sample values. Hence our scenario generation method is more robust to data.

Thirdly, our scenario generation method can easily incorporate expert opinions and judgement, or other modelling information in our method. If one were to use an alternate scenario generation method such as moment matching scenario generation (one of the easiest and simplest scenario generation methods) then one would need to choose appropriate values for γ_t^i $\forall i$, t in Eq. (3), however, there is no specific method for choosing γ_t^i and so the scenario generation process can become arbitrary to some extent. Moreover, if one aimed to incorporate expert opinion or judgement for a single scenario $i = 1$ (a common reason for this is incorporating a 'worst case scenario' in the scenario tree) then the choice of γ_t^1 may unintentionally impact the value of all other scenarios $\forall i \neq 1$. Hence expert opinion or judgement incorporation is not a completely straightforward process in moment matching.

In our scenario generation method, one can incorporate expert opinion by changing the strata one draws samples from, or one can directly adjust the value of samples within the clusters. The processes are straightforward to implement for either method. If we adjust the value of samples within each cluster then we can directly utilise quantitative data value as our expert opinion or judgement, unlike arbitrary choices of γ_t^i. Moreover, our scenario generation process does not lead to unintentional changes in the value of other scenario values.

4. Numerical experiment

In this section we conduct numerical experiments to demonstrate our new scenario generation method. We explain our method, present our results and then discuss our results.

4.1. Method

In this section we conduct numerical experiments to demonstrate our scenario generation method using performance measurement data from IBM Watson Analytics. An algorithm is given in Appendix 1. Our data contains information on monthly pay and the number of years of working in a company. This reflects a common application of performance management, where management would hypothetically like to see a positive trend between years of work and income. This is because it is frequently assumed, on aggregate, that a worker is more skilled according to the more years of experience he has obtained. Consequently correct performance management should reveal that workers with greater years of work should earn higher incomes.

The data is collected from approximately 1500 workers, giving 1500 data points. The workers are sampled over working years between 0 to 40 years, and the monthly income is sampled in the range \$0 to \$20,000 per month. Typically in performance management the managers would fit a SLR model to obtain some insight on performance. As no data exists beyond 40 years, a possible management application would be forecasting scenarios for monthly income for working ages beyond 40 years. We would therefore need to generate scenarios for working ages at 50 years.

The scenario generation of monthly incomes for working ages at 50 years is a realistic application for performance management because many workers are living longer, retirement ages have also been increased and workers are aiming to work longer. Consequently, a firm would need to know the expected monthly income required for employees with longer working years. This would be an important question because firms may need to set aside funds to pay for higher earning workers.

We apply our scenario generation method using two SLRs Y_t^1 and Y_t^2, for $t = 50$ years, and create scenarios from their distributions. We denote the scenarios from our scenario generation method at time t by $\tilde{Y}_t$. To provide a comparison of our scenario generation method we also generate scenarios using a standard SLR model, that is fitting an SLR to the original data. We will denote this standard SLR by Y_t^η with model

$$Y_t^\eta = \beta_0^\eta + \beta_1^\eta x_t + \varepsilon_t^\eta.$$

The model Y_t^η resulting distribution at $t = 50$ years will be used to obtain scenarios by Monte Carlo based scenario generation.

4.2. Results

In this section we give our results.

4.3. Analysis

In Fig. 1 we have a plot of the original empirical data of working years against monthly income (\$), for approximately 1500 data points. Although it can be visually seen from a plot of the data in Fig. 1 that we do not have a linear trend, SLR is still a commonly applied tool for data analysis in many social science applications, such as economics or performance measurement. For any regression model there exists uncertainty in parameter estimation, for example in β_i^j where $i = 0, 1$ and $j = 1, 2, \eta$. However, as mentioned previously, it is understood in performance management that the data will be highly non-linear as it tends to cluster (such as in Fig. 1) and the regression line is only an approximation of the data (such as in Fig. 1). The SLR is still the preferred method for performance management because it is a highly tractable method and easy to implement. Although there may be uncertainty in the parameter values, the focus of our study and performance management is the fitted regression line (and the associated scenarios).

As one can see from Fig. 1, performance analysis data tends to occur in clusters, as we have previously mentioned. One can see approximately two clusters occurring in the data at 0–20 years, and then a second group at data values 20–40 years. In Fig. 1 the regression line for Y_η is shown, and as one might expect there is a positive relationship between Working Years and Monthly Income. This implies that workers (in general) are being correctly rewarded in relation to their performance (specifically their skills and experience).

In Fig. 2 we have a plot of the resampled data, used in our scenario generation method. In other words Fig. 2 provides a plot of the samples arising from cluster based sampling of the original data. One can see by inspection that Fig. 2 looks similar to Fig. 1; we have two clusters of data in the same ranges of 0–20 years and 20–45 years. This is a reassuring result because we want to generate scenarios that are representative of the original data, hence a highly significant difference between Figs. 1 and 2 would not indicate this. We have also fitted a SLR line in Fig. 2 for comparison to Y_η in Fig. 1. We note that both regression lines are similar in gradient and y-intercept, implying that both sets of data have similar regression properties.

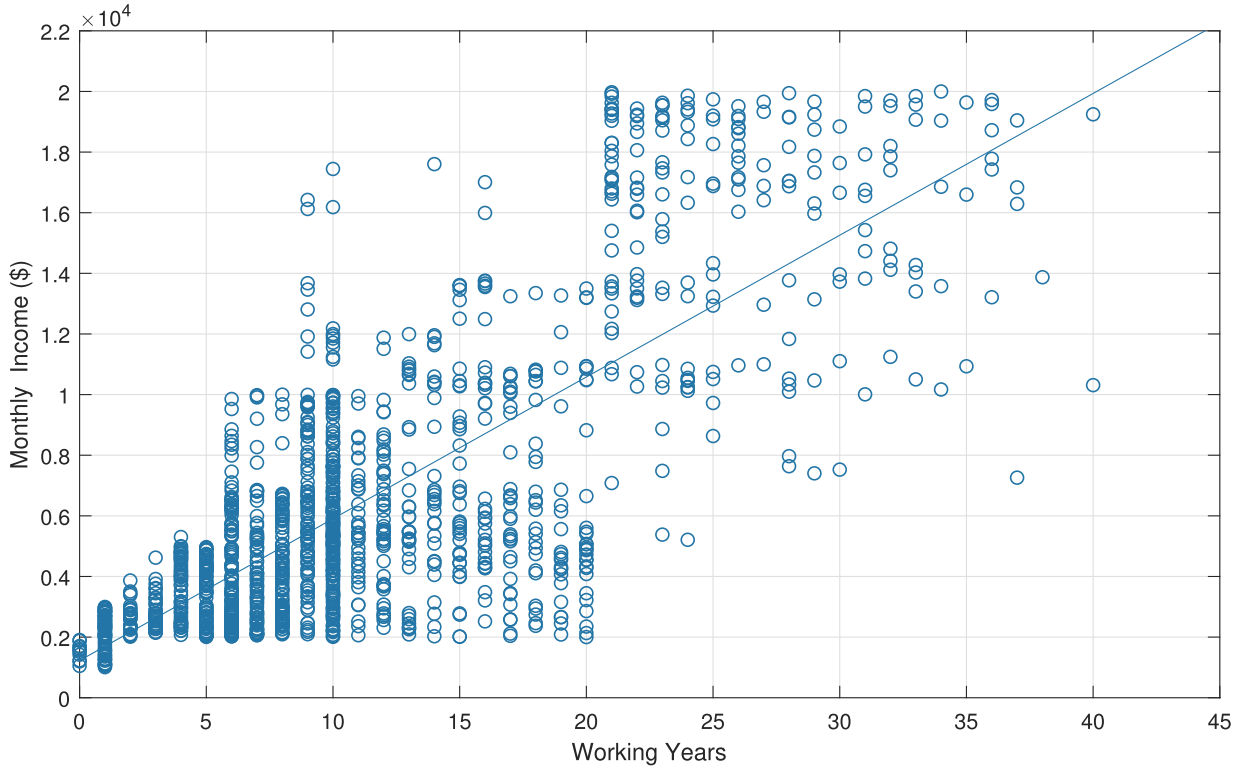

Fig. 1. Graph of empirical (Original) data with regression line Y_t^{η}.

In Table 1 we have the scenarios generated from $\tilde{Y}_t$ and Y_{η}. We have also calculated the percentage difference in the value of the scenarios in the final column. As one can see, whilst the value of the scenarios are not dissimilar, there are significant differences, particularly for scenarios 1–5 where there is an approximately 20% difference in value. This is a significant difference and provides useful information to

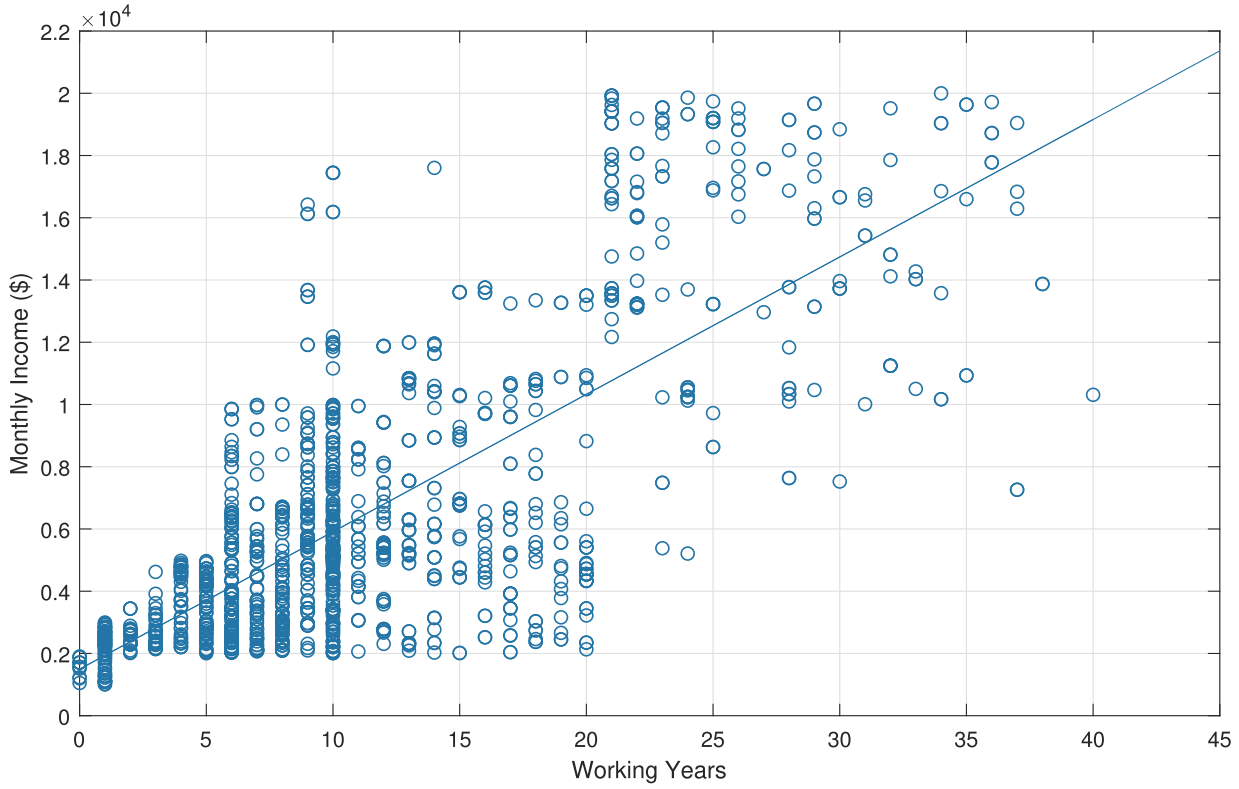

Fig. 2. Graph of sampled data with a regression line.

Table 1
Scenario values for $\tilde{Y}_t$ and Y_t^{η}.

Scenario No.	$\tilde{Y}_t$ Scenarios	Y_t^{η} Scenarios	Percentage Difference (%)
1	15896.09	19530.94	−18.61
2	16523.46	20458.31	−19.23
3	16900.10	21015.06	−19.58
4	18356.52	23167.92	−20.77
5	19784.90	25279.33	−21.73
6	24874.10	25730.62	−3.33
7	25984.96	26915.25	−3.46
8	26978.06	27974.29	−3.56
9	27639.53	28679.68	−3.63
10	29497.15	30660.66	−3.79

performance management, that is under 5 scenarios we can expect a lower payment to workers compared to using Y_{η} scenario generation.

The magnitude in the difference in values is important for performance management decision making. For example, it is common for employee disputes to occur over pay rises that are less than 5% (for example since the start of the financial crisis public sector workers in the UK have taken industrial action over far smaller pay rises). Hence a 20% difference pay would be considered significant. Moreover, if our model forecasts 20% lower payment required for staff then this would potentially provide significant savings for a company.

An additional advantage of $\tilde{Y}_t$ scenarios in Table 1 is that the scenarios are more reliable, that is we can have more confidence in these scenarios compared to Y_{η} scenarios. This is because $\tilde{Y}_t$ scenarios are produced by using more than set of (resampled) data and random cluster sampling. However, Y_{η} scenarios are produced from one set of data and a single SLR equation. Our scenarios are therefore produced over a greater range of regression data. Furthermore, in producing Y_{η} scenarios from one set of data the scenarios are more vulnerable to producing scenarios specific to a particular set of data, which may not be representative of the future. Consequently Y_{η} scenarios are less likely to be reliable scenarios.

In Table 2 we calculate the kth statistical moments associated with the scenario trees, using the scenarios in Table 1 for $\tilde{Y}_t$ and Y_{η}. We notice that our method produces significantly different moment values for $k = 1, 2, \ldots, 5$. Therefore a more robust scenario generation process provides different distribution properties. The different moment values not only affects the distribution of future values, which could affect the performance measurement decisions for monthly income, but also impact risk measurement of the distributions. The statistical moments are frequently used as a risk measure for statistical distributions, hence the change in moments implies a change in the risk profile of the distribution of pay. An organisation with a higher or lower risk of low pay would impact future performance management decisions.

In Table 3 we provide the empirical cumulative distribution functions $F(x)$ for Y_{η} and $\tilde{Y}_t$ generated scenarios. The graphs of $F(x)$ are also plotted in Figs. 3 and 4. As one can observe from the graphs, there is higher probability of lower monthly incomes in $\tilde{Y}_t$ compared to Y_{η}, particularly around the $15,000–20,000 range. The Y_{η} scenarios give a higher probability on higher monthly incomes in the range of $25,000

Table 2
Scenario tree kth moment values.

Moment Number k	$\tilde{Y}_t$ Scenario Tree	Y_t^{η} Scenario Tree
1	22243.48	24941.20
2	519.6×10^6	635×10^6
3	12.7×10^{12}	16.5×10^{12}
4	3.2×10^{17}	4.35×10^{17}
5	8.29×10^{21}	1.17×10^{22}

Table 3
Cumulative distribution function $F(x)$ for each scenario generation method.

F(x)	$\tilde{Y}_t$ Method	Y_t^{η} Method
0	0	0
0.1	15896.09	19530.94
0.2	16523.46	20458.31
0.3	16900.10	21015.06
0.4	18356.52	23167.92
0.5	19784.90	25279.33
0.6	24874.10	25730.62
0.7	25984.96	26915.25
0.8	26978.06	27974.29
0.9	27639.53	28679.68
1	29497.15	30660.66

onwards.

Our scenario generation method therefore provides a significantly different distribution function. In other words, a more robust scenario generation process changes the cumulative probabilities for different income values. The distribution of income in firms (and in economies in general) is a frequently investigated metric of employee conditions, particular with respect to fair pay (in fact after the start of the Global Financial Crisis the pay of Chief Executive Officers of banks with respect to junior staff was heavily reported in the media). Therefore our scenario generation method provides useful insight into such performance analyses.

5. Conclusion

Scenario generation is an important decision analysis tool and a fundamental aspect of Stochastic Programming. In this paper we have demonstrated a new method for scenario generation that is applicable to regression analysis and performance management. We have provided computational results on our method and benchmarked our performance against a standard scenario generation method, in the numerical experiments section. We have shown that our scenario generation method can produce representative scenarios, using cluster based sampling to ensure we sample relevant segments of the population data.

Secondly, our method is more robust to overfitting as we use multiple regression lines to produce scenarios. In using multiple regression lines, our scenario generation is more robust compared to using a single regression line, and so a more reliable scenario generation process. Therefore our scenario generation method should provide a cost saving to firms (either through their own stochastic programming optimisation method or for any forecast) and improving firm management.

Thirdly, our scenario generation method enables us to take into account modelling and expert opinion. This is achieved by adjusting the sampling process for the scenario generation method, for example incorporating worst case scenarios in the scenario generation. Finally, our scenario generation method is also particularly suitable for performance management applications, where such data exhibits clustering and they typically make use of regression modelling methods.

In terms of future work the paper could be extended to other performance management and measurement applications, for example other industry sectors or different operations of a business (such as marketing). In other operations of a business (such as sales) there are higher levels of uncertainty and therefore scenario generation may prove more useful to businesses. Additionally, it well known that some industries are far more unstable than other industries (for example the pharmaceutical industry is considered far more stable than the airline industry), hence scenario generation may offer more insight in such uncertain industries.

Secondly, we would also like to develop the paper to combine the scenario generation process with other scenario generation methods,

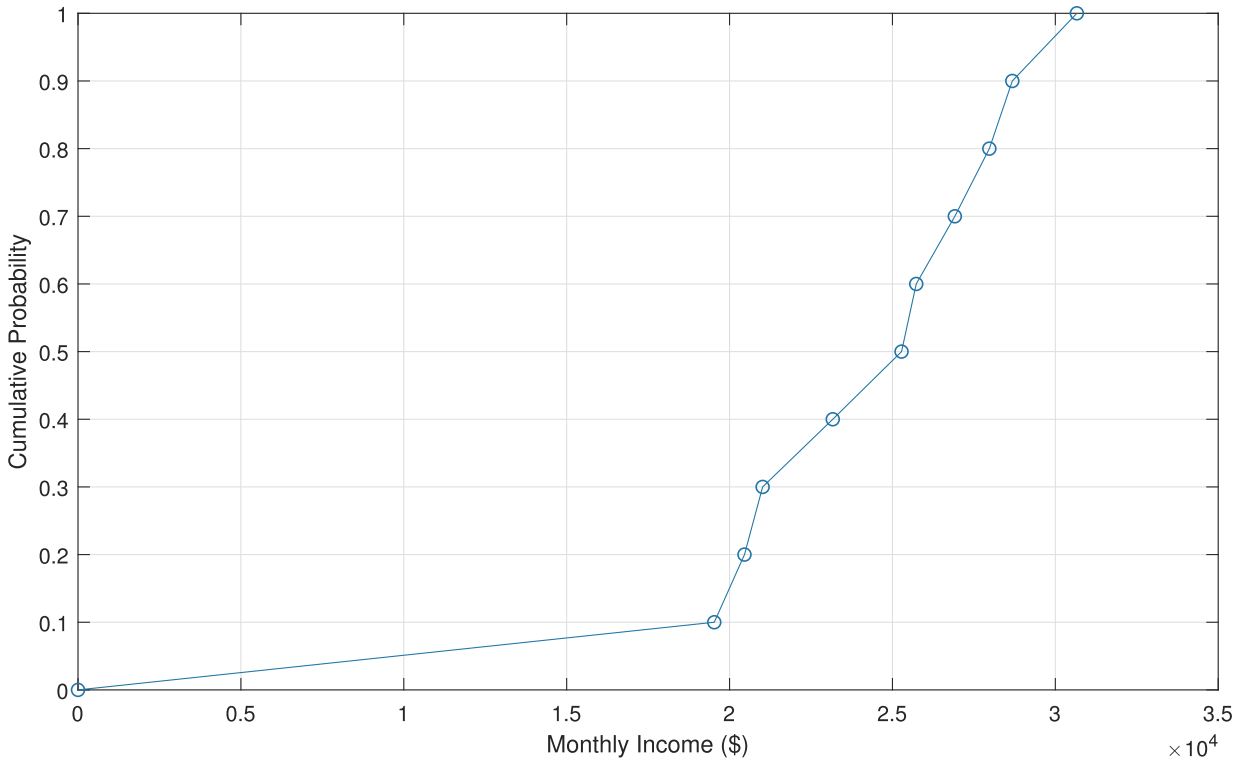

Fig. 3. Cumulative distribution function plot for Y_t^{η}.

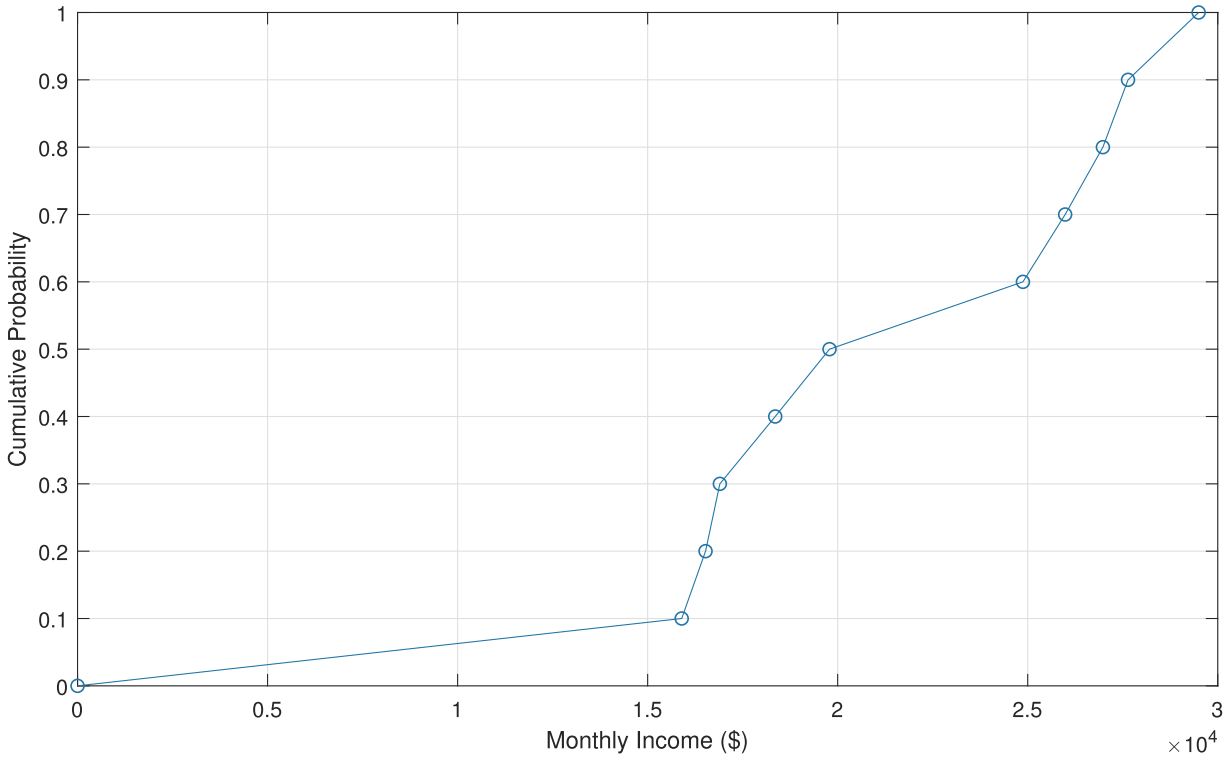

Fig. 4. Cumulative distribution function plot for $\tilde{Y}_t$.

such as moment matching. In combining scenario generation methods we can increase the number of properties we wish to incorporate within our scenario generating process, and therefore produce better quality scenarios. Finally, we would like to apply our scenario generation method to different stochastic programming optimisation models and see the impact of our scenario generation process. We would also like to investigate how our scenario generation method can be applied to similar models to regression, such as generalised linear models. Generalised linear models are another set of important statistical models (similar to simple linear regression but not as common) and so it would be useful to investigate if scenario generation from such models leads to improved performance analysis.

Appendix 1

Algorithm for numerical experiment:

1. Initialise number of iterations $N = 1$.
2. Obtain data $\{\mathbf{v}_1, \mathbf{v}_2, ...,\mathbf{v}_T\}$ containing T datapoints from performance dataset. The datapoint $\mathbf{v}_j = \{v_{j1}, v_{j2}\}$ for performance data, where we assume v_{j1} is the independent variable and v_{j2} is the dependent variable.
   The dataset can be raw data or cleaned data.
3. Assign the number of clusters m for data $\{\mathbf{v}_1, \mathbf{v}_2, ...,\mathbf{v}_T\}$.
   Choice of m can be set by decision maker, analyst or plotting the graph for performance data $\{\mathbf{v}_1, \mathbf{v}_2, ...,\mathbf{v}_T\}$. Alternatively, one can apply a quantitative metric.
4. Calculate centroid values $\{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_m\}$ for clusters $\phi_1, ..., \phi_m$, where $\mathbf{C}_j = \{C_{j1}, C_{j2}\}$ for performance data.
   If $N = 1$ then initialise $\{\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_m\}$ with random value from data $\{\mathbf{v}_1, \mathbf{v}_2, ...,\mathbf{v}_T\}$.
5. Assign $\mathbf{v}_j$ to cluster ϕ_i, $\forall\, j \in \{1, 2, ..., T\}$, $i \in \{1, 2, ...,m\}$.
   (a) Assign $\mathbf{v}_j$ where $j \in \{1, 2, ..., T\}$ to cluster ϕ_i, where $i \in \{1, 2, ..., m\}$, using objective function

   $$\min_{j \in 1,..,T} \Lambda(\|\mathbf{v}_j - \mathbf{C}_i\|),\ \forall\, i \in 1, 2, ..., m,$$

   where $\Lambda(.)$ is the specified distance measure:

   $$\Lambda(\|\mathbf{v}_j - \mathbf{C}_i\|) = \sqrt{(v_{j1} - C_{i1})^2 + (v_{j2} - C_{i2})^2}.$$

   (b) Increment N.
   (c) If $N > 2$ goto next step, otherwise goto step 4.
   (d) Calculate change in set membership for ϕ_i, $\forall\, i \in 1, 2, ..., m$, for iterations $(N - 1)$ and $(N - 2)$.
   (e) Goto step 6 if:
      (1) number of iterations $N > 2$, and no change in set membership for ϕ_i, $\forall\, i \in 1, 2, ..., m$, for iteration $(N - 1)$ and iteration $(N - 2)$; or
      (2) number of iterations $N > S$, where S is number of iterations limit.
      Otherwise goto step 4.
6. Stratify sample clustered data:
   (i) For each cluster ϕ_i for $i \in 1, 2, ..., m$ sort data by v_{j2}, $\forall j$.
   (ii) Stratify clusters into n strata by v_{j2}, $\forall j$, for each ϕ_i, $\forall\, i \in 1, 2, ..., m$, and sample each strata where Θ_i is the set of data samples for strata i.
7. Calibrate regression model: for each simple linear regression model Y_t^i where $i = 1, ..., n$ and $Y_t^i = \beta_0^i + \beta_1^i x_t + \epsilon_t^i$, fit Y_t^i to Θ_i.
8. Generate scenarios at time $t = \tau$ by taking samples from distributions Y_τ^i, for $i = 1, ..., n$.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.orp.2018.100095

Conflict of interest

The authors have no conflict of interest.

References

[1] April J, Glover F, Kelly J, Laguna M. Simulation-based optimization. In: Proceedings of the 35th conference on winter simulation 2003;1(1):71–8.
[2] Arumugam V, Antony J, Linderman K. The influence of challenging goals and structured method on six sigma project performance: a mediated moderation analysis. Eur J Oper Res 2016;254(1):202–13. https://doi.org/10.1016/j.ejor.2016.03.022.
[3] Beraldi P, Bruni ME. A clustering approach for scenario tree reduction: an application to a stochastic programming portfolio optimization problem. Top 2013;22(3):934–49. https://doi.org/10.1007/s11750-013-0305-9.
[4] Bhattacharya A, Mohapatra P, Kumar V, Dey PK, Brady M, Tiwari MK, et al. Green supply chain performance measurement using fuzzy ANP-based balanced scorecard: a collaborative decision-making approach. Prod Plann Control 2013;25(8):698–714. https://doi.org/10.1080/09537287.2013.798088.
[5] Birge J, Louveaux F. Introduction to stochastic programming. Springer New York; 1997.
[6] Bollerslev T. Generalized autoregressive conditional heteroskedasticity. J Econ 1986;31(3):307–27.
[7] Buhlmann P. Sieve bootstrap for time series. Bernoulli 1997;3(2):123–48.
[8] Carino D, Kent T, Myers D, Stacy C, Sylvanus M, Turner A, et al. The russell-Yasuda kasai model: an asset/liability model for a japanese insurance company using multistage stochastic programming. Interfaces (Providence) 1994:29–49.
[9] Demirel O, Willemain T. Generation of simulation input scenarios using bootstrap methods. J Oper Res Soc 2002:69–78.
[10] Dempster M, Thorlacius A. Stochastic simulation of international economic variables and asset returns: the falcon asset model. In: Proceedings of the 8th International AFIR Colloquium 1998;90(1):29–45.
[11] Dupacova J. Scenario based stochastic programs: resistance with respect to sample. Ann Oper Res 1996;64:21–38.
[12] Dupačová J, Gröwe-Kuska N, Römisch W. Scenario reduction in stochastic programming. Math Prog 2003;95(3):493–511.
[13] Ermoliev Y, Wets RJ-B. Numerical techniques for stochastic optimization. Springer-Verlag; 1988.
[14] Fleten S, Kristoffersen T. Stochastic programming for optimizing bidding strategies of a nordic hydropower producer. Eur J Oper Res 2007;181(2):916–28.
[15] Fleten S, Kristoffersen T. Short-term hydropower production planning by stochastic programming. Comput Oper Res 2008;35(8):2656–71.
[16] Gibbons R, Kaplan RS. Formal measures in informal management: can a balanced scorecard change a culture? Am Econ Rev 2015;105(5):447–51. https://doi.org/10.1257/aer.p20151073.
[17] Glasserman P. Monte carlo methods in financial engineering. Springer, New York; 2004.
[18] Gülpınar N, Rustem B, Settergren R. Simulation and optimization approaches to scenario tree generation. J Econ Dyn Control 2004;28(7):1291–315.
[19] Hansen JV, Mcdonald JB, Turley RS. Partially adaptive robust estimation of regression models and applications. Eur J Oper Res 2006;170(1):132–43. https://doi.org/10.1016/j.ejor.2004.06.008.
[20] Heitsch H, Römisch W. Scenario reduction algorithms in stochastic programming. Comput Optim Appl 2003;24(2):187–206.
[21] Hoyland K, Wallace S. Generating scenario trees for multistage decision problems. Manag Sci 2001;47(2):295–307.
[22] Jobst N, Zenios S. Tracking bond indices in an integrated market and credit risk environment. Quant Financ 2003;3(2):117–35.
[23] Kaut M, Wallace S. Evaluation of scenario-generation methods for stochastic programming. Stoch Program E-Print Ser 2003;14(1).
[24] Klaassen P. Discretized reality and spurious profits in stochastic programming

models for asset/liability management. Eur J Oper Res 1997;101(2):374–92.
[25] Kouwenberg R, Zenios S. Stochastic programming models for asset liability management. Handbook of asset and liability management 1. Elsevier; 2001. p. 1–71.
[26] Krasko V, Rebennack S. Two-stage stochastic mixed-integer nonlinear programming model for post-wildfire debris flow hazard management: mitigation and emergency evacuation. Eur J Oper Res 2017;263(1):265–82. https://doi.org/10.1016/j.ejor.2017.05.004.
[27] Kyriakidis P. Sequential spatial simulation using latin hypercube sampling. Quant Geol Geostat 2008;14(1):65–74.
[28] Lacerda RTDO, Ensslin L, Ensslin SR, Knoff L, Junior CMD. Research opportunities in business process management and performance measurement from a constructivist view. Knowl Process Manag 2015;23(1):18–30. https://doi.org/10.1002/kpm.1495.
[29] Lepage R, Billard L. Exploring the limits of bootstrap. Wiley Series; 1992.
[30] Lurie P, Goldberg M. An approximate method for sampling correlated random variables from partially-Specified distributions. Manag Sci 1998;44(2):203–18.
[31] Micheli P, Mura M. Executing strategy through comprehensive performance measurement systems. Int J Oper Prod Manag 2017;37(4):423–43. https://doi.org/10.1108/ijopm-08-2015-0472.
[32] Minasny B, McBratney A. A conditioned latin hypercube method for sampling in the presence of ancillary information. Comput Geosci 2006;32(9):1378–88.
[33] Mulvey J, Gould G, Morgan C. An asset and liability management system for towers Perrin-Tillinghast. Interfaces (Providence) 2000;30(1):96–114.
[34] Pulakos ED, Hanson RM, Arad S, Moye N. Performance management can be fixed: an on-the-job experiential learning approach for complex behavior change. Ind Organ Psychol 2015;8(01):51–76. https://doi.org/10.1017/iop.2014.2.
[35] Restrepo MI, Gendron B, Rousseau L-M. A two-stage stochastic programming approach for multi-activity tour scheduling. Eur J Oper Res 2017;262(2):620–35. https://doi.org/10.1016/j.ejor.2017.04.055.
[36] Richards G, Yeoh W, Chong AYL, Popovic A. Business intelligence effectiveness and corporate performance management: an empirical analysis. J Comput Inf Syst 2017:1–9. https://doi.org/10.1080/08874417.2017.1334244.
[37] Ridier A, Chaib K, Roussy C. A dynamic stochastic programming model of crop rotation choice to test the adoption of long rotation under price and production risks. Eur J Oper Res 2016;252(1):270–9. https://doi.org/10.1016/j.ejor.2015.12.025.
[38] Sagaert YR, Aghezzaf E-H, Kourentzes N, Desmet B. Tactical sales forecasting using a very large set of macroeconomic indicators. Eur J Oper Res 2018;264(2):558–69. https://doi.org/10.1016/j.ejor.2017.06.054.
[39] Samuelson P. Proof that properly anticipated prices fluctuate randomly. Ind Manag Rev 1965;6(2):41–9.
[40] Smith J. Moment methods for decision analysis. Manag Sci 1993;39(3):340–58.
[41] Speklé RF, Verbeeten FH. The use of performance measurement systems in the public sector: effects on performance. Manag Account Res 2014;25(2):131–46.
[42] Wang S, Huang G. A multi-level Taguchi-factorial two-stage stochastic programming approach for characterization of parameter uncertainties and their interactions: an application to water resources management. Eur J Oper Res 2015;240(2):572–81. https://doi.org/10.1016/j.ejor.2014.07.011.
[43] Ziemba W. The stochastic programming approach to asset, liability, and wealth management. CFA Institute USA; 2003.