

Junike, Gero; Flaig, Solveig; Werner, Ralf

Article — Published Version

Validation of machine learning based scenario generators

Journal of Risk and Insurance

Suggested Citation: Junike, Gero; Flaig, Solveig; Werner, Ralf (2026) : Validation of machine learning based scenario generators, Journal of Risk and Insurance, ISSN 1539-6975, Vol. 93, Iss. 2, pp. 508-533,
<https://doi.org/10.1111/jori.70040>

This Version is available at:

<https://hdl.handle.net/10419/342740>

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.



<http://creativecommons.org/licenses/by/4.0/>

Validation of machine learning based scenario generators

Gero Junike¹  | Solveig Flaig² | Ralf Werner³ 

¹Department of Mathematics, Ludwig-Maximilians-Universität München, München, Germany

²Kapitalanlage-Controlling, Deutsche Rückversicherung AG, Hansaallee 177, Düsseldorf 40549, Germany

³Institut für Mathematik, Universität Augsburg, Augsburg 86159, Germany

Correspondence

Gero Junike, Ludwig-Maximilians-Universität München, Department of Mathematics.

Email: gero.junike@math.lmu.de

Funding information

Deutsche Rueckversicherung AG

Abstract

Machine learning (ML) methods are becoming increasingly important for designing economic scenario generators for internal models. Validating data-driven models requires different methods than validating classical, theory-based models. We discuss two novel aspects of such validation: first, checking the multivariate distribution of risk factors, and second, detecting unwanted memorization effects. The first task is necessary because, in ML-based methods, dependencies are driven by data rather than derived from a financial-mathematical theory. To address this first issue, we propose using an existing test from the literature. The second task is necessary because it cannot be ruled out that ML-based models merely reproduce empirical data rather than generating new scenarios. For the second issue, we introduce a novel memorization ratio together with a thorough discussion. We include numerical experiments based on real market data and validate a simple autoencoder-based scenario generator.

KEYWORDS

economic scenario generator, machine learning, memorization ratio, nearest neighbor distance, Solvency 2

JEL CLASSIFICATION

C14, C45, C63, G22

This is an open access article under the terms of the [Creative Commons Attribution](https://creativecommons.org/licenses/by/4.0/) License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

© 2026 The Author(s). *Journal of Risk and Insurance* published by Wiley Periodicals LLC on behalf of American Risk and Insurance Association.

1 | INTRODUCTION

Under Solvency 2, insurance companies are required to calculate a Solvency Capital Requirement, that is, the value-at-risk at the 99.5% level of their own funds over a 1-year horizon. This is usually done by generating suitable scenarios using an economic scenario generator (ESG) and taking the empirical 99.5% percentile of the loss.

An introduction to ESGs from an actuarial point of view is for instance provided by Pedersen et al. (2016) or Varnell (2011). Current internal models for market and non-life underwriting risk often use Monte-Carlo simulation techniques to derive the risk of the (sub) modules and then use correlations or copulas for aggregation, see Pfeifer and Ragulina (2018). Alternatively, in a historical simulation, similar to bootstrapping, empirical data is simply copied to “generate” future scenarios, see Hendricks (1996) and Beder (1995) for an analysis of the empirical performance of such simulation. A critical view on these kind of methods is for example due to Pritsker (2006): in particular, it is not clear how to choose the appropriate length of the historical sample to accurately compute the value-at-risk.

Studies that have looked at deriving economic scenarios using neural networks have gained interest over the last years, cf. Wiese et al. (2020) for a recent contribution. Some of these studies, see for instance the two GAN-based studies Buch et al. (2023) or Flaig and Junike (2022) and the references therein, also consider value-at-risk calculation and come broadly to the conclusion that such approaches perform as good as regulatory-approved methods. A detailed comparison of a variational autoencoder (VAE) approach with 12 other approaches is for example provided by Arian et al. (2022), showing that the VAE approach works indeed very well. In summary, it can be concluded that both GAN-based and autoencoder-based approaches can be considered as appropriate ESGs. Regulators in several countries, too, have authored papers that address considerations associated with the use of ML methods in internal models. For example, in 2019, the Nederlandsche Bank issued a paper discussing how artificial intelligence is currently used in finance; see van der Burgt (2019). There, the authors state that future ML applications will “use broader and better data to develop predictive risk models” and will “increase the precision of risk assessment,” see van der Burgt (2019, pp. 28 and 29). The German regulators, Bundesbank and BaFin, assert that “the use of ML methods can help to quantify risks more accurately and enhance process quality, thereby improving financial firms’ risk management”; nonetheless, these authors also note that “the limited transparency of the model’s behavior has consequences for the [...] model validation”, see BaFin (2021, p. 3 and p. 7).

One of the key processes for internal models in insurance companies is validation, see European Union (2009, Art. 124). However, it is not appropriate to limit the validation of ML-based models to the methods used for classical financial-mathematical models. In this context, the German regulator, BaFin (2021, p. 11), warns that the validation for an internal model can basically remain the same as for classical ESGs, but any special features of the ML-based approach must additionally be taken into account and validated. Therefore, it is important to determine what additional validation measures need to be undertaken in order to prove the validity of a given model when the scenario generator is ML-based.

In classical internal market risk models, the dependencies between the risk factors (interest rates, equities, foreign exchange) are modeled using financial mathematical tools such as correlations or copulas, see Sandström (2016) or Pfeifer and Ragulina (2018). In contrast, ML-based methods induce dependencies based on the data, eliminating the need for specific dependency modeling. Therefore, ML-based methods require additional validation.

So far, studies that have examined the use of ML methods to generate financial scenarios, validation is carried out primarily by visual means or using only a few statistical parameters, as is the case in Ni et al. (2021) and Wiese et al. (2020). Recently, in a more quantitative way, Cont et al. (2025) propose an evaluation approach that focuses on the tail of the marginal distributions.

With ML-based models at least two new important issues have arisen which must be addressed during the model validation process:

- (i) Since in ML-based models the dependencies are not explicitly modeled, an additional validation has to be performed to show that the dependencies (or, more generally, the multivariate distribution) generated by the model resemble the empirical multivariate distribution including the empirical dependencies.
- (ii) Another issue associated with generative ML models is the so-called *memorization effect*. In this phenomenon, instead of generating new scenarios for risk management purposes, the ESG model simply replicates the empirical data used in the learning process. It is well known that neural networks can memorize some training data, see Wei et al. (2025) and references therein. Arora et al. (2018), for example, state that one of the most obvious pitfalls associated with a GAN training is that a GAN may simply memorize the training data. Theoretical investigations concerning memorization in GANs can be found in Nagarajan et al. (2018). A similar statement about memorization holds true for auto-encoders as well, see Radhakrishnan et al. (2018). Both types have already been applied as ESGs in the literature.

The literature on estimating the memorization effect in generative methods focuses mainly on large language models, see Wei et al. (2025) and image generation, see Borji (2019) and Borji (2022) for survey papers. In particular, Lopez-Paz and Oquab (2016) and Xu et al. (2018) propose a 1-nearest neighbor classifier to measure memorization. We follow a similar approach to define a memorization ratio. Bai et al. (2021) propose a memorization-informed Fréchet inception distance with memorization penalty to evaluate GANs based on cosine similarity. However, the proposed measures of the memorization effect are mainly heuristically motivated metrics which focus on image generation. In contrast to these papers, we provide a theoretical analysis and investigate the asymptotic behavior of our memorization ratio as the sample size approaches infinity. Meehan et al. (2020) and van den Burg and Williams (2021) propose a probabilistic approach to measure memorization. Unlike these approaches, our memorization measure does not require data to be withheld for validation. This means that the amount of training data is not reduced, which is particularly favorable in applications where the training data is scarce.

In the following, we suggest two approaches how to deal with issues (i) and (ii): To address point (i), we propose to use a test based on nearest neighbor distances, as studied by Mondal et al. (2015), which tests the equality of two multivariate distributions. In particular, the test covers both dependencies and marginal distributions at the same time. For further reading on alternative tests based on nearest neighbor distances, see Bickel and Breiman (1983); Henze (1988); Schilling (1986); Weiss (1960), and Ebner et al. (2018). To address point (ii), we develop a new measure called *memorization ratio* to capture the memorization effect. We classify an empirical data point as memorized if a generated data point lies in an unusually small neighborhood around that empirical data point. Our memorization ratio is easy to implement and to interpret. We show that the memorization ratio converges to a dimension-independent constant when the amount of data increases.

We test both measures on simulated and real market data. In particular, our experiments include a real-world example: we use an autoencoder-based method to generate interest rate scenarios in a high-dimensional space. The training data consists of a term structure interest rate time series from 2000 to 2022. The autoencoder-based method is validated using a nearest neighbor test and our new memorization ratio, demonstrating that the memorization ratio is indeed effective. We further provide in-sample and out-of-sample validation for our data sets. In particular, out-of-sample validation is a common technique in machine learning to estimate the generalization error, see for instance Goodfellow et al. (2016, Sec. 8), or Gu et al. (2021) who apply out-of-sample validation for autoencoders in a financial context. For our numerical experiments, we chose an effective yet simple autoencoder-based setup for presentation purposes.¹ Although more sophisticated methods, such as GANs or VAEs, might produce better results, their validation would remain the same. More sophisticated methods, especially for the construction of yield curve scenarios, include for example Richman and Scognamiglio (2024) or Horvath et al. (2026).

The remaining paper is structured as follows: In Section 2, we introduce the two proposed validation techniques and in Section 3, we show how these techniques can be used for model validation. We consider a one-dimensional real market dataset (S&P 500) for illustrative purposes. We also consider an autoencoder trained on real market data to generate a high-dimensional interest rate term structure. Section 4 concludes. In Appendix B, we give a brief introduction to neural networks and autoencoders.

2 | VALIDATION TECHNIQUES FOR MACHINE LEARNING BASED SCENARIO GENERATORS

During the usual validation cycle, insurers already investigate a variety of statistical properties of a scenario generator. For example, stability and sensitivities are typically already part of established validation procedures. Additionally, forecasting validation could be included. However, within an ML-based approach, some additional validation is required as we have already pointed out in the introduction: Since the multivariate distribution of the risk factors is not derived from a financial mathematical model, it has to be verified that the generator only produces samples from the same (or at least very similar) distribution as implied by the data sample. Further, since a pure resampling method is not intended by an ML-based generator, it is necessary to analyze for potential memorization effects.

Here, we suggest a purely data-driven nonparametric evaluation approach, because the nonparametric nature is one of the main advantages of using ML-based generators. Such methods which can be applied to any model are generally referred to as *model agnostic*, see for example, Borji (2019, p. 4). Furthermore, we prefer quantitative measures which are easily interpretable and computationally not too expensive, thus fostering their use in validation purposes in a Solvency 2 regime. For this purpose, we split the recommended validation techniques into two components:

- (1) the alignment of the multivariate distributions in the given empirical data and in the generated data, and

¹At this point, we would like to clarify that the use of autoencoders as ESGs is not considered to be a main contribution of this paper.

(2) the memorization effect.

Measures for these two categories are discussed in the subsequent Sections 2.1 and 2.2. Like any other test statistics, these kind of methods could be used not only for validation, but also for hyperparameter optimization of the ML models. Of course, additional validation, for example, of marginal tail behavior, could be carried out using the quantitative measures based on value-at-risk and expected shortfall as proposed in Cont et al. (2025, p. 21) or by comparison to benchmark portfolios as in Flaig and Junike (2022, Chapter 3).

For the mathematical discussion in the remainder of this paper, we rely on the following notation: Let $(\Omega, \mathcal{F}, P)$ be a probability space and $d \in \mathbb{N}$ the number of risk factors that are modeled. Further, let $E : \Omega \rightarrow \mathbb{R}^d$ denote a random vector describing the empirical data. The data generated by some ML-based generator is denoted by the random vector $G : \Omega \rightarrow \mathbb{R}^d$. Finally, let $E_1, \dots, E_M$, and $G_1, \dots, G_N$ be independent copies of the random vector E and G , respectively. Since the consideration of memorization effects is mainly important for continuous distributions, we make the following main assumption:

Assumption H0. The random variables $E_1, \dots, E_M, G_1, \dots, G_N$ are independent and identically distributed and have a piecewise continuous and bounded probability density function.

2.1 | Measure for the alignment of the multivariate distributions: Nearest neighbor coincidence

The test on the alignment of the multivariate distribution including dependency and marginal distributions of the empirical and the generated data can be reformulated as a multivariate two-sample test where we want to measure the equality of two multivariate distributions based on two sets of independent observations. For this purpose, and in alignment with the above considerations, we prefer the *nearest neighbor coincidence measure* as defined by Schilling (1986) and as further developed by Mondal et al. (2015). This test statistic is rather easy to interpret and to implement, computationally cheaper than alternative methods (e.g., based on optimal transport) and the numerical experiments by Mondal et al. (2015) are very promising. Of course, alternative possibilities for checking equality of distributions are available, for example, maximum mean discrepancy, average log-likelihood, F1-score, and many more, see for instance Borji (2019).

For the definition of the nearest neighbor coincidence let us consider the indicator function $\delta_{E_m}(r)$ which takes the value 1 if the r -nearest neighbor (measured by the Euclidean distance) of E_m within the set $\{E_1, \dots, E_M, G_1, \dots, G_N\} \setminus \{E_m\}$ is an empirical data point and 0 if it is a generated data point. Accordingly, the indicator function $\delta_{G_n}(r)$ is 1 if the r -nearest neighbor is a generated data point and 0 otherwise. As in Mondal et al. (2015), we define for $k \in \mathbb{N}$:

$$T_{NN1,k} = \frac{M \left| T_{E,k} - \frac{M-1}{N+M-1} \right| + N \left| T_{G,k} - \frac{N-1}{M+N-1} \right|}{N+M}, \quad (1)$$

where

$$T_{E,k} = \frac{1}{Mk} \sum_{m=1}^M \sum_{r=1}^k \delta_{E_m}(r) \quad \text{and} \quad T_{G,k} = \frac{1}{Nk} \sum_{n=1}^N \sum_{r=1}^k \delta_{G_n}(r).$$

The test statistic $T_{E,k}$ (the test statistic $T_{G,k}$) counts for each empirical (generated) data point the number $r \leq k$ of neighbors which are also empirical (generated) data points and takes the average. Under Assumption H0, $T_{NN1,k}$ converges to zero in probability, see proof of Theorem 5 in Mondal et al. (2015). If the empirical data points and the generated data points are rather distinct from each other (i.e., stem from different distributions), $T_{NN1,k}$ should take larger values due to lack of full mixing of the two samples. In particular, for $M = N$ and if the generated data is far off the empirical data, $T_{E,k}$ and $T_{G,k}$ should be both close to their maximum value 1 and $T_{NN1,k}$ should be close to $\frac{1}{2}$.

Remark 1. Mondal et al. (2015) prove that $\sqrt{N+M} \cdot T_{NN1,k}$ is asymptotically distributed as a sum of two correlated half normals if Assumption H0 holds and if N and M grow to infinity with $\frac{M}{N} \rightarrow \alpha$ for some $\alpha > 0$. It is further shown that under Assumption H0, the expectation of $T_{E,k}$ is equal to $\frac{M-1}{N+M-1}$ and the expectation of $T_{G,k}$ is equal to $\frac{N-1}{M+N-1}$.

Remark 2. To compute $T_{NN1,k}$, we have to find the minimum entry of a vector of length $M+N$ exactly M respectively N times, to obtain $T_{E,k}$, resp. $T_{G,k}$. The computational effort for the computation of $T_{NN1,k}$ is therefore at most $O((N+M)^2)$.

An implementation of $T_{NN1,k}$ can be found in Appendix D, where we also explain how to treat ties when sorting distances.

2.2 | Measure for the detection of memorization: Memorization ratio

The statistic $T_{NN1,k}$ will lead to very good scores if the generated data are drawn with repetition from the empirical ones (bootstrapping). From a risk management perspective, this is not the optimal result because the main intention of employing a generative ML-based generator is to create new scenarios that could happen rather than memorizing scenarios that have actually taken place, see Chen et al. (2018, p. 2). Bai et al. (2021, p. 1) emphasize that “unintentional memorization is a serious and common issue in popular generative models”. Therefore, we need some (preferably interpretable) quantitative measure to detect whether generated data points match empirical ones or not.

In a multi-dimensional space, it is highly unlikely for generated and empirical data points to match exactly, so we classify an empirical data point as being memorized if a generated data point lies in “an unusual small neighborhood” around this empirical data point.

Definition 1. Let $\rho \in (0, 1]$. The *memorization ratio* $\Pi_{M,N}^\rho$ is defined as:

$$\Pi_{M,N}^\rho = \frac{1}{M} \sum_{m=1}^M \mathbf{1}_{\left[0, \rho \frac{1}{d} R_m\right)} \left(\min_{n=1, \dots, N} \|G_n - E_m\|_2 \right) \quad (2)$$

where

$$R_m := \min_{m' \neq m} \|E_m - E_{m'}\|_2, m = 1, \dots, M. \quad (3)$$

The random variable R_m denotes the distance between E_m and its nearest empirical neighbor. An Euclidean ball of radius $\rho^{\frac{1}{d}} R_m$ has a volume equal to a fraction ρ of the volume of a ball of radius R_m . An empirical data point E_m is considered *memorized* if there is at least one generated data point inside the ball of radius $\rho^{\frac{1}{d}} R_m$ around E_m . This is visualized in Figure 1. The memorization ratio equals the percentage of elements $E_1, \dots, E_M$ which are memorized. If all empirical data is memorized, then $\Pi_{M,N}^\rho = 1$. If no empirical data is memorized, then $\Pi_{M,N}^\rho = 0$. An implementation of $\Pi_{M,N}^\rho$ can be found in Appendix D.

Remark 3. The parameter ρ allows to control the interpretation of memorization: The smaller ρ , the less likely it is that an empirical data point is deemed memorized. In the extreme case that ρ approaches zero, an empirical data point will only be considered memorized if there is an identical generated data point. This happens, for example, when the empirical data is simply copied, for example, during bootstrapping. The greater ρ , the more likely it is that an empirical data point is considered memorized. The special case $\rho = 1$ is related to the measure “1-NN accuray (real)” in Xu et al. (2018, p. 3). Most importantly, our approach is more general and backed by mathematical results: for instance, the subsequent Theorem 1 establishes convergence of the memorization ratio $\Pi_{M,N}^\rho$ to a dimension-independent limit.

Theorem 1. Let $\alpha > 0$ and $\rho \in (0, 1]$. Under Assumption H0, it holds in the mean square sense that

$$\lim_{M, N \rightarrow \infty, \frac{M}{N} = \alpha} \Pi_{M,N}^\rho = \frac{\rho}{\rho + \alpha}.$$

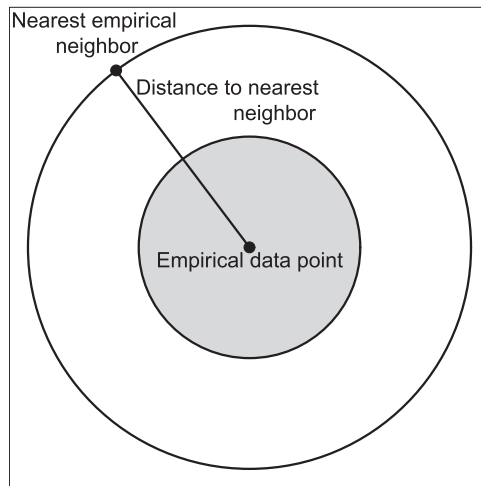


FIGURE 1 The empirical data point is considered memorized if there is at least one generated point within the gray area. The gray areas corresponds to a ball of radius $\rho^{\frac{1}{d}}$ times the distance to the nearest empirical neighbor. We have $d = 2$ and $\rho = 0.25$. The gray area corresponds to 25% of the whole area of the circle.

Proof. The proof can be found in Appendix A. □

Remark 4. According to Theorem 2 of Weiss (1960), the limit $\frac{\rho}{\rho + \alpha}$ of the memorization ratio under H0 equals the asymptotic probability that an arbitrarily empirical data point is memorized.

The convergence of the memorization ratio to its theoretical limit stated in Theorem 1 can also be observed empirically in Table 1 for normal distributed as well as non-normal distributed random variables. We see that the memorization ratio approaches its theoretical limit already for relatively small values of M and N .

The benefit of the memorization ratio can be visualized in Figure 2. Some empirical data is compared to two different sets of generated data. Both generated data sets are fine according to the statistic $T_{NN1,k}$, that is, are similarly distributed as the empirical data. However, one generated data set leads to a high memorization ratio and can thus be ruled out. In other words, Figure 2 illustrates how the memorization ratio helps to distinguish between generators with and without memorization effect.

Remark 5. If a generated data point $G = (G^1, ..., G^d)$ is identical to an empirical data point $E = (E^1, ..., E^d)$ in all dimensions except, say, the first dimension, it depends on the application whether E should be deemed “memorized” or not. Choosing the Euclidean distance as in Definition 1, we obtain

$$\|G - E\|_2 = |G^1 - E^1|$$

and E would usually *not* be considered as memorized if G^1 and E^1 are different enough. In some applications, this might not be desirable, that is, E should be deemed memorized

TABLE 1 Convergence of the memorization ratio for different distributions for the empirical and generated data sets: Consider four independent random variables: let X_1, X_2 be standard normal distributed, X_3 exponentially distributed with rate one and X_4 Cauchy distributed with location zero and scale one. Let $E_1, ..., E_M, G_1, ..., G_N$ be independent copies of $(X_1, \gamma X_1 + \sqrt{1 - \gamma^2} X_2)$ with correlation $\gamma = 0.75$ for the column “Two corr. Gaussian.” Let them be independent copies of (X_3, X_4) for the column “Exp. and Cauchy.” For the column “20 uniform,” let them be independent copies of 20 independent, uniform distributed random variables. We show the memorization ratio for these distributions for different values of M, N and ρ and compare the memorization ratio to its theoretical limit. We average the memorization ratio over 100 simulations. Each entry in the table comes with a standard error of at most ± 0.006 . The values in the table hardly change choosing the norm $\|\cdot\|$ for the memorization ratio as defined in Remark 5.

	ρ	Two corr. Gaussian	Exp. and Cauchy	20 uniform	Theoretical limit
$M = 100, N = 100$	0.25	0.200	0.207	0.275	0.200
$M = 100, N = 400$	0.25	0.496	0.506	0.598	0.500
$M = 200, N = 100$	0.25	0.109	0.114	0.159	0.111
$M = 100, N = 100$	0.5	0.333	0.338	0.392	0.333
$M = 100, N = 400$	0.5	0.661	0.676	0.706	0.667
$M = 200, N = 100$	0.5	0.198	0.205	0.234	0.200

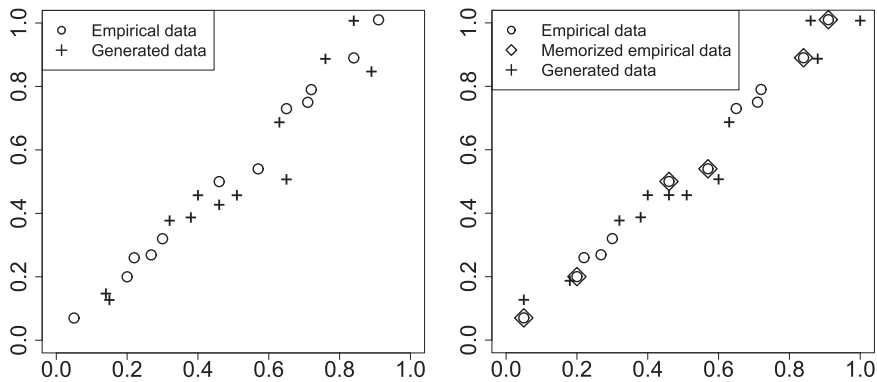


FIGURE 2 Example for the benefit of the memorization ratio. Some empirical data is compared to different generated data sets: on the left: $T_{NN1,k} = 0.02$, $\Pi_{M,N}^0 = 0$; on the right: $T_{NN1,k} = 0.02$, $\Pi_{M,N}^0 = 0.5$, where $k = 3$ and $\rho = 0.25$. The empirical dataset is the same in both boxes. However, half of the generated data points on the right are memorized.

since it is almost indistinguishable from G . One solution would be to choose another metric for the memorization ratio, for example, based on the norm

$$\|x\| := \underbrace{\frac{2}{\sqrt{\pi}} \left(\frac{\Gamma\left(\frac{d}{2} + 1\right)}{d!} \right)^{\frac{1}{d}}}_{=: \beta(d)} (|x^1| + \dots + |x^d|), x \in \mathbb{R}^d.$$

A unit ball under $\|\cdot\|$ has the same volume as a unit ball under $\|\cdot\|_2$. Since $\beta(d) \rightarrow 0$ for $d \rightarrow \infty$, we then have

$$\|G - E\| = \beta(d)|G^1 - E^1| \rightarrow 0, d \rightarrow \infty$$

and E is always deemed memorized if d is large enough. This is also confirmed numerically in Table 2. In other words, Table 2 shows how the choice of norm affects the memorization ratio.

Remark 6. Motivated by a reviewer's remark, we comment on a potential generalization of Theorem 1: the convergence result should also hold for identically distributed E_m with a finite range dependence (i.e., all E_{m+k} are independent of E_m for $k \geq K$); thus allowing for finitely overlapping estimation windows in time series. Section C in the appendix indicates that such a statement should hold true, but we leave a potentially more technically involved proof for future research.

Remark 7. Denote by P_E the distribution of E and by P_G the distribution of G . By P_E^M and P_G^N we denote the empirical distribution of an empirical sample of size M and a generated sample of size N . Arora et al. (2017) and Cont et al. (2025) define the generalization error of P_G by

$$\left| d(P_E^M, P_G^N) - d(P_E, P_G) \right|, \quad (4)$$

TABLE 2 We apply the memorization ratio (MR) with different norms to the following empirical and generated data sets: Take $M = N = 1000$, $E_1, \dots, E_M$ are independent and multivariate normally distributed in $\mathbb{R}^d$ with mean zero and covariance matrix equal to the identity. Set G_1 independent of $E_1, \dots, E_M$ but with mean equal to $(8, \dots, 8)$ and covariance matrix equal to the identity and $G_i := E_i, i = 2, \dots, M$. The memorization ratio (MR) is obtained for $\rho = 0.25$ and using the norm $\|\cdot\|$ and $\|\cdot\|_2$. The entries in the table come with a standard error of at most ± 0.0007 .

d	MR with $\ \cdot\ $	MR with $\ \cdot\ _2$
10	0.013	0.0
20	0.97	0.002
30	1.0	0.017

where d represents a divergence, for example, the Wasserstein distance, quantile divergence, the neural net distance, so forth, see Arora et al. (2017); Cont et al. (2025) and references therein. A large probability of a small generalization error means that the generator has a similar performance with the empirical distributions and the true distributions, that is, generalizes well from the training dataset. However, this definition does not capture the memorization effect: Assume P_E and P_G are one-dimensional standard normal distributions. Consider some empirical data of size M distributed according to P_E and generate data by simply bootstrapping (draw M times independently with repetition) from the empirical data. If d is the Wasserstein distance, the generalization error of P_G converges to zero in probability for $M \rightarrow \infty$ but the training sample is memorized since bootstrapping does not generate new, unseen samples.

3 | NUMERICAL EXPERIMENTS

In this section, consider two different setups: in Section 3.1, we apply the concepts to one-dimensional real market data for illustrating purpose. In Section 3.2, we investigate in detail a purely data-driven setup and deal with high-dimensional, real-market interest rate data and autoencoders.

3.1 | Illustrating one-dimensional example

In this section, we show in a simple setting how the memorization ratio and the test statistic $T_{NN1,k}$ can be used to evaluate various approaches to generate new samples from a training set. We consider a training set consisting of the yearly log-returns of the S&P 500 from 1997 to 2011 and a test set of the years 2012 to 2023. We discuss several ways to generate new samples from the training data, that is, bootstrapping, a parametric approach and a ML-based method. Thanks to the memorization ratio and statistic $T_{NN1,k}$, we can rule out certain data generation processes. We consider the following data generation techniques:

- **Bootstrapping:** to generate new samples, we simply draw independently with repetition from the training set (with $N = M$). The probability that a scenario from the training set shows up

in the bootstrap-sample, and is hence memorized, is $1 - \left(1 - \frac{1}{M}\right)^M \approx 1 - e^{-1} = 0.63$, cf. Table 3.

- Parametric model: we use the normal distribution with mean and standard deviation equal to the empirical mean and standard deviation of the training dataset to generate new samples.
- Data-driven with kernel-smoothing: we calibrate a kernel-smoothed cumulative distribution function with Gaussian kernel with global bandwidth $h > 0$ to the training data and use that distribution to generate new samples via the inversion transform. We define the CDF

$$F_h(x) = \frac{1}{M} \sum_{m=1}^M \Phi\left(\frac{x - E_m}{h}\right), x \in \mathbb{R},$$

where Φ is the CDF of a standard normal random variable and $E_1, \dots, E_M$ is the training data. To generate new data, uniformly random numbers are passed to the inverse of F_h .

The empirical CDF of the training set and the CDFs of the various data generation methods are shown in Figure 3. The figure shows that some CDFs of a data generation process either overfit or underfit the empirical CDF. Both the ML approach with $h = 0.1$ and the Monte Carlo simulation accurately capture the empirical data. In Table 3, we provide the test statistic $T_{NN1,k}$ and the memorization ratio for the generated data versus the training set (in-sample) and the generated data versus the test set (out-of-sample). We observe that all data generation methods except the one based on kernel-smoothing with $h = 0.1$ and the parametric model show severe deficits: Bootstrapping and kernel-smoothing with a very small h result in a very high memorization ratio: overfitting takes place because the empirical CDF is just memorized. Underfitting occurs when the bandwidth for kernel-smoothing is too large, because the kernel-smoothed CDF does not reproduce the empirical distribution function very well. In general, underfitting can be easily discovered by large values of the test statistic $T_{NN1,k}$. We note that in

TABLE 3 In-sample and out-of-sample validation of several approaches to generate new data such as bootstrapping, Monte Carlo and ML techniques. We use the memorization ratio (MR) with $\rho = 0.25$ and the test statistic $T_{NN1,k}$ with $k = 3$ for an in-sample validation, where $M = N = 15$: the empirical data is the training set, i.e., the yearly log-returns of the S&P 500 from 1997 to 2011. We use the test set, i.e., the yearly log-returns of the S&P 500 from 2012 to 2023, as empirical dataset for out-of-sample validation, where $M = N = 12$. Unacceptable values are displayed in bold. We average over 100 simulations. Since $M = N$, under H0, the memorization ratio converges to $\frac{\rho}{\rho+1} = 0.2$ for in-sample validation and out-of-sample validation. The entries in the table have a standard error of no more than ± 0.01 .

Data generation Method	$T_{NN1,k}$ in-sample	MR in-sample	$T_{NN1,k}$ out-of-sample	MR out-of- sample
Bootstrap	0.06	0.64	0.08	0.11
Data-driven, $h = 10^{-7}$	0.05	0.65	0.07	0.14
Data-driven, $h = 1$	0.22	0.08	0.27	0.07
Data-driven, $h = 0.1$	0.06	0.19	0.07	0.19
Parametric model	0.06	0.17	0.07	0.21

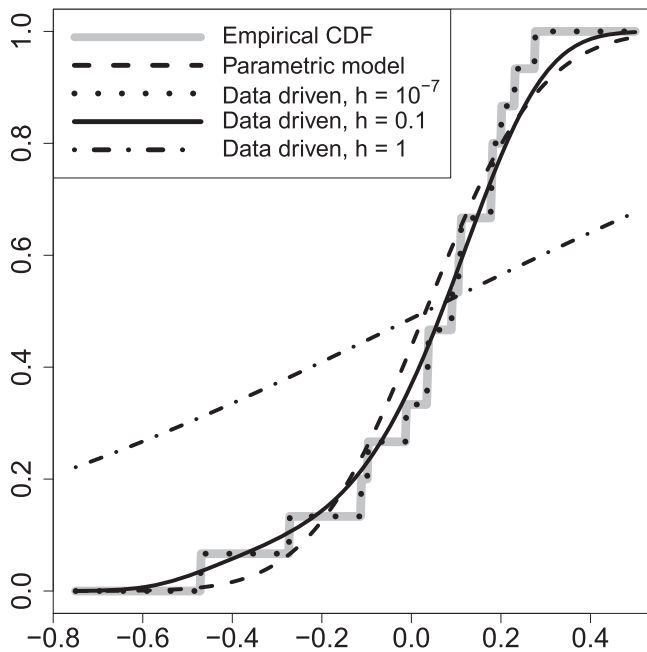


FIGURE 3 Comparison of the CDF of the empirical data of the yearly log-returns of the S&P 500 from 1997 to 2011 to the CDFs of generated data, where the data is generated by techniques such as bootstrapping, Monte Carlo and ML.

this very simple setting, the parametric model fits the distribution reasonably well. In other words, the memorization ratio and the test statistic $T_{NN1,k}$ help identify overfitting and underfitting in data generation methods. Unfortunately, especially for larger dimensions, adequate parametric models cannot always be determined. For example, in Section 3.2, we use an autoencoder-based approach to generate yield curves (of dimension 31); a setup where most parametric models are known to have some issues, especially in the dependence structure of different maturities.

Remark 8. The smaller h , the closer the empirical and generated data are, and the smaller the statistic $T_{NN1,k}$, but the larger the memorization ratio. On the other hand, if h is large, the generated data will be more and more different from the empirical data: the memorization ratio will be low, but the statistic $T_{NN1,k}$ will be (too) high. The opposite behavior of these two measures is also observed in Figure 4. How should one trade off? We suggest choosing h so that the memorization ratio is below its theoretical limit and the statistic $T_{NN1,k}$ is as small as possible. Figure 4 suggests that there is a fairly large range for h such that both measures return acceptable values.

3.2 | Interest rate scenario generation by autoencoders

After illustrating the main principles based on a low-dimensional and simple example, we now consider a purely data-driven setup. We start with interest rate data on a daily basis from 2000-08-01 to 2023-01-31 (more specifically German government rates, taken from the web site

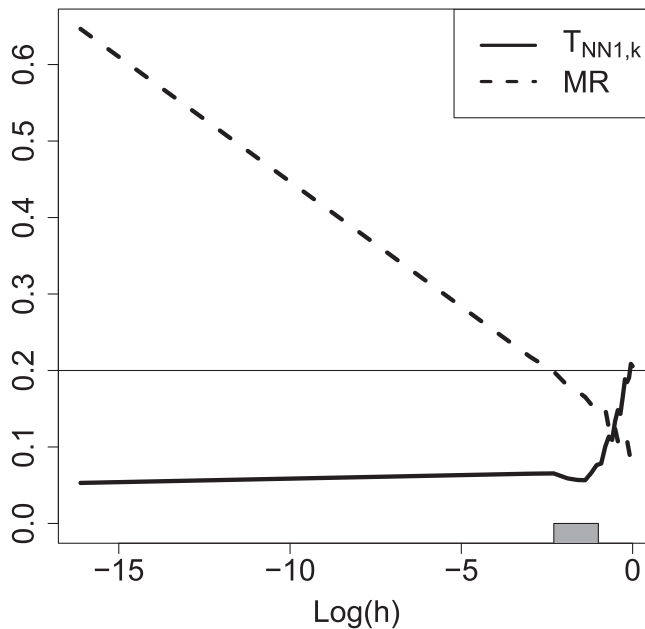


FIGURE 4 We show the memorization ratio (MR) with $\rho = 0.25$ and the statistic $T_{NN1,k}$ with $k = 3$ to generate new samples by a (data-driven) kernel-smoothing approach with different bandwidths h from the training set, that is, the yearly log-returns of the S&P 500 from 1997 to 2011. The theoretical limit of the memorization ratio is 0.2. Bandwidths leading to acceptable values for the memorization ratio and the statistic $T_{NN1,k}$ lay in the gray area. We average over 100 simulations.

of the Deutsche Bundesbank²), containing 5716 data points of dimension 31 (maturities 6M, 1Y, 2Y, ..., 30Y). Since we are interested in yield curve scenarios over the period of a month, we take the observations 1, 21, 41, ..., 5701, leading to 284 independent 1M-differences (1-month differences) which can be interpreted as realizations of $E_1, \dots, E_{284} \in \mathbb{R}^{31}$. We use the first 260 1M-difference curves $E_1, \dots, E_{260}$ as training data and hold out the remaining 24 1M-difference curves $E_{261}, \dots, E_{284}$ as validation data to train an autoencoder for scenario generation. Autoencoders are a special type of neural networks, which are briefly explained in Section B.

Given generated 1M-differences and assuming serial independence,³ annual interest rate scenarios for SCR calculations can then be simply obtained by adding 12 1M-differences. All implementations in this section have been carried out in Matlab R2024b, mainly based on the Deep Learning Toolbox.

Next, we briefly describe how the autoencoders is defined and trained: Consider two neural networks, an *encoder* network $\mathcal{E}_\phi : \mathbb{R}^{31} \rightarrow \mathbb{R}^2$ with parameters ϕ and a *decoder* network $\mathcal{D}_\theta : \mathbb{R}^2 \rightarrow \mathbb{R}^{31}$ with parameters θ . We choose the encoder and decoder both as fully-connected feed-forward networks with one hidden layer consisting of 62 neurons and encoding dimension 2 (= dimension of the so-called latent space). The autoencoder is trained by minimizing the following loss function via the L-BFGS optimization algorithm:

²https://www.bundesbank.de/dynamic/action/de/statistiken/zeitreihen-datenbanken/zeitreihen-datenbank/759778/759778?listId=www_skms_it03a.

³A runs test for serial independence showed no indication of dependence.

$$L(\phi, \theta) := \frac{1}{260} \sum_{m=1}^{260} \|E_m - Y_m\|_1 \longrightarrow \min! \quad \text{with } Y_m := \mathcal{D}_\theta(\mathcal{E}_\phi(E_m)).$$

L represents the reconstruction error of the autoencoder, that is, the average absolute difference between the empirical data and the encoded and afterwards decoded empirical data.

The autoencoder leads to an in-sample reconstruction error of approx. 2 basis points (bp) per tenor, and out-of-sample 2bp per tenor as well, which shows that encoding/decoding can be performed with quite good quality.

The encoder reduces a 31-dimensional interest rate curve to an artificial two-dimensional vector. The decoder transforms a two-dimensional vector into a 31-dimensional vector that can be interpreted as a possible 1M-difference curve. Figure 5 displays the different types of 1M-difference curves that the decoder can generate by varying the two-dimensional input. As can be seen, a wide variety of 1M-differences can be generated; that is, the autoencoder can produce a broad range of market scenarios.

In Figure 6, top panel, we have illustrated the 260 two-dimensional encoded 1M-difference yield curves, that is, $\mathcal{E}_\phi(E_1), \dots, \mathcal{E}_\phi(E_{260})$. In the bottom panel of Figure 6, we have illustrated a random sample according to a two-dimensional normal distribution with the same mean and covariance matrix as $\mathcal{E}_\phi(E_1), \dots, \mathcal{E}_\phi(E_{260})$. Figure 6 indicates a good fit of the encoded training data by a normal

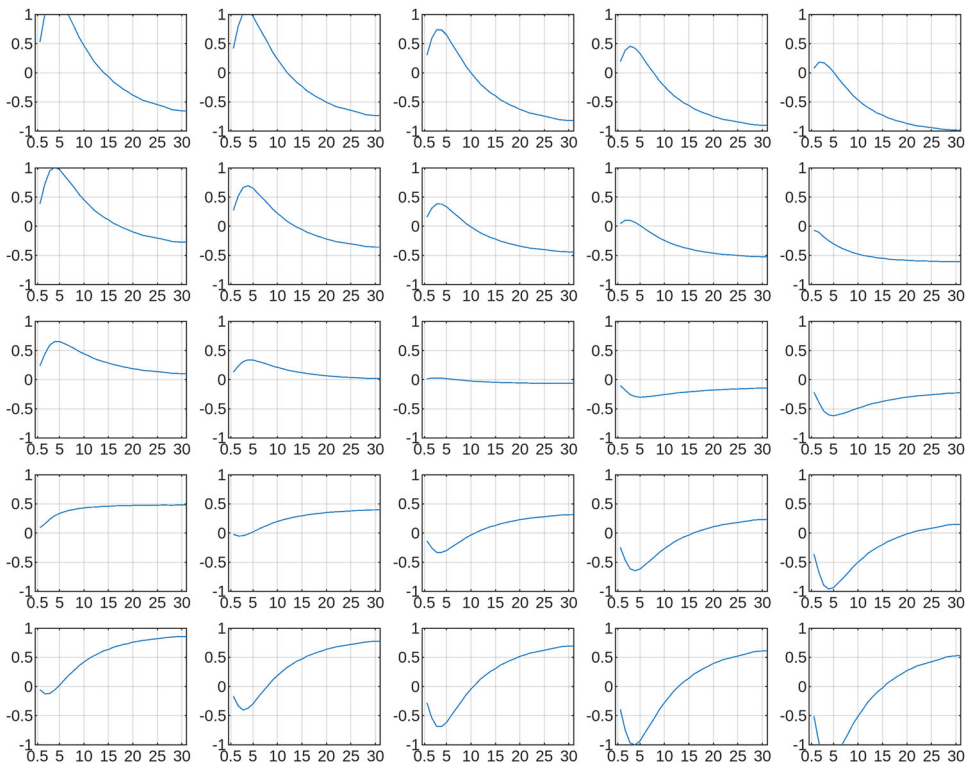


FIGURE 5 Different types of 1M-differences yield curves are generated by applying the decoder to 25 artificially chosen two-dimensional real vectors. The output of the decoder is a 31-dimensional vector which is interpreted as 1M-differences yield curves for maturities 6M, 1Y, 2Y, ..., 30Y. We use linear interpolation between the different maturities to obtain continuous curves. In particular, we show $\mathcal{D}_\theta(Z) \in \mathbb{R}^{31}$ over the 31 maturities where we chose for illustrative purposes $Z \in \{-2, -1, 0, 1, 2\}^2$.

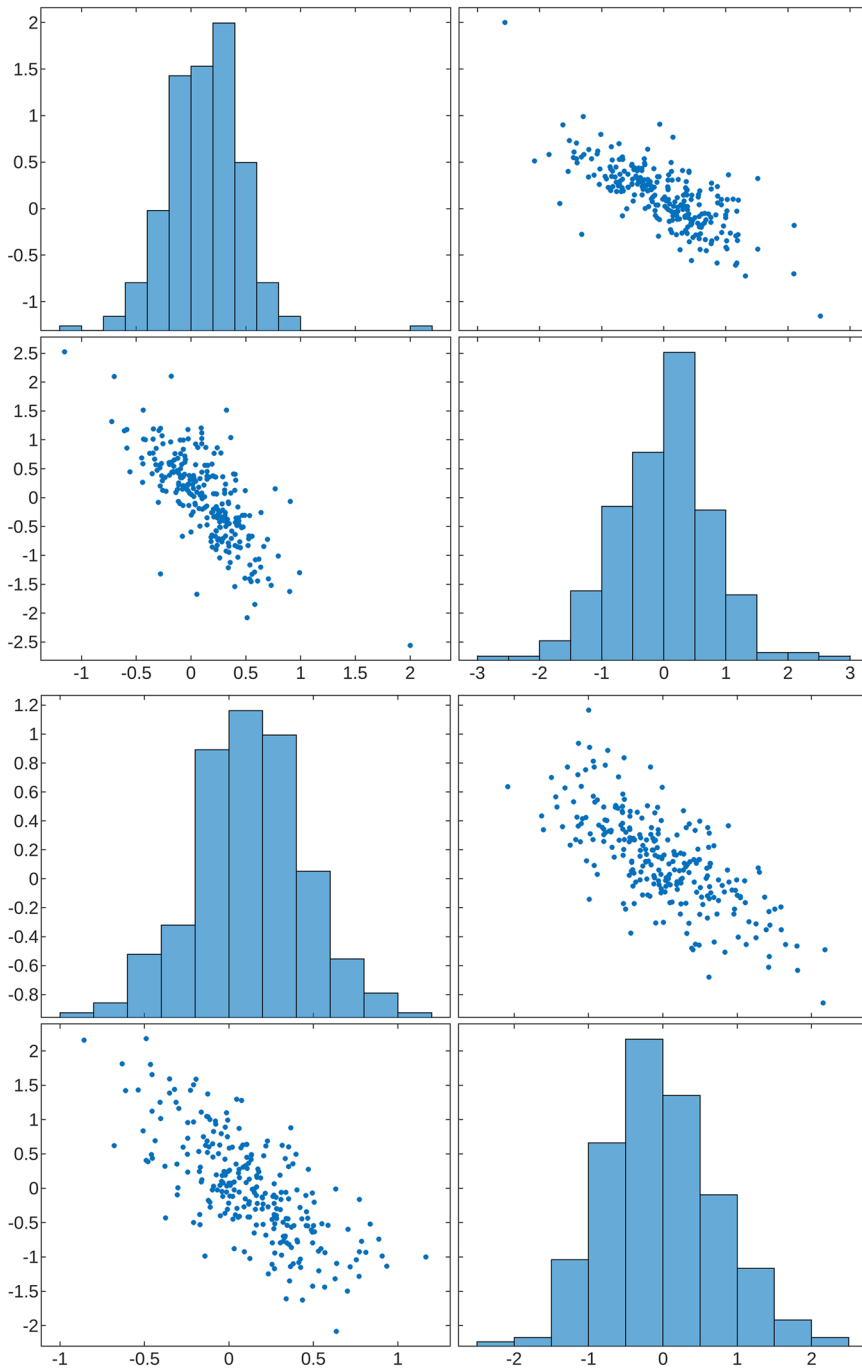


FIGURE 6 Top panel: marginal distribution and scatter plot of the 260 encoded observations $\mathcal{E}_\phi(E_1), \dots, \mathcal{E}_\phi(E_{260})$. Bottom panel: marginal distribution and scatter plot of 260 randomly generated samples from a two-dimensional normal distribution with the same mean and covariance matrix as $\mathcal{E}_\phi(E_1), \dots, \mathcal{E}_\phi(E_{260})$.

distribution. That is, normally distributed random variables can be used as input for the decoder to produce new 31-dimensional interest rate curves. We will pass two-dimensional realizations Z of a normal distribution with the same mean and covariance matrix as $\mathcal{E}_\phi(E_1), \dots, \mathcal{E}_\phi(E_{260})$ to the decoder to use the decoder in a generative manner as an ESG. Since the first and second moment of $\mathcal{D}_\phi(Z)$ do not necessarily exactly match those of the empirical data E , we adjust the autoencoder output by some independent correction term⁴ ε to obtain the generated data $G := \mathcal{D}_\phi(Z) + \varepsilon$. In the case that the encoded training set cannot be easily approximated in such a way, not even for larger encoding dimensions, the autoencoder can be enhanced to an *adversarial* or to a *variational autoencoder*, see Kingma and Welling (2019) and Makhzani et al. (2015) for more details.

Based on the trained autoencoder and its split into an encoder and decoder part, we then generate the following data sets: In the set S_1 , we collect the 260 original 1M-difference yield curves, i.e.,

$$S_1 := \{E_1, \dots, E_{260}\}.$$

In the set S_2 , we collect the 260 encoded and afterwards decoded 1M-difference yield curves, i.e.,

$$S_2 := \{\mathcal{D}_\theta(\mathcal{E}_\phi(E_m)), m = 1, \dots, 260\}.$$

In S_3 we collect the ML-based generated data: Let $Z_1, \dots, Z_{260}$ be samples of a two-dimensional normal distribution with the same mean and covariance matrix as $\mathcal{E}_\phi(E_1), \dots, \mathcal{E}_\phi(E_{260})$ and define

$$S_3 := \{\mathcal{D}_\theta(Z_m) + \varepsilon_m, m = 1, \dots, 260\},$$

where ε_m is chosen as in Footnote 4. We apply the statistic $T_{NN1,k}$ with $k = 3$ and the memorization ratio with $\rho = 0.25$ for an analysis of the ML-based generated dataset S_3 and we compare S_2 with S_3 : As expected, S_2 is not too far from S_1 ($T_{NN1,k} = 0.2$), while the memorization ratio between S_1 and S_2 is 0.63, which is a clear indicator for memorization. Due to the reconstruction error, the memorization error remains significantly beneath 1. We can further observe that the generated data S_3 is very close to the empirical data S_1 ($T_{NN1,k} = 0.07$). S_1 is closer to S_3 than to S_2 because of the correction term ε . The memorization ratio between S_1 and S_3 is only 0.32. The theoretical limit under H0 is 0.2, which is slightly below 0.32. For comparison, the memorization ratio when the empirical and generated data are 260 independent, standard normally distributed random variables in 31 dimensions, respectively, is 0.28.

To further validate our results, we run the following out-of-sample test: Let S_1^* denote the validation set consisting of 24 1M-difference yield curves, i.e.,

$$S_1^* = \{E_{261}, \dots, E_{284}\}.$$

The $T_{NN1,k}$ statistic with $k = 3$ between S_1 and S_1^* is 0.01. We therefore accept the hypothesis that samples S_1 and S_1^* come from the same distribution. This is an important observation for

⁴We choose ε multivariate normally distributed with mean $\mu_\varepsilon = E[E - \mathcal{D}_\phi(Z)]$ and covariance matrix $\Sigma_\varepsilon = UW^+U^T$, where UW^+U^T is the eigenvalue decomposition of the matrix $\text{cov}[E] - \text{cov}[\mathcal{D}_\phi(Z)]$ and $(W^+)_{i,i} = \max(0, W_{i,i})$. This way we match the first moment perfectly and we match the second moment of the empirical data E and the generated data G as good as possible.

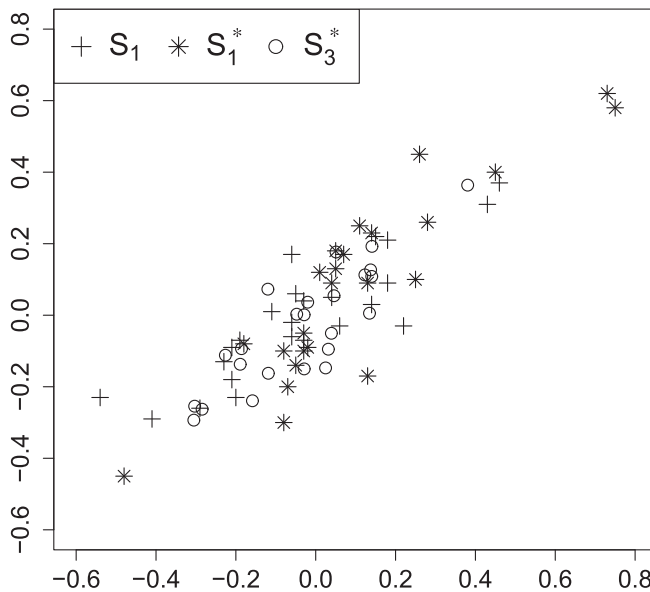


FIGURE 7 The figure shows the 3Y rate difference versus the 9Y rate difference (1M-differences) in percentage points for the first 24 samples of the training data S_1 , the generated data S_3^* and the out-of-sample data S_1^* .

the following analysis. Let S_3^* be the first 24 ML-based generated data from S_3 . The $T_{NN1,k}$ statistic with $k = 3$ between S_1^* and S_3^* is 0.05, that is, also out-of-sample, we accept the hypothesis that data generated by the autoencoder has the same distribution as the underlying distribution of the empirical data.

Except memorization caused by randomness, it is impossible for the generated set S_3^* to memorize any element of the out-of-sample empirical data S_1^* since the autoencoder has never seen the validation set S_1^* during training. The memorization ratio with $\rho = 0.25$ between S_1^* and S_3^* is 0.29, which is almost the same as the memorization ratio between the trainings set S_1 and the generated data S_3 . From this out-of-sample validation, we conclude that the autoencoder does not systematically memorize any training data, since the memorization ratios for in-sample and out-of-sample data are almost equal and close to its theoretical limit.

A visual inspection of the data in Figure 7 indicates that the training data S_1 , the generated data S_3^* and the out-of-sample validation data S_1^* come from the same distribution. At most, two points in S_1^* are noticeable, since they are slightly further away from the other points. The figures further indicate that the training data is not memorized.

The memorization ratio and the $T_{NN1,k}$ statistic together imply that the generated data does not memorize the training data (except by chance), but produces a distribution very close to the out-of-sample distribution.

4 | CONCLUSIONS

Regulators in different countries are discussing the possibility of using machine learning techniques, such as (variational) autoencoders or generative adversarial networks as economic scenario generators in an insurance context. They highlight the necessity of providing further

model validation compared to classical economic scenario generators based on Monte Carlo. In particular, the insurance company must check that underfitting does not occur, meaning that the machine learning technique can learn the multivariate empirical distribution of historical financial data. This can be tested using a suitable two-sample test, for example, as proposed by Mondal et al. (2015).

Insurance companies must also ensure that overfitting does not occur; that is, the machine learning technique must generate new scenarios rather than memorizing the training data. We provide a new measure, the memorization ratio (see Section 2.2). We show that the memorization ratio converges when the amount of data increases. Numerical experiments on simulated and real market data using several data generation methods (bootstrapping, kernel-smoothing, Monte Carlo, and autoencoders) show that our memorization ratio is able to detect the memorization effect. Our findings have relevant implications for the insurance and financial industries since they help to avoid overfitting of economic scenario generators based on machine learning techniques.

The literature has also proposed measures of the memorization effect. However, these measures mainly focus on image generation and do not analyze what happens when the samples size increases. In contrast, our applications focus on financial data in the context of actuarial science and we also provide several theoretical insights into the proposed measure.

On the theoretical side, future research may extend the main results on the convergence of the memorization ratio. For example, it is practically relevant to consider cases with (some weak serial) dependence in the time series data. Our numerical experiments suggest that this is feasible. Additionally, gaining more insight into the small sample bias of the memorization ratio would add value to the validation process. Finally, obtaining confidence intervals for the memorization ratio would be beneficial for constructing a statistical test on memorization. From a practical standpoint, extensive additional numerical studies and comparisons of different ESG methods could provide more experience with high-dimensional data.

ACKNOWLEDGMENTS

We thank two referees for many helpful comments that significantly improved the paper. Solveig Flaig would like to thank Deutsche Rückversicherung AG for the funding of this research. Opinions, errors, and omissions are solely those of the authors and do not represent those of Deutsche Rückversicherung AG or its affiliates. Open Access funding enabled and organized by Projekt DEAL.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

DATA AVAILABILITY STATEMENT

All real data analyzed during this study are publicly available. URLs are included in this article. The full code of the experiments are presented in <https://doi.org/10.7910/DVN/CXML8C>.

ORCID

Gero Junike  <https://orcid.org/0000-0001-8686-2661>

Ralf Werner  <https://orcid.org/0000-0001-8204-3757>

REFERENCES

- Arian, H., Moghimi, M., Tabatabaei, E., & Zamani, S. (2022). EncoDed Value-at-risk: A machine learning approach for portfolio risk measurement. *Mathematics and Computers in Simulation*, 202, 500–525.
- Arora, S., Ge, R., Liang, Y., Ma, T., & Zhang, Y. (2017). Generalization and equilibrium in generative adversarial nets (GANs). In D. Precup, & Y. W. Teh (Eds.), *International conference on machine learning* (pp. 224–232). PMLR.
- Arora, S., Risteski, A., & Zhang, Y. (2018). Do GANs learn the distribution? Some theory and empirics. In *International Conference on Learning Representations*. <https://openreview.net/forum?id=BJehNfW0-BaFin> (2021). Machine learning in risk models: Characteristics and supervisory priorities. <https://www.bundesbank.de/resource/blob/793670/61532e24c3298d8b24d4d15a34f503a8/mL/2021-07-15-ml-konsultationspapier-data.pdf>. [accessed on 2023/10/20].
- Bai, C.-Y., Lin, H.-T., Raffel, C., & Kan, W. C.-w. (2021). On training sample memorization: Lessons from Benchmarking generative modeling with a large-scale competition. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (pp. 2534–2542). <https://dl.acm.org/doi/abs/10.1145/3447548.3467198>
- Beder, T. S. (1995). VaR: Seductive but dangerous. *Financial Analysts Journal*, 51(5), 12–24.
- Bickel, P. J., & Breiman, L. (1983). Sums of functions of nearest neighbor distances, moment bounds, limit theorems and a goodness of fit test. *The Annals of Probability*, 11(1), 185–214.
- Borji, A. (2019). Pros and cons of GAN evaluation measures. *Computer Vision and Image Understanding*, 179, 41–65.
- Borji, A. (2022). Pros and cons of GAN evaluation measures: New developments. *Computer Vision and Image Understanding*, 215, 103329.
- Buch, R., Grimm, S., Korn, R., & Richert, I. (2023). Estimating the value-at-risk by temporal VAE. *Risks*, 11(5), 79.
- Chen, Y., Li, P., & Zhang, B. (2018). Bayesian renewables scenario generation via deep generative networks. In 2018 52nd Annual Conference on Information Sciences and Systems (CISS) (pp. 1–6). IEEE. <https://ieeexplore.ieee.org/abstract/document/8362314>
- Cont, R., Cucuringu, M., Xu, R., & Zhang, C. (2025). Tail-GAN: Learning to simulate tail risk scenarios. *Management Science*. <https://doi.org/10.1287/mnsc.2023.00936>
- Ebner, B., Henze, N., & Yukich, J. E. (2018). Multivariate goodness-of-fit on flat and curved spaces via nearest neighbor distances. *Journal of Multivariate Analysis*, 165, 231–242.
- European Union. (2009). Directive 2009/136/EC of the European Parliament and of the council. *Official Journal of the European Union*, 337, 11.
- Fernandez-Arjona, L. (2021). A neural network model for solvency calculations in life insurance. *Annals of Actuarial Science*, 15(2), 259–275.
- Flaig, S., & Junike, G. (2022). Scenario generation for market risk models using generative neural networks. *Risks*, 10(11), 199.
- Goodfellow, I. J., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press. <http://www.deeplearningbook.org>
- Gu, S., Kelly, B., & Xiu, D. (2021). Autoencoder asset pricing models. *Journal of Econometrics*, 222(1), 429–450.
- Hendricks, D. (1996). Evaluation of value-at-risk models using historical data. *Economic Policy Review*, 2(1), 39–69.
- Henze, N. (1988). A multivariate two-sample test based on the number of nearest neighbor type coincidences. *The Annals of Statistics*, 16(2), 772–783.
- Horvath, B., Plenk, J., & Vuletić, M. (2026). *Market generators: A paradigm shift in financial modeling* (pp. 127–159). Springer Nature Switzerland. ISBN 978-3-031-97239-3. https://doi.org/10.1007/978-3-031-97239-3_4
- Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders.
- Lopez-Paz, D., & Oquab, M. (2016). Revisiting classifier two-sample tests. *arXiv*. <https://arxiv.org/abs/1610.06545>
- Makhzani, A., Shlens, J., Jaitly, N., & Goodfellow, I. J. (2015). Adversarial autoencoders. CoRR, abs/1511.05644. <http://arxiv.org/abs/1511.05644>
- Meehan, C., Chaudhuri, K., & Dasgupta, S. (2020). A non-parametric test to detect data-copying in generative models. In *International Conference on Artificial Intelligence and Statistics*.
- Mondal, P. K., Biswas, M., & Ghosh, A. K. (2015). On high dimensional two-sample tests based on nearest neighbors. *Journal of Multivariate Analysis*, 141, 168–178.
- Nagarajan, V., Raffel, C., & Goodfellow, I. J. (2018). Theoretical insights into memorization in GANs. In *Neural Information Processing Systems Workshop (Vol. 1)*.

- Ni, H., Szpruch, L., Sabate-Vidales, M., Xiao, B., Wiese, M., & Liao, S. (2021, November). Sig-Wasserstein GANs for time series generation. In *Proceedings of the second ACM international conference on AI in finance* (pp. 1–8). <https://dl.acm.org/doi/abs/10.1145/3490354.3494393>
- Pedersen, H., Campbell, M. P., Christiansen, S. L., Cox, S. H., Finn, D., Griffin, K., Hooker, N., Lightwood, M., Sonlin, S. M., & Suchar, C. (2016). Economic scenario generators: A practical guide. *The Society of Actuaries* (July 2016). <https://www.soa.org/globalassets/assets/Files/Research/Projects/research-2016-economic-scenario-generators.pdf>
- Pfeifer, D., & Ragulina, O. (2018). Generating VaR scenarios under Solvency II with product beta distributions. *Risks*, 6(4), 122.
- Pritsker, M. (2006). The hidden dangers of historical simulation. *Journal of Banking & Finance*, 30(2), 561–582.
- Radhakrishnan, A., Yang, K., Belkin, M., & Uhler, C. (2018). Memorization in overparameterized autoencoders. *arXiv*. <https://openreview.net/forum?id=BJeZw4B334>
- Richman, R., & Scognamiglio, S. (2024). Multiple yield curve modeling and forecasting using deep learning. *ASTIN Bulletin: The Journal of the IAA*, 54(3), 463–494.
- Sandström, A. (2016). *Handbook of solvency for actuaries and risk managers: Theory and practice*. CRC Press.
- Schilling, M. F. (1986). Multivariate two-sample tests based on nearest neighbors. *Journal of the American Statistical Association*, 81(395), 799–806.
- van den Burg, G., & Williams, C. (2021). On memorization in probabilistic deep generative models. *Advances in Neural Information Processing Systems*, 34, 27916–27928.
- van der Burgt, J. (2019). General principles for the use of Artificial Intelligence in the financial sector. <https://www.dnb.nl/media/voffsrc/general-principles-for-the-use-of-artificial-intelligence-in-the-financial-sector.pdf>. [accessed on 2023/10/20].
- Varnell, E. M. (2011). Economic scenario generators and Solvency II. *British Actuarial Journal*, 16(1), 121–159.
- Wei, J., Zhang, Y., Zhang, L. Y., Ding, M., Chen, C., Ong, K.-L., Zhang, J., & Xiang, Y. (2025). Memorization in deep learning: A survey. *ACM Computing Surveys*, 58(4), 1–35.
- Weiss, L. (1960). Two-sample tests for multivariate distributions. *The Annals of Mathematical Statistics*, 31(1), 159–164.
- Wiese, M., Knobloch, R., Korn, R., & Kretschmer, P. (2020). Quant GANs: Deep generation of financial time series. *Quant Finance*, 20(9), 1419–1440.
- Xu, Q., Huang, G., Yuan, Y., Guo, C., Sun, Y., Wu, F., & Weinberger, K. (2018). An empirical study on evaluation metrics of generative adversarial networks. *arXiv*.

How to cite this article: Junike, G., Flaig, S., & Werner, R. (2026). Validation of machine learning based scenario generators. *Journal of Risk and Insurance*, 93, 508–533. <https://doi.org/10.1111/jori.70040>

APPENDIX A: CONVERGENCE RESULTS

Theorem 2 (Weiss, 1960). Let $(\Omega, \mathcal{F}, P)$ be a probability space. Let $E_1, \dots, E_M, G_1, \dots, G_N$ be independent and identically distributed d -variate random variables. Assume E_1 has a piecewise continuous and bounded probability density function f . Let $\mu \in (0, 1]$ and

$$D_m^\mu := \mu \min_{m' \neq m} \|E_m - E_{m'}\|_2, m = 1, \dots, M$$

and let $S_m, m = 1, \dots, M$, be the number of points $G_1, \dots, G_N$ contained in the open sphere $\{x : \|x - E_m\|_2 < D_m^\mu\}$. Then the joint distribution of S_i, S_l is the same as the joint distribution of $S_{i'}$ and $S_{l'}$ for any $i \neq l$ and $i' \neq l'$. It further holds for any $\alpha > 0$ that

$$\lim_{M \rightarrow \infty, \frac{M}{N} = \alpha} P[S_1 = s] = Q(s) := \int_{\mathbb{R}^d} \frac{\mu^{-d}\alpha}{(1 + \mu^{-d}\alpha)^{s+1}} f(x) dx, s \in \mathbb{N}_0 \quad (\text{A1})$$

and

$$\lim_{M \rightarrow \infty, \frac{M}{N} = \alpha} P[S_1 = s_1 \cap S_2 = s_2] = Q(s_1)Q(s_2), s_1, s_2 \in \mathbb{N}_0.$$

Proof. Lionel Weiss proved the Theorem for $\mu = \frac{1}{2}$ in Weiss (1960), but his proof works exactly the same for any $\mu \in (0, 1]$. $\square$

Proof. Proof of Theorem 1: Let $\mu := \rho^{\frac{1}{d}} > 0$. Let Q, D_m^μ, S_m as in Theorem 2. Denote by f the density of E_1 . It holds that

$$Q(0) = \frac{\alpha}{\mu^d + \alpha} \underbrace{\int_{\mathbb{R}^d} f(x) dx}_{=1} = \frac{\alpha}{\rho + \alpha} < \infty.$$

Define R_m as in Equation (3) then $\rho^{\frac{1}{d}}R_m = D_m^\mu$ by the definition of μ . Define

$$Z_m := \mathbf{1}_{\left[0, \rho^{\frac{1}{d}}R_m\right)} \left(\min_{n=1, \dots, N} \|G_n - E_m\|_2 \right), m = 1, \dots, M.$$

Then $\Pi_{M,N}^\rho = \frac{1}{M} \sum_{m=1}^M Z_m$ and

$$\begin{aligned} \mathbf{E}[Z_m] &= P \left[\min_{n=1, \dots, N} \|G_n - E_m\|_2 < \rho^{\frac{1}{d}}R_m \right] \\ &= 1 - P \left[\forall n : \|G_n - E_m\|_2 \geq \rho^{\frac{1}{d}}R_m \right] \\ &= 1 - P[S_m = 0] \\ &= 1 - P[S_1 = 0]. \end{aligned}$$

It follows by Theorem 2 that

$$\lim_{M \rightarrow \infty, \frac{M}{N} = \alpha} \mathbf{E}[Z_m] = 1 - Q(0) = \frac{\rho}{\rho + \alpha}. \quad (\text{A2})$$

As $Z_m^2 = Z_m$, it follows that

$$\lim_{M \rightarrow \infty, \frac{M}{N} = \alpha} \text{Var}(Z_m) = 1 - Q(0) - (1 - Q(0))^2 = \frac{\rho\alpha}{(\rho + \alpha)^2} < \infty, m = 1, 2, \dots$$

Let $i \neq l$. Then

$$\begin{aligned}
\text{Cov}(Z_i, Z_l) &= \mathbf{E}[Z_i Z_l] - \mathbf{E}[Z_i] \mathbf{E}[Z_l] \\
&= P\left(\min_{n=1, \dots, N} \|G_n - E_i\|_2 < \rho^{\frac{1}{d}} R_i \cap \min_{n=1, \dots, N} \|G_n - E_l\|_2 < \rho^{\frac{1}{d}} R_l\right) - (\mathbf{E}[Z_1])^2 \\
&= 1 - P[S_i = 0 \cup S_l = 0] - (\mathbf{E}[Z_1])^2 \\
&= 1 - P[S_1 = 0 \cup S_2 = 0] - (\mathbf{E}[Z_1])^2 \\
&= 1 - P[S_1 = 0] - P[S_2 = 0] + P[S_1 = 0 \cap S_2 = 0] - (\mathbf{E}[Z_1])^2.
\end{aligned}$$

Therefore we have that

$$\lim_{M \rightarrow \infty, \frac{M}{N} = \alpha} \text{Cov}(Z_i, Z_l) = 1 - Q(0) - Q(0) + Q^2(0) - (1 - Q(0))^2 = 0.$$

By Bienaymé's identity it follows that

$$\begin{aligned}
\mathbf{E}\left[\left|\Pi_{M,N}^\rho - \mathbf{E}[Z_1]\right|^2\right] &= \text{Var}\left(\frac{1}{M} \sum_{i=1}^M Z_i\right) \\
&= \frac{1}{M^2} \left(\sum_{i=1}^M \text{Var}(Z_i) + \sum_{i \neq l} \text{Cov}(Z_i, Z_l) \right) \\
&= \frac{1}{M^2} (M \text{Var}(Z_1) + (M^2 - M) \text{Cov}(Z_1, Z_2)).
\end{aligned}$$

Thus,

$$\lim_{M \rightarrow \infty, \frac{M}{N} = \alpha} \mathbf{E}\left[\left|\Pi_{M,N}^\rho - \mathbf{E}[Z_1]\right|^2\right] = 0. \quad (\text{A3})$$

Further, by the Minkowski inequality, it holds that

$$\sqrt{\mathbf{E}\left[\left|\Pi_{M,N}^\rho - \frac{\rho}{\rho + \alpha}\right|^2\right]} \leq \sqrt{\mathbf{E}\left[\left|\Pi_{M,N}^\rho - \mathbf{E}[Z_1]\right|^2\right]} + \sqrt{\mathbf{E}\left[\left|\mathbf{E}[Z_1] - \frac{\rho}{\rho + \alpha}\right|^2\right]}.$$

Apply Equation (A2) and Equation (A3) to conclude. $\square$

APPENDIX B: BACKGROUND: BASICS OF NEURAL NETWORKS

A *neural network* is a system that is broadly inspired from the human brain. It can transform a (one- or more-dimensional) input into a (one- or more-dimensional) output. A neural network consists of *neurons* which are arranged in *layers*. The neurons of one layer are connected to the neurons of the next layer. All starts with the *input layer*, which is then followed by one or more so called *hidden layers* and ends with the *output layer*, as Figure B1 based on Fernandez-Arjona (2021, Fig. 1) illustrates. A detailed and more thorough treatment of neural networks can be found in Goodfellow et al. (2016).

All neurons transform their multi-dimensional input (which comes from either the input layer or the proceeding hidden layer) via a so called *activation function* into an one-dimensional output which is then transferred to all the neurons in the next layer.

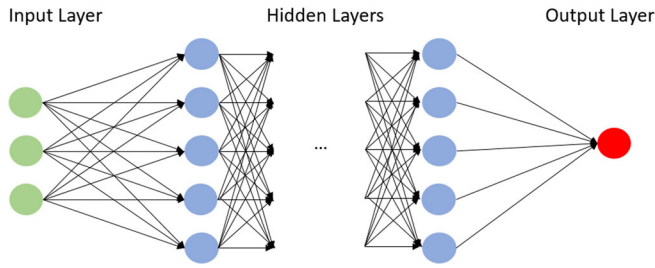


FIGURE B1 Neural network structure, based on Fernandez-Arjona (2021, Fig. 1).

A typical choice for an activation function g is the rectified linear unit $g(t) = \max(t, 0)$. For a neuron input $x \in \mathbb{R}^n$ and *parameters* or *weights* $w \in \mathbb{R}^{n+1}$ the output of the neuron is then given by $g(\sum_{i=1}^n w_i x_i + w_{n+1})$.

The parameters in the activation function in each neuron are often initialized randomly. Afterwards, these parameters are optimized when giving the neural network training data consisting of pairs of input data and the desired output so that the output of the neural network resembles the desired output given the training data.

APPENDIX C: DEPENDENT DATA

For illustration purposes, we simulate four-dimensional financial daily data by a (correlated) geometric Brownian motion. We apply a rolling (estimation) window to this data in order to construct financial scenarios over a larger time horizon. Let $w \in \mathbb{N}$ be some appropriate window size, for example, $w = 5$, $w = 21$ and $w = 63$ corresponds to weekly, monthly or quarterly data. If $w > 1$, the data becomes auto-correlated in the time-dimension, due to window overlapping. We see in Figure C1 that the memorization ratio still seems to converge (even though the independence assumption in H0 is violated) with the same order of convergence.

The time series are constructed as follows: Let $M, N \in \mathbb{N}$. Assume a business year has 252 days and a month consists of 21 business days. Let $\delta = \frac{1}{252}$. Let $W^1, \dots, W^d$ be independent standard Brownian motions. Let $\Sigma \in \mathbb{R}^{d \times d}$ be symmetric positive definite. Let A such that $AA^T = \Sigma$. Define

$$S_t^i = S_0^i \exp \left[\left(r - \frac{1}{2} \sum_{h=1}^d A_{ih}^2 \right) t + \sum_{h=1}^d A_{ih} W_t^h \right], i = 1, \dots, d.$$

We set $S_0^i = 1$, $r = 0$, $\Sigma_{ii} = \sigma^2$, $\Sigma_{ij} = \sigma^2 \gamma$ with $\sigma = 0.2$ and correlation $\gamma = 0.7$. We consider a d -dimensional time series of daily prices of length $M + N + 2w$, i.e.,

$$S_0^i, \dots, S_{\delta(M+N+2w-1)}^i, i = 1, \dots, d.$$

Based on this time series of $M + N + 2w$ daily prices, we will construct two samples of length M and N of log-returns over a time horizon of w days: Let

$$E_m^i = \log \left(\frac{S_{\delta(m-1+w)}^i}{S_{\delta(m-1)}^i} \right), i = 1, \dots, d, m = 1, \dots, M$$

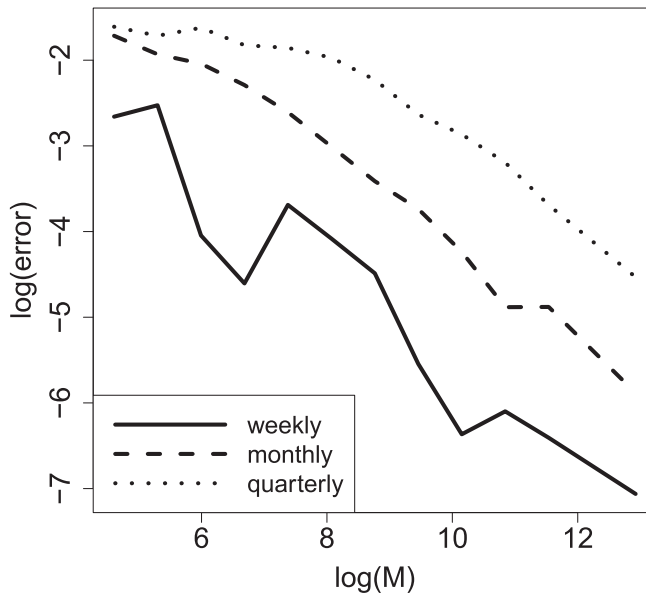


FIGURE C1 Memorization ratio for three different window sizes for increasing M and N . We choose $N = M$ and $\rho = 0.25$ and consider a correlated geometric Brownian motion in four dimensions as data generation method for the empirical and the generated data. The error is the absolute difference of the memorization ratio and its limit.

and

$$G_n^i = \log \left(\frac{S_{\delta(M+w+n-1+w)}^i}{S_{\delta(M+w+n-1)}^i} \right), i = 1, \dots, d, n = 1, \dots, N.$$

If $w = 1$ then E_m^i correspond to daily log-returns and E_m^i and $E_{m'}^i$, for $m \neq m'$ are independent. If $w = 21$ then E_m^i corresponds to monthly log-returns. The two groups $E_1, \dots, E_M$ and $G_1, \dots, G_N$ are independent. However, in the time-dimension, the points inside a group are not necessarily independent: Let $m > m'$. If $m \geq w + m'$ then E_m^i and $E_{m'}^i$ are independent since the Brownian motions have independent increments, otherwise, they are correlated. A similar statement holds for G_n^i and $G_{n'}^i$.

APPENDIX D: R CODE

The following R code implements the statistic $\Pi_{M,N}^\rho$ in the function `memorizationRatio()` and the statistic $T_{NN1,k}$ in the function `TNN1k()`. When sorting neighbors according to distance, in cases of a tie, we first collect all neighbors from the same set. For example, among all neighbors of an empirical (generated) data point that have the same distance, we first count other empirical (generated) data points as neighbors, and then generated (empirical) data points. Feel free to replace the for-loops with the function `mclapply` from the *parallel* package to parallelize the code.

```
#Input: matrix D and row number i.
#Returns: vector of sorted distances of vector D[i,] to rows of matrix D[-i,].
disIn = function(D, i){
  return(sqrt(rowSums((matrix(D[i,], nrow = (nrow(D)-1), ncol = ncol(D), byrow = T) -
    D[-i,])^2)))
}
```

```

}
#Input: matrix D and vector x with length equal to number of columns of D.
#Returns: vector of length nrow(D) of sorted distances of x to rows of D.
disOut = function(x, D){
  return(sqrt(rowSums((matrix(x, nrow = nrow(D), ncol = ncol(D), byrow = T) - D)^2)))
}
#Input: vectors x and y and integer k in {1,..., length(x)}
#Returns: the number of elements of x which are in the first k entries of the contacted and
sorted vector c(x, y).
numberSameNearestNeighbor = function(x, y, k){
  tmp = matrix(c(x, y, rep(1, length(x)), rep(0, length(y))), ncol = 2)
  return(sum(tmp[order(tmp[, 1]),][1:k, 2]))
}
#Input: M x d matrix of empirical data, N x d matrix of generated data, rho in (0,1]
#Returns: memorization ratio
memorizationRatio = function(E, G, rho){
  d = ncol(E)
  M = nrow(E)
  N = nrow(G)
  MR = 0
  for(m in 1:M){
    myDisOut = disOut(E[m,], G) #Distance of Em to generated data points
    myDisIn = disIn(E, m) #Distance of Em to other empirical data points
    if(min(myDisOut) < (min(myDisIn) * rho^(1/d))){
      MR = MR + 1
    }
  }
  return(MR/M)
}
#Input: M x d matrix of empirical data, N x d matrix of generated data, k in {1, 2,...}
#Returns: the statistic TNN1k
TNN1k = function(E, G, k){
  d = ncol(E)
  M = nrow(E)
  N = nrow(G)
  TEk = 0
  TGk = 0
  for(m in 1:M){
    myDisOut = disOut(E[m,], G) #Distance of Em to generated data points
    myDisIn = disIn(E, m) #Distance of Em to other empirical data points
    TEk = TEk + numberSameNearestNeighbor(myDisIn, myDisOut, k)
  }
  for(n in 1:N){
    myDisOut = disOut(G[n,], E) #Distance of Gn to empirical data points
    myDisIn = disIn(G, n) #Distance of Gn to other generated data points
    TGk = TGk + numberSameNearestNeighbor(myDisIn, myDisOut, k)
  }
}

```

$$\begin{aligned} & \text{TEk} = \text{TEk}/(\text{M} * \text{k}) \\ & \text{TGk} = \text{TGk}/(\text{N} * \text{k}) \\ & \text{return}((\text{M} * \text{abs}(\text{TEk} - (\text{M} - 1)/(\text{M} + \text{N} - 1)) + \text{N} * \text{abs}(\text{TGk} - (\text{N} - 1)/(\text{M} + \text{N} - 1)))/(\text{N} + \text{M})) \\ & \} \end{aligned}$$

APPENDIX E: DATA

See Table E1.

TABLE E1 S&P 500, yearly log-returns between beginning of January and end of December for the years 1997 till 2023.

Year	Log-returns	S&P 500 Jan.	S&P 500 Dec.	Training/Testing
1997	0.2751	737.01	970.43	Training
1998	0.2317	975	1229.23	Training
1999	0.1793	1228.1	1469.25	Training
2000	−0.0973	1455.22	1320.28	Training
2001	−0.1113	1283.27	1148.08	Training
2002	−0.2719	1154.67	879.82	Training
2003	0.1994	909.03	1109.64	Training
2004	0.0892	1108.48	1211.92	Training
2005	0.0377	1202.08	1248.29	Training
2006	0.1114	1268.8	1418.3	Training
2007	0.0347	1418.3	1468.36	Training
2008	−0.4714	1447.16	903.25	Training
2009	0.1796	931.8	1115.1	Training
2010	0.1044	1132.99	1257.64	Training
2011	−0.0113	1271.89	1257.6	Training
2012	0.1104	1277.06	1426.19	Testing
2013	0.2342	1462.42	1848.36	Testing
2014	0.1168	1831.98	2058.9	Testing
2015	−0.0070	2058.2	2043.94	Testing
2016	0.1065	2012.66	2238.83	Testing
2017	0.1690	2257.83	2673.61	Testing
2018	−0.0727	2695.81	2506.85	Testing
2019	0.2524	2510.03	3230.78	Testing
2020	0.1423	3257.85	3756.07	Testing
2021	0.2530	3700.65	4766.18	Testing
2022	−0.2226	4796.56	3839.5	Testing
2023	0.2210	3824.14	4769.83	Testing