Graph-Guided Time Series Generation with Applications to Financial Correlation Modeling

Howard Caulfield
University of Limerick
howard.caulfield@ul.ie

James P. Gleeson
University of Limerick
james.gleeson@ul.ie

Abstract

Generating multivariate time series that maintain desired characteristics while controlling the dependency structure is an open challenge. We propose a graph-guided time series generation model that generates both desired node and hierarchical dependency structures. We highlight the effectiveness of the method through pricing a correlation trade for the US Presidential Election.

1 Introduction

Time series modeling and generation remains an active research topic despite an extensive history. A core component of this challenge is controlling the dependency structure between multiple time series. In this paper we focus on time series from financial markets, which are driven by complex dependency structures that are central to risk management, asset pricing and portfolio optimization. While deep generative models have modeled characteristics of financial time series with some success, they fail to capture conditional dependencies [1]. Similarly, other than [2, 3], these models typically lack explicit mechanisms to incorporate desired dependency structure. Furthermore, graph and network effects have often been touted as explanations for complex system behaviours (e.g., financial contagion) observed in financial time series [4–7].

Current approaches such as equity factor models like BARRA [8] either require domain knowledge or well specified dependency structures (e.g., vine copulas which are less feasible for large universes of financial instruments, [9] demonstrates a use case for 96 stocks). Alternative methods to generate correlated time series require well defined correlation matrices. In high-dimensional settings, this can be difficult given noisy samples and limited sample size. Advanced approaches such as eigenvalue clipping [10] and Ledoit-Wolf covariance regularization estimation [11] offer steps towards solutions but can significantly alter the realised dependency structure.

In this work we offer an alternative solution to incorporating desired dependencies while retaining desired time series characteristics. The key idea is to use a graph abstraction of the dependency structure, allowing edge types based on this abstraction to help guide the generation process. We use FiLM [12] convolutions to learn conditional time series generators across relational types. We introduce a novel loss function which allows for our model to control for hierarchical relationships and dependency structures within the generated time series. We highlight the model's ability to achieve desired dependency structures at both node scale and on aggregate. Lastly, we demonstrate a practical use case of generating correlated time series, pricing correlation of US stock index and sectors during the timeframe of the 2024 US Presidential Election.

2 Methodology

2.1 Data

We test the model on a substantial financial dataset which allows for investigation of hierarchical graph components. The empirical dataset includes 392 tradable instruments, 2 index exchange traded

Caulfield et al., Graph-Guided Time Series Generation with Applications to Financial Correlation Modeling (Extended Abstract). Presented at the Fourth Learning on Graphs Conference (LoG 2025), Hybrid Event, December 10–12, 2025.

funds (ETFs), 11 sector ETFs and 379 single stock names. For the empirical application, we use option data from the 30^{th} August 2024. Details of index coverage and component weightings can be found in the appendix A.4.

2.2 Problem Formulation

We define the graph-guided time series generation problem as follows. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{G}$ represents the graph (the desired structural relationships), $\mathcal{V} = \{1, \ldots, N\}$ is the set of nodes (each representing a stock) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges abstracted from the dependency values. To accommodate hierarchical dependencies, we introduce *supernodes* S, i.e., aggregated nodes to represent the hierarchical feature (e.g., weighted sum of nodes such as a stock index) where $S = \Sigma w_i \cdot v_i \; \forall v_i \in \mathcal{U}$ and $\Sigma w_i = 1$ where $\mathcal{U}$ represents the set of nodes in the supernode S and w_i represents the weight of node v_i feature for supernode S. We let $X \in \mathbb{R}^{T \times N}$ denote the real multivariate time series defined over all nodes of the graph, where T is the number of time steps and N is the number of variables (i.e., $|V|$). Including supernodes extends the time series to $X \in \mathbb{R}^{T \times N'}$ where $N' - N = |S|$. Let C be a conditioning variable (e.g., past time series of each node) which is used to condition the generator. The joint data is then $(X, C, \mathcal{G}) \sim P_{X,C,\mathcal{G}}$. We define a generator function $\tilde{X} = H(Z, C, \mathcal{G})$, where $Z \sim P_Z$ is a latent noise variable. The goal is to learn H such that $H(Z, C, \mathcal{G}) \sim P_{X|C,\mathcal{G}}$. That is, for every conditioning variable C and desired graph structure $\mathcal{G}$, the generator produces time series samples that match the true conditional distribution and desired graph structure $P_{H(Z,C,\mathcal{G})} \approx P_{X|C,\mathcal{G}} \;\; \forall (C, \mathcal{G})$.

2.3 Loss Functions

We use a multi-objective loss function (similar to the soft constraints of [13]), which combines maximum mean discrepancy distance [14] with a gaussian distance kernel over different time series characteristics. Maximum mean discrepancy (MMD) compares samples from the true distribution P and the synthetic distribution Q. Minimizing this distance under the kernel feature expansion is equivalent to minimizing the distance between the two distributions P and Q:

$$\text{MMD}^2(P, Q) = \mathbb{E}_{x,x' \sim P}\left[k(x, x')\right] + \mathbb{E}_{y,y' \sim Q}\left[k(y, y')\right] - 2\mathbb{E}_{x \sim P, y \sim Q}\left[k(x, y)\right], \tag{1}$$

where $k(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$ and kernel bandwidth parameter $\sigma > 0$. Bandwidth choice is an open research problem, similar to [14] we average the loss term over multiple bandwidth scales[1]. Loss terms are weighted to give approximately equal importance. We also included some constraint losses for initial training steps, e.g., bounds on cumulative series sums over entire generated output. In the final model, the cumulative loss function includes MMD over the many characteristics, e.g., moments of expected conditional returns and desired correlation. The equivalent *supernode* losses are also calculated and added separately.

2.4 Architecture Overview

Figure 1 illustrates the graph-guided generation procedure[2]. The process starts by concatenating sampled noise (z) to the conditioning data x per node. Edges are abstracted from correlation matrices, selecting the top-k absolute correlation values as edges. These are then bucketed by value into a relationship type r. We use this relation to guide our generation. The conditioning data x is passed through a FiLM convolution f_r. The convolution $f_r(x)$ produces synthetic data point(s) (x_{gen}), which are then concatenated to the conditioning vector x and the cycle continues to generate for the desired number of synthetic data points. Pseudo-code is included in appendix F.

3 Results

3.1 Graph-Guided Time Series

Figure 2 shows the performance of the graph-guided generator at achieving desired dependencies[3]. The dependency of *supernodes* are compared to the generated dependencies from true index nodes

[1] We use an additional loss to help target *relatively more important* correlations within the dataset. The metric is similar to that of top-k losses used in information retrieval (see appendix C).

[2] In appendix B, figure 8 we detail the generation flow from the FiLM Convolution.

[3] Further confirming results using alternative measures can be found in appendix E.

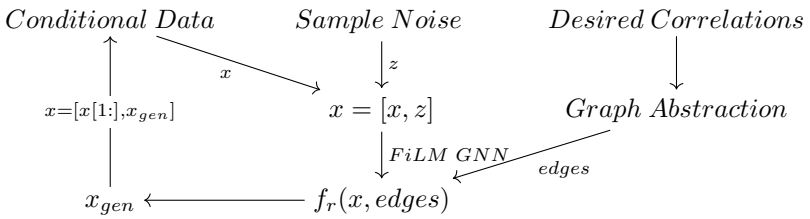
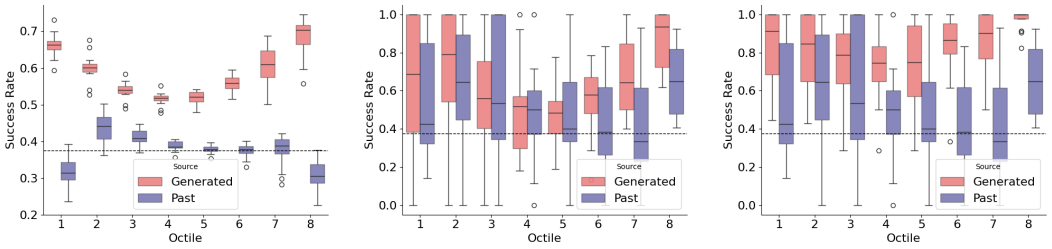


Figure 1: Graph-Guided Autoregressive Network.

(i.e., index nodes that are treated as normal nodes). The performance from the true index nodes can be also viewed as an inspection of the model's performance on a smaller universe of instruments (albeit with a stronger than normal stock correlation network). To determine the performance of the generator, we split the desired correlation values into octiles. We examine the success rate of the graph-guided generator at correctly generating correlation of time series. Figure 2a represents the success rates for all nodes. We find strong agreement between generated and desired correlations, whereas past correlations perform similar to a uniform random correlation generator (dashed black line). Figure 2b represents the success rates for all supernodes, i.e., the correlation structure of aggregated nodes vs. the desired dependency of the aggregated structures. There is clearly more variance here, particularly for lower octiles (i.e., typically negative correlations). Again, on average the generated data has good agreement between desired and generated. The past (conditioned) dependency structure also does better than random on average, indicating that the dependency structure of aggregated financial graph relationships are less dynamic. Lastly, figure 2c represents a subset of the graph nodes, our index nodes. They manage to nearly exactly match the dependency structure, with success rate very close to 1. The dashed lines for octiles 1,2,6,7,8 represent that the body of the boxplot is focused at 1.0.



(a) All Nodes **(b)** Supernodes **(c)** Index Nodes only

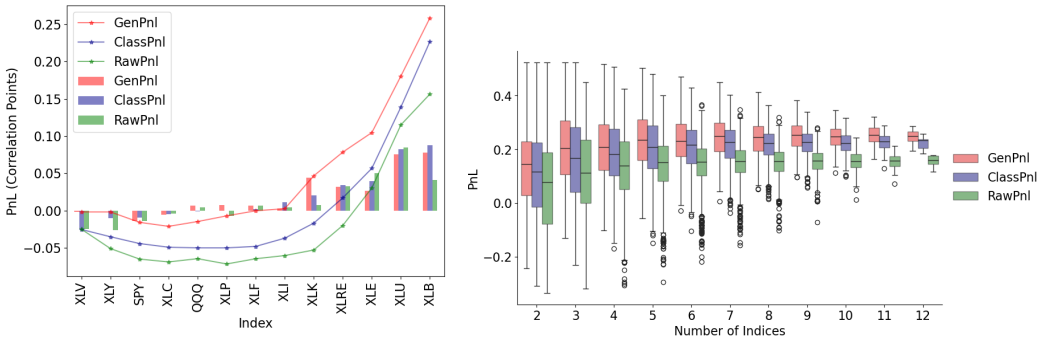
Figure 2: The boxplots represent how often (success rate, y-axis) an edge is classified into its desired octile (x-axis) or neigbouring octiles, e.g., we desire correlation of stock i, j to be in octile 1, if the generated correlation is in either octile 1 or 2, we consider this a successful generation. The red (blue) represents the success rates of the generated (conditioning/past) correlation data. High success in the generated data confirms accurate dependency generation. The strong performance of *Index nodes only* indicates the accuracy of the model for lower dimensional graph-guided generation.

3.2 The Implied Correlation trade

In the case study below, we outline a practical demonstration of using the graph-guided generator. We demonstrate enhanced correlation pricing over raw implied correlation of option indices for the November expiry in 2024 that included the US Presidential election[4].

In figure 3, we examine the Profit and Loss (PnL) of a portfolio of correlation trades. Using a simple sizing strategy, we build a long-short flat correlation trade. For the generated model, we compare implied correlation premium (implied correlation minus the median generated realised correlation) in percentile terms across all the indices and generations. We average this value per index and we

[4]For implementation details see appendix A.2 and further details of correlation trading are outlined in appendix A. For a detailed formal introduction see [15].



(a) **(b)**

Figure 3: In figure 3a, PnL of the strategies is displayed in terms of correlation points (i.e., if we are long correlation, the PnL represents implied correlation - realised correlation). The red, blue and green lines represent cumulative PnL for the strategies using generated realised correlation, historical realised correlations and implied correlation only. The bars represent the PnL per index. In figure 3b, the y-axis represents PnL in correlation points while the x-axis the number of indices available to trade, ranging from 2 to 12. The boxplots indicate the PnL distribution of all combinations for our 3 strategies. The PnL from generated data outperforms both other methods over all combinations. The certainty of outperformance increasing as the number of possible indices to choose from increases.

then make a long-short basket and normalize size to 1.0 for both baskets. The *classical* strategy (blue curve and bars) replicates this approach but uses historical realised correlation levels. The *raw* strategy (green) is based on an index's implied correlation distance to median implied correlation of all indices. The strategy based on the generated correlations (red) outperforms both the classical and raw strategies. The generated approach accumulates approximately 27% and 63% greater profit than the classical and implied-only(raw) methods (see figure 3a). Replication of correlation swaps requires trading many instruments. To emulate this real-world constraint, we limit the number of indices we can trade. We examine the performance of all possible subsets to determine the robustness of the method, i.e., we look at the PnL from all strategies using all combinations of indices where we can trade from 2 to 12 indices (see figure 3b). The generated method outperforms the other approaches. The outperformance is clearer as the number of indices available to trade increases. When faced with going long or short one index (i.e., only 2 indices are available to trade), the boxplot body of the generative generated approach (leftmost red) remains above 0, indicating that for more than 75% of all cases, it correctly predicts the relative level of correlation premium between all pairs of indices.

4 Conclusion

In this work, we propose a novel approach to generating multivariate financial time series using graph-guided generation. This method contributes to graph-based generative modeling, demonstrating how to incorporate graph-guidance for high-dimensional temporal data. The generative model demonstrates success in generating time series and producing desired correlation structure. Importantly, it maintains hierarchical structure within the generated time series. The generative model offers a flexible and nearly off-the-shelf time series generator for practitioners. Lastly, we showcase a practical application, using the model to price correlation around the 2024 US presidential election.

As with any experimental work, there were many design choices. A full ablation study of these is beyond the scope of this work, but may garner useful insights going forward. One key question is the weighting of the loss, while the approximate equal weighting approach works well, a more systematic method would be worth investigating, e.g., adapting gradient normalization methods of [16, 17].

Similarly, we do not identify all clear effects (other than successful dependency generation) of the different relationship types on the generation process (appendix E.3). There is room for further investigation here which would help in general approaches to modeling complex multivariate time series.

Acknowledgements

This work was partly funded by Taighde Éireann – Research Ireland under grant numbers 18/CRT/6049 (H.C.) and 12/RC/2289 P2 (J.P.G.).

References

[1] Howard Caulfield and James P Gleeson. Systematic comparison of deep generative models applied to multivariate financial time series. *arXiv preprint arXiv:2412.06417*, 2024. 1

[2] Giuseppe Masi, Matteo Prata, Michele Conti, Novella Bartolini, and Svitlana Vyetrenko. On correlated stock market time series generation. *Proceedings of the Fourth ACM International Conference on AI in Finance*, 2023. 1

[3] Zhao Bai, Fangda Guo, Yuxin Xi, Zhuoming Zhu, Yu Guo, and Rongfang Bie. Timegae: A multivariate time-series generation method via graph auto encoder. In *International Conference on Database Systems for Advanced Applications*, 2024. URL https://api.semanticscholar.org/CorpusID:273232483. 1

[4] Mervyn A King and Sushil Wadhwani. Transmission of volatility between stock markets. *The review of financial studies*, 3(1):5–33, 1990. 1

[5] Xingyue Stacy Pu, Stephen Roberts, Xiaowen Dong, and Stefan Zohren. Network momentum across asset classes. *Stephen and Dong, Xiaowen and Zohren, Stefan, Network Momentum across Asset Classes (August 7, 2023)*, 2023.

[6] Jingda Yan and Jialin Yu. Cross-stock momentum and factor momentum. *Journal of Financial Economics*, 150(2):103716, 2023.

[7] Francis X Diebold and Kamil Yılmaz. On the network topology of variance decompositions: Measuring the connectedness of financial firms. *Journal of Econometrics*, 182(1):119–134, 2014. 1

[8] Aamir Sheikh. Barra's risk models. *Barra Research Insights*, pages 1–24, 1996. 1

[9] Thomas Nagler, Christian Bumann, and Claudia Czado. Model selection in sparse high-dimensional vine copula models with an application to portfolio risk. *Journal of Multivariate Analysis*, 172:180–192, 2019. 1

[10] Jean-Phillipe Bouchaud and Marc Potters. Financial applications of random matrix theory: a short review. 2015. 1

[11] Olivier Ledoit and Michael Wolf. Honey, i shrunk the sample covariance matrix. *UPF economics and business working paper*, (691), 2003. 1

[12] Marc Brockschmidt. Gnn-film: Graph neural networks with feature-wise linear modulation. In *International Conference on Machine Learning*, pages 1144–1152. PMLR, 2020. 1, 10

[13] Yifan Bao, Yihao Ang, Qiang Huang, Anthony KH Tung, and Zhiyong Huang. Towards controllable time series generation. *arXiv preprint arXiv:2403.03698*, 2024. 2

[14] Yujia Li, Kevin Swersky, and Rich Zemel. Generative moment matching networks. In *International Conference on Machine Learning*, pages 1718–1727. PMLR, 2015. 2

[15] Sébastien Bossu. *Advanced equity derivatives: Volatility and correlation*. John Wiley & Sons, 2014. 3, 6, 7

[16] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019. 4

[17] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018. 4

[18] Yuanrong Wang, Antonio Briola, and Tomaso Aste. Topological portfolio selection and optimization. In *Proceedings of the Fourth ACM International Conference on AI in Finance*, pages 681–688, 2023. 6

[19] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018. 10

[20] Louis Bachelier. Théorie de la spéculation. In *Annales scientifiques de l'École normale supérieure*, volume 17, pages 21–86, 1900. 10

[21] Adil Rengim Cetingoz and Charles-Albert Lehalle. Synthetic data for portfolios: A throw of the dice will never abolish chance. *arXiv preprint arXiv:2501.03993*, 2025. 11

[22] Hao Ni, Lukasz Szpruch, Magnus Wiese, Shujian Liao, and Baoren Xiao. Conditional sig-wasserstein gans for time series generation. *Mathematical Finance*, 2023. 17

A Correlation Trade

To trade correlation between a basket of stocks and its respective index with vanilla options, one would use a net theta (i.e., option greek that measures change in time value of option) zero portfolio to mimic correlation swaps. This is a form of dispersion trading (trading stock index implied volatility versus the implied volatility of the basket [15]).

A.0.1 Pricing the US Election.

A practical use case for multivariate financial time series is scenario risk or valuation. On the 5^{th} of November, the 45^{th} USA presidential election results were announced. The presidential election offers a moment in time to capture the power of expected correlations and to some degree volatility. The contrast of policies in presidential candidates in a close race can offer exceptional dispersion circumstances. We have demonstrated in our results how the graph-guided approach provides an extra tool for practitioners.

A.0.2 Creating the Election correlation expectations.

We take a very naive approach to determining how correlation based on a Trump victory would look. We look at the 10 trading days over which Trump had the greatest jump in election odds based on the most recent betting history using implied Betfair odds (see Figure 6). The underlying assumption is as follows, over the course of these 10 days the change in odds will be reflected in market movements. To add some variation to the correlation matrices, we bootstrap eight trading days without replacement to determine the expected correlation matrix, akin to the method demonstrated in [18].

A.1 Note on option data.

The data for pricing only includes options data as of the 30^{th} of August 2024. The November expiry options is not included in some option names, so where needed we interpolate through the forward volatility curve the November volatility smile. Option prices used are recorded as of 17:00 GMT.

A.2 Model Adjustments

We introduce two simple adjustments to the generator to allow for pricing the November expiry. We ensure the conditioned time series includes returns of randomly drawn earnings events sizes, e.g., if stock S is announcing earnings on the 25^{th} of October, we sample from a distribution based on historical earnings moves for stock S and adjust the conditioning for this. Similarly, when generating, we use the same process to ensure the stock S move is amended for earnings. On the day of the election, all stock returns will be adjusted by an implied move (estimated from the option data).

We generate all returns based on the dependence structure of the randomly sampled conditioning time series until the day of the election, after which we use the bespoke correlation matrices to generate the final 8 days until expiration.

A.3 Correlation Trade Experiment Assumptions

As we do not have access to correlation swaps, we use pricing from vanilla options. We have limited price data, accordingly we make some simplifying assumptions:

- We can represent the indices reasonably with a subset of underlying stocks (See section A.4 for the tracking error of the indices used).
- All options are European (we use this assumption to derive implied volatility and option Greeks).
- Linear interpolation over term structure and smile is accurate.
- Events are independent, instantaneous and are treated as normally distributed across the volatility surface.
- We ignore potential dividends around the event.
- We assume all earnings over the October and November expiries can be drawn from a stock specific log normal distribution based on earnings moves from the previous 4 years.
- To derive volatility for expiration's where there are no listed options, we use linear interpolation based on an *earnings-free* forward volatility curve,i.e. we remove the volatility of the earnings event from the term structure.
- To derive the election earnings premium, we use this *earnings-free* forward curve.
- On the index side, we assume no other events than the election, e.g., we ignore the FOMC meeting on the 7th of November.
- Implied correlation (as per [15]) can be estimated with $(\frac{\sigma_{index}}{\overline{\sigma_{basket}}})^2$, where σ_{index} is the implied volatility of the index, $\overline{\sigma_{basket}}$ is the average implied volatility within the basket.

Naturally this induces a lot of caveats, however the primary purpose is demonstration. For practitioners, adjusting these assumptions (or exposures to some assumptions) should be relatively straightforward.

A.4 Index Tracking Error

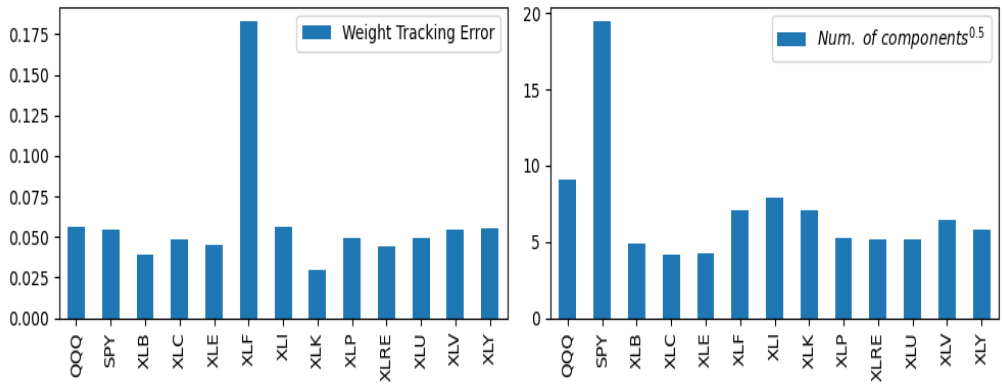


Figure 4: The figure on the left represents the tracking error for each index, i.e., given our 392 stocks , what is the difference between 100% index weighting and the the weighting of the stocks we have included in our analysis. XLF (Financial ETF) has the largest tracking error. Our universe typically represents over 95% for each of the other indices. The figure on the right , represents the square root of the number of stocks within our universe which is part of the representative index i.e., a value of 5 indicates that we have 25 stocks within that index. With the tracking error, this provides information on how diverse an index is. SPY is the most diverse and the sector ETFs, XLC (communications) and XLE (energy) are the least diverse (most concentrated).

A.5 Trump vs. Harris Betfair odds

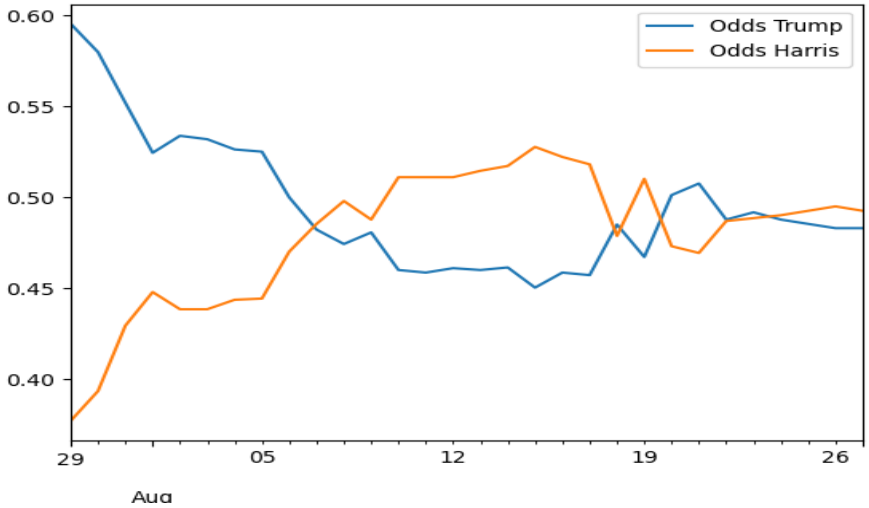


Figure 6: Implied probability of a Trump (Blue) vs. Harris (Orange) Presidential election victory from BetFair throughout August 2024.

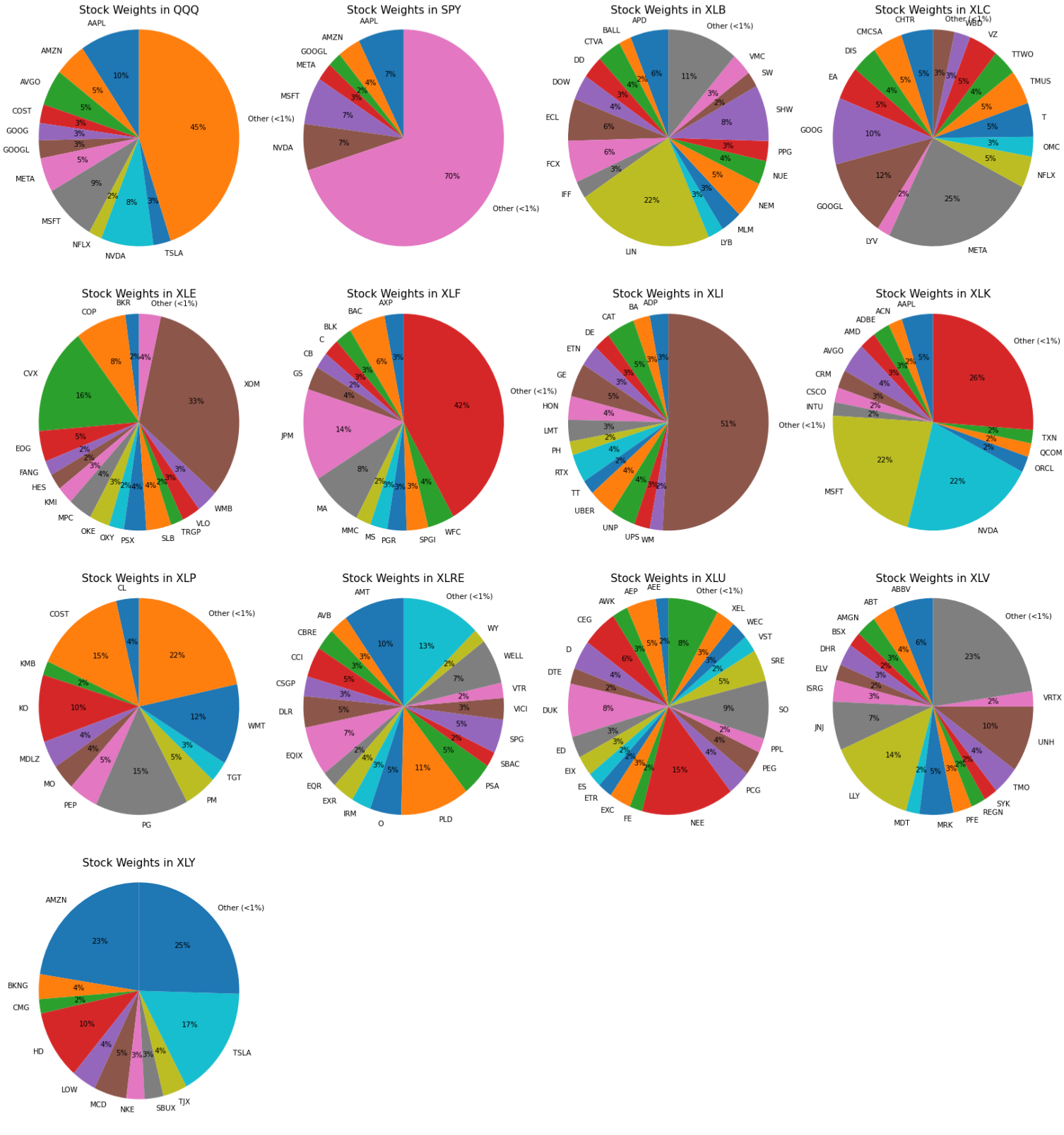
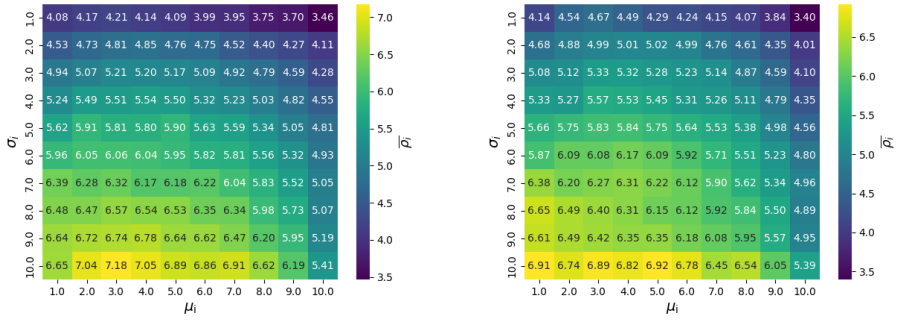


Figure 5: Renormalised component weights for the indices. Components with weight less than 1% are included in label *other*.

B FiLM Inner Architecture

FiLM (Feature-wise Linear Modulation) [19] was first introduced to allow conditioning via learned scale and shift parameters of neural networks given external information e.g., visual processing based on a question. It was then extended to graph neural networks [12] where the expressivity of the convolution for graph neural networks was emphasized.

FiLM offers a bridge towards the earliest mathematical modelling of financial time-series [20]. FiLM learns a β and γ which are analogous to both μ and σ (which represent drift and scale respectively) in financial time series. We also find strong evidence empirically (figure 7) that both drift and scale have a strong relationship with expected correlation.



(a) 10 Trading Days **(b)** 40 Trading Days

Figure 7: The heatmap represents the correlation deciles i.e., 7 (bright colour) implies the average correlation for a stock is the 70th percentile. The y-axis represents the decile of the stock's volatility and the x-axis represents the decile of the stock's drift (over 10 and 40 days respectively). There is a clear positive relationship between diffusion and correlation. The relationship with drift is more nuanced, where there seems to be a leverage effect, particularly as volatility increases.

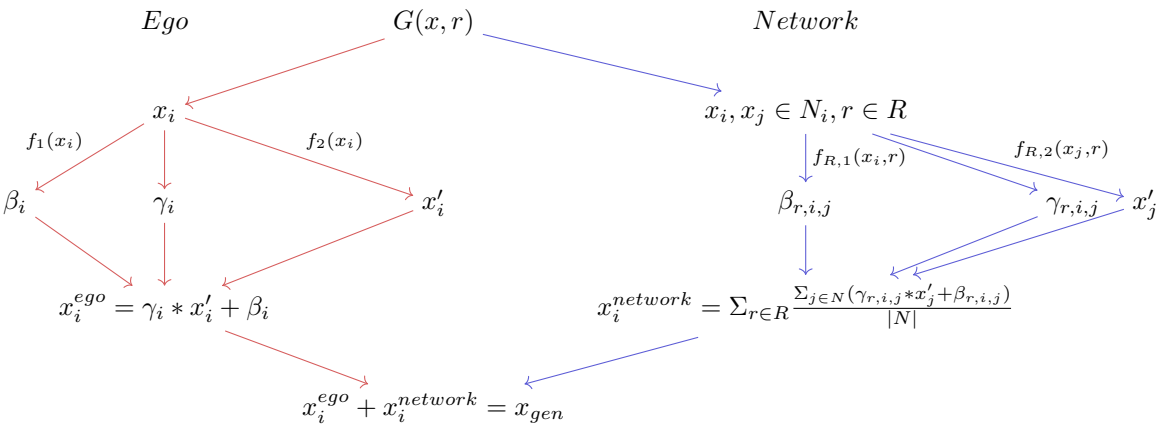


Figure 8: The FiLM convolution [12] involves a node centric (ego) transform and a neighbourhood (N) and edge-type (r) based transformation. The conditional data and concatenated sampled noise is represented by x_i. The Ego transform (left hand side) performs two operations. $f_1(x_i)$ outputs a scale (γ_i) and shift (β_i) for the transformation of the ego features $f_2(x_i)$. The network-based transformations (right hand side) includes similar operations for neighbour-based features dependent on edge type (R). Neighbourhood representations are averaged and added to ego node features.

C Top-k Loss

Let $C_{\text{desired}} \in \mathbb{R}^{B\times N\times N}$ and $C_{\text{generated}} \in \mathbb{R}^{B\times N\times N}$ be batches (B) of the desired and generated correlation matrices for N instruments, respectively. We define the top-k structural correlation loss as,

$$\mathcal{L}_{\text{Top-k}} = \sum_{k\in\mathcal{K}} \frac{1}{\log(k)+1} \cdot \frac{1}{B}\sum_{b=1}^{B}\frac{1}{N}\sum_{i=1}^{N}\left\| M_k^{(b,i)} - \hat{M}_k^{(b,i)} \right\|_1,$$

where $\mathcal{K} = \{1, 5, 10, \ldots, 100\}$, $M_k^{(b,i)}$ is a binary mask of top-k absolute correlations in row i of $C_{\text{desired}}^{(b)}$, $\hat{M}_k^{(b,i)}$ is the same for $C_{\text{generated}}^{(b)}$ and $\|\cdot\|_1$ represents the L1 norm. Using varying values of k ensures that the loss is balanced across different correlation scales.

We ran an ablation test (over 5 different seeds) to determine how conditional moments are affected by the inclusion of the top-k loss. In general, we saw that higher conditional moments, such as skewness and kurtosis of generated time series, are less accurate. In contrast, inclusion of this loss improved both conditional volatility and the dependency structure, in particular for the index nodes and supernodes. Indices represent the strongest correlations for many stocks (normal nodes). This loss provides a more concerted way to learn the hierarchical dependency structure.

D A Note on Supernodes

We found that when the supernodes and their dependency structure are excluded, generated data has unrealistic aggregate behaviour of stock baskets. More specifically, the aggregate variance learned by the generator is too high variance. Including these constraints could also address the issue with synthetic data for portfolio optimization purposes as described in [21]. The generator matches both the behaviour of the financial time series at both a micro (idiosyncratic risks) and a macro (portfolio risks) level.

E Additional Results

Below we outline further the performance of the graph-guided generator in terms of its capability in achieving desired dependencies. We also examine more specifically *supernodes* (section E.2.1) and compare their dependency to generated dependencies from true index nodes (i.e., that are treated as normal nodes). The dependency performance from the true index nodes can be also viewed as an inspection of the model's performance on a smaller basket (albeit with stronger than normal base correlation)[5]. We outline preliminary analysis of the scale and shift parameters learned per edge type (section E.3). Lastly, we detail further results from the empirical experiment (section E.4).

E.1 Measure

To further examine how well the generated data matches the desired correlation structure we inspect the differences of the averaged Jaccard Index between past, future and generated correlation networks.

The Jaccard Index is defined as follows; let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be two undirected networks on the same set of nodes V, where edges are defined by thresholding correlation matrices at some threshold τ. The edge sets are:

$$E_i = \{(u, v) \in V \times V \mid \rho_i(u, v) \geq \tau\}, \quad i = 1, 2,$$

where $\rho_i(u, v)$ is the correlation between nodes u and v in network G_i.

The Jaccard Index $J(G_1, G_2)$ is then defined as:

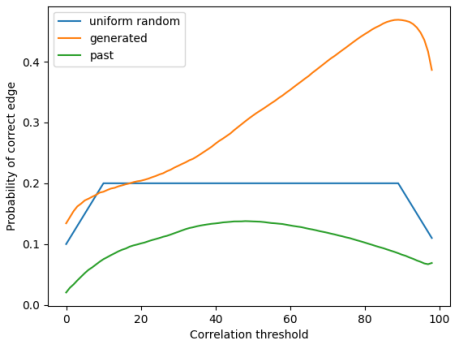
$$J(G_1, G_2) = \frac{|E_1 \cap E_2|}{|E_1 \cup E_2|} \tag{2}$$

where $|\cdot|$ denotes the cardinality (number of edges). The proportion of common edges relative to the total edges present in either network after thresholding is quantified by $J(G_1, G_2)$.

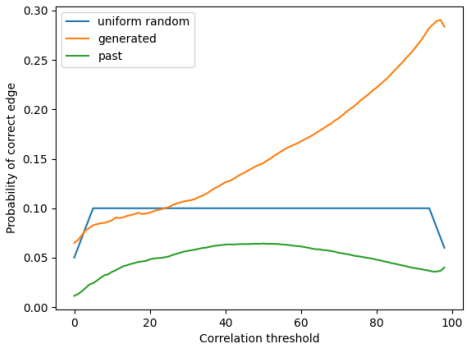
[5]The values in the figures below are calculated over generations for the entire test dataset unless otherwise specified.

E.2 Dependency Performance

Figures 9 to 15 further detail the performance of the graph-guided generator in capturing the desired dependency structure.

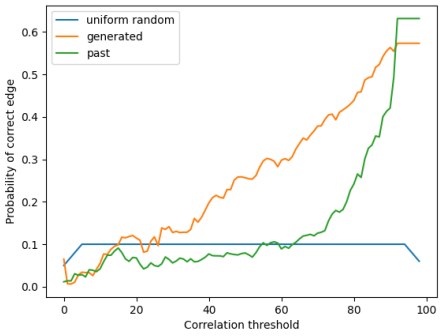


(a) Jaccard Index within a 10 correlation point threshold.

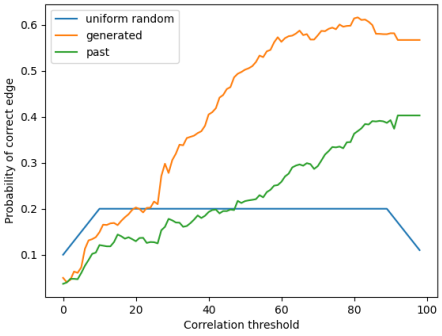


(b) Jaccard Index within a 5 correlation point threshold.

Figure 9: In Figure 9 we demonstrate the probability of generating a correct correlation within a certain threshold i.e., $p(|\rho_{(x,y)^{desired}} - \rho_{(x,y)^{actual}}| < thresh)$. The blue line indicates uniform probability of generating the desired correlation. In general we see that the generated (orange) series are more accurate for higher correlation values, better than random chance and reaching close to 50% (25%) chance of generating the desired dependency structure within the 10 (5) correlation point threshold. The green line indicates how similar the correlation of the conditioning data is with the desired correlation structure (typically less than random).

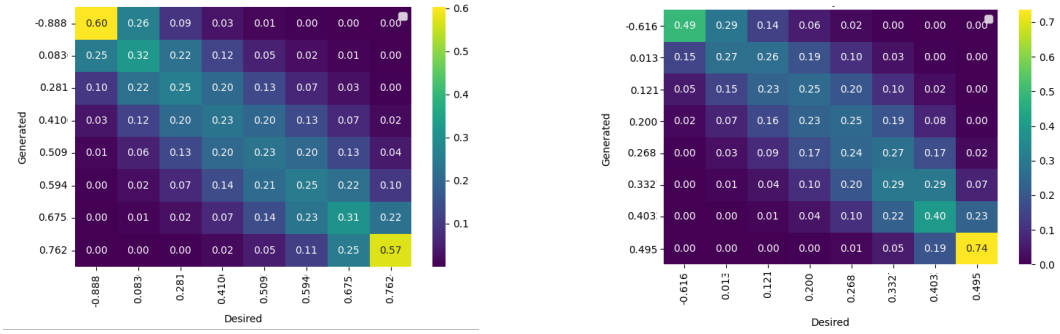


(a) Jaccard Index for Supernodes within a 10 correlation point threshold.



(b) Jaccard Index for Supernodes within a 5 correlation point threshold.

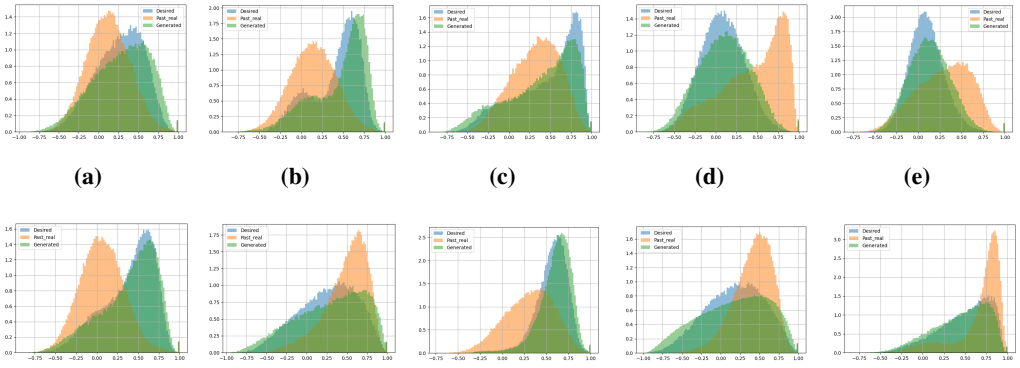
Figure 10: Similar to figure 9, figure 10 shows the threshold based Jaccard index for the desired supernode structure. In general, the generated relationships are quite accurate and improves with the level of correlation (as above). This also demonstrates that the conditional correlation structure for these supernodes is more stable at higher correlation levels.



(a) 20 Day Generation Correlation Probability Matrix

(b) 60 Day Generation Correlation Probability Matrix

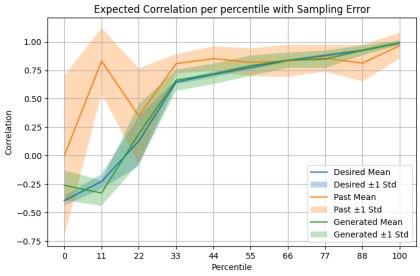
Figure 11: The matrices in figure 11 represent the probability of a desired correlation octile (midpoint value on x-axis) coinciding with a generated correlation octile (midpoint value on y-axis). The diagonal represents the true positive rate. The octiles are based on the desired correlation matrix. In general we see near total probability in each octile and it's neighbours. This result improves with longer generation periods (figure 11b).



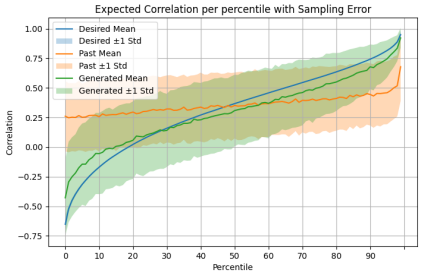
(a) **(b)** **(c)** **(d)** **(e)**

(f) **(g)** **(h)** **(i)** **(j)**

Figure 12: The histograms in figure 12 represent the conditional (orange), desired (blue) and generated (green) correlations for different generations. The generated correlations match the desired correlation distributions quite well. The generator is able to match a wide variety of correlation distributions.



(a) The cumulative density function (CDF) of the index nodes correlation structure.

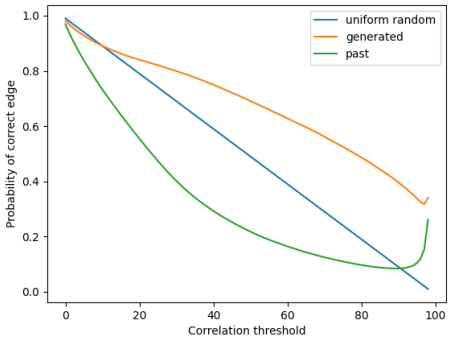


(b) The cumulative density function of the generated time series correlation structure.

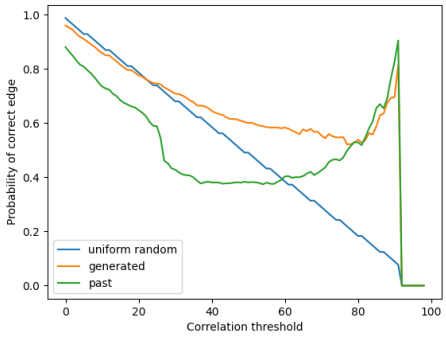
Figure 13: Figure 13 demonstrates how closely aligned the generated correlation matrices are to the desired correlation matrices. The x-axis represents the percentile level and the y-axis the corresponding correlation value. The blue (desired) and green (generated) CDFs track closely, while the orange (past) provides a baseline (random case) to the desired correlations.

E.2.1 Supernode Behaviour

Figures 14 and 15 reflect supernode behaviour i.e., how the graph-guided generator performs at capturing aggregate dependency structure.

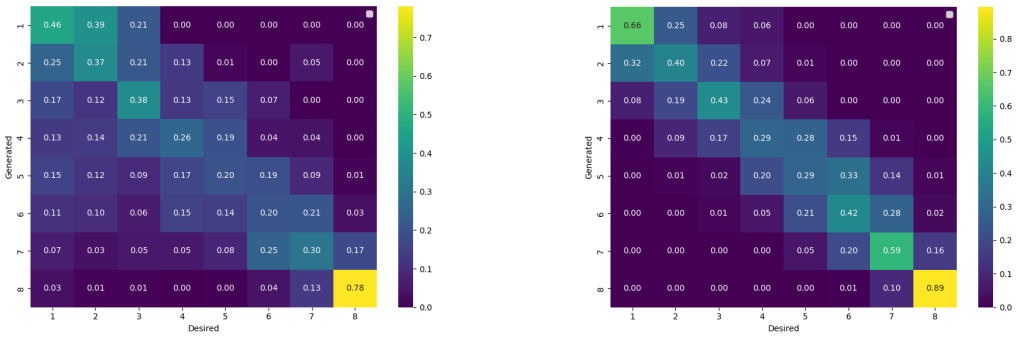


(a) General Jaccard Index



(b) Supernode Jaccard Index

Figure 14: Figure 14a displays the Jaccard Index of all generated instruments, while figure 14b displays the Jaccard Index of the supernodes. The y-axis is the Jaccard Index number (level of agreement, higher is better) and the x-axis is the correlation percentile (i.e., edges only exist above this correlation threshold). The results echo those in figures 9 to 13, both figures demonstrate superior performance (generated, orange line) to both random chance (blue line) and using conditioning data correlation structure (green line). The supernode dependency structure has a minium value of approximately 60%, i.e., in the worst case the generated model gets 60% success rate of edges in correlation threshold networks.



(a) Supernode Correlation Probability.

(b) Generated index correlation probability.

Figure 15: As in figure 11, Figure 15 shows the probabilities of generating an edge correlation in the correct octile (values on the diagonal). On the left is the supernode dependency structure and on the right is the generated index structure. While the supernode structure does not perform as perfectly as the index nodes, it still performs well. The correlation distribution for index nodes is a narrower distribution, octiles four to eight all have average values above 60% correlation. The primary difference in performance is in the negative correlations (octiles 1 and 2), the supernode generations finds it more difficult to capture the aggregated dependency as accurately.

E.3 Scale (γ) and shift (β) for different relationships

The histograms in figure 16 represent the the average β and γ learned from the last layer of the generative model over different seeds (i.e., the conditional scale and shift parameters generated from the FiLM convolution). While averaging the values in the last layer is an oversimplification of the mechanism at hand, it is interesting to examine nonetheless. For a more detailed analysis, more context is required i.e. conditioning data, volatility etc.

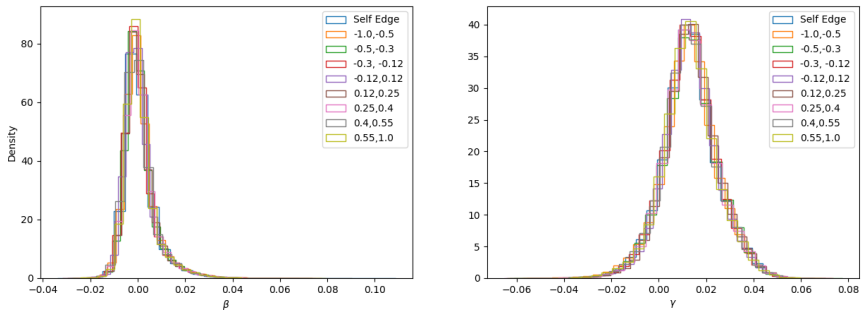
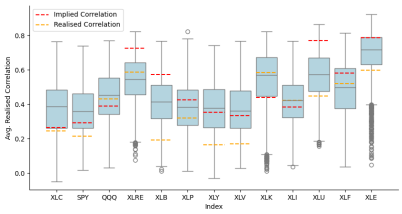


Figure 16: The histograms in figure 16 represent the distribution of the β (left figure) and γ (right figure) values. To interpret these values, we make two assumptions. Firstly, the intermediate representation of the time series acts similarly to financial time series in general. Secondly, our learned parameters β and γ reflect a drift and volatility effect respectively. Both parameters have similar distribution over all edge types. The β parameter is approximately log-normally distributed with a mean around zero, i.e., on average, there is no drift effect. The γ is more normally distributed with on average a positive effect. The γ values indicate that on *average* we do see positive spillover from the both ego node (i.e., autocorrelation) and from neighbouring nodes. To determine the true effect of β and γ, we would require the conditional data and perhaps a better representation of the intermediate layer.

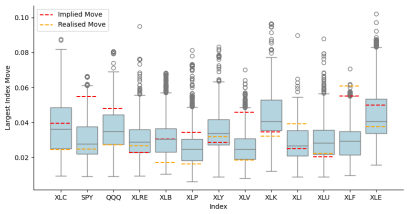
E.4 November Expiry Correlation Trade Results

Below we include further results from the empirical application outlined in section 3.2 in the main text. We examine the generated correlations versus both implied and realised. We also examine the

max moves generated by each supernode versus the implied event size and realised event size (figure 17). In figure 18, we examine the realised correlation generated before and after the US election event. Lastly in figure 19, we demonstrate how well the generated data fits the true realised implied correlation premium vs. using only the implied level and using the implied level versus historic realised levels.



(a) Generated Realised Correlation



(b) Generated Max Absolute Daily Move

Figure 17: In figure 17a we can see that the realised correlations in general match up well with the realised (dashed yellow lines). The implied correlation tends to have a premium over the realised correlation. In figure 17b, we compare the roughly estimated *implied* event move with the true largest index move over the period, with the distribution of *supernode* moves. The body of the distributions also track realisations quite well with the exception of XLF (Finance sector ETF) . Tracking error (i.e., the difference between the true ETF and the approximation from our universe of stocks) may be the potential source of error for XLF. XLF is the index with the largest tracking error, see figure 4 for more details.

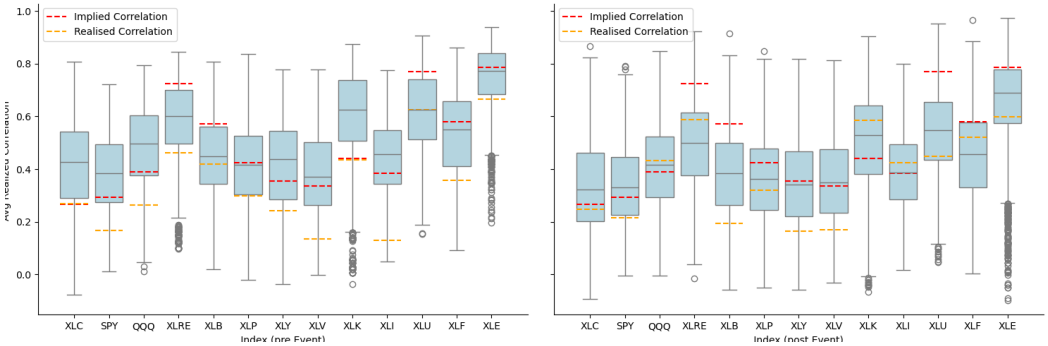


Figure 18: Figure 18 displays the distribution of pre (left) and post (right) election realised correlation for each index from the generated time series. The change in realised correlations across indices is mixed. The *true* realised correlations post-election match better with the generated data, the majority of realised correlations are within the body of the distributions. The pre-event distributions (i.e., completely random correlation distributions) tend to overestimate the *true* realised correlation before the event. A more informed use of conditional correlations prior to the event may improve this, e.g., using implied volatility closer to event time or amending pre-event average correlation levels to match those of previous election levels.

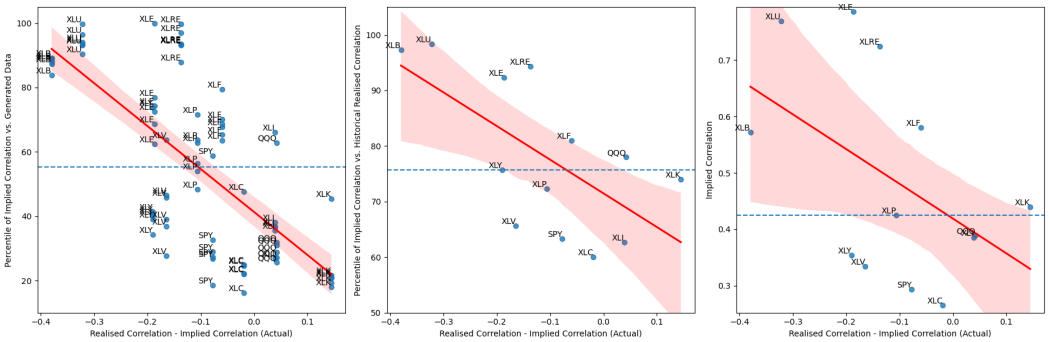


Figure 19: The left plot of figure 19 compares the percentile of implied correlation per generated distributions of realised correlations (y-axis) vs. the realised correlation premium (x-axis) over the expiry. The middle figure compares the percentile of implied correlation per historical distributions of realised correlations (y-axis) vs. the realised correlation premium (x-axis) over the expiry. The right figure plots raw implied correlation (y-axis) vs. the realised correlation premium (x-axis). The red line represents best fit and the shaded area represents 95% confidence intervals using bootstrapping. The dashed line represents the median values across all the indices. All figures highlight greater likelihood of a negative realised correlation for higher percentiles (raw implied correlation). However, without calibration we find that generated percentiles would have proved a more robust method for bet sizing correlation trades, see figure 3a.

F Pseudo Code

In this section we provide pseudo codes for algorithms described in the main text. Algorithm 1 refers to the base graph neural network structure. Algorithm 2 outlines the generation process. Algorithm 3 details the training procedure. Algorithm 2 is inspired by the method defined in [22].

Algorithm 1 NET-GNN with FiLMConv Layers

Input: Node features x (i.e., the concatenated feature vector and sampled noise), edge indices e, edge types r
Initialize GNN with R FiLMConv layers
for $r = 1$ to R **do**
    $x \leftarrow \text{PReLU}(\text{FiLMConv}_r(x, e))$
end for
Output: Final node representation x

Algorithm 2 AR-FNN

Input: Graph data $\mathcal{G} = (x, e, r)$, latent noise z, shared noise z_{shared}, correlated edge noise z_{edge}, jump noise z_{jump}, edge types r, event volatility v (optional), post-event edges e^{post},post event edge types r^{post}, event time T_{event}.
Project node input: $x \leftarrow \text{PReLU}(\texttt{lin\_proj}(x))$
Apply FiLMConv GNN to process structural graph: $h \leftarrow \text{NET}(x, e)$
Concatenate node features: $h \leftarrow [h \,\|\, z \,\|\, z_{\text{shared}}]$
if event-based conditioning is enabled, $T \geq \text{T}_{event}$ **then**
    Apply GNN to v using edges e^{post} and types r^{post}
else
    Apply GNN to v using edges e and types r
end if
Output: Generated data point $\hat{x}$

Algorithm 3 Training Procedure for Graph-Guided Generator

Initialize generator $G = \text{AR-FNN}(\cdot)$ with input/output dims and hyperparameters
Initialize optimizer G_{opt} (Adam) and scheduler
for epoch $= 1$ to num_epochs **do**
  for each batch in training loader **do**
    Randomly sample time window $q \in \{5, 10, 20, 40, 100\}$
    Generate latent vectors: z, z_{jump}, and z_{shared}
    Compute current correlation matrix $\text{corr}(x_{\text{real}})$
    Sample desired correlation matrix C_{desired}
    if correlation conditioning is enabled **then**
      Transform noise z to match C_{desired} using correlated noise model
    end if
    Compute edge types from desired correlation matrix.
    if events enabled **then**
      Repeat steps above to compute post-event edge attributes and correlations
    end if
    Select top-k strongest correlations using C_{desired}
    Extract valid edge indices and values $\rightarrow (\text{edge\_index}, \text{edge})$
    if events enabled **then**
      Also extract valid edge indices and values for post-event edges.
    end if
    Input x_{real}, z, and edge data to G to generate x_{fake}
  end for
end for