Quasi-random sampling for multivariate distributions via generative neural networks

Marius Hofert (with Avinash Prasad and Mu Zhu)

2021-01-13

UNIVERSITY OF WATERLOO

Department of Statistics and Actuarial Science

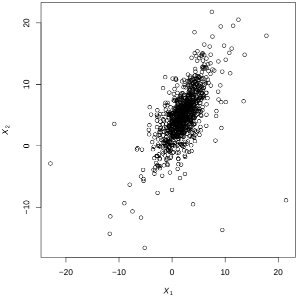
1 Classical copula modeling

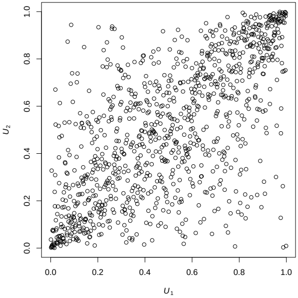
- **General goal:** Modeling a df H or $\boldsymbol{X} \sim H$ with continuous margins $F_1, \dots, F_d$ (for possibly high-dimensional, computational applications).
- **Examples:**
  - Static: $\boldsymbol{X}$ models a joint loss in a risk management context; e.g., $\mu = \mathbb{P}(\boldsymbol{X} > \boldsymbol{x})$ or $\mathrm{ES}_\alpha(S) = \mathbb{E}(S \,|\, S > F_S^{-1}(\alpha))$ for $S = X_1 + \dots + X_d$.
  - Dynamic: $\boldsymbol{X}$ models joint innovations of an ARMA–GARCH process or increments of dependent geometric Brownian motions
- **Sklar's Theorem:**
  - Analytical: $H(\boldsymbol{x}) = C(F_1(x_1), \dots, F_d(x_d))$ for a copula C, a distribution function with $\mathrm{U}(0,1)$ margins.
  - Stochastic: $\boldsymbol{X} = (F_1^{-1}(U_1), \dots, F_d^{-1}(U_d)) \sim H$ (quantile transformation) and $\boldsymbol{U} = (F_1(X_1), \dots, F_d(X_d)) \sim C$ (probability transf.).
- **Why:** Useful under asymmetric information (margins known, dependence unknown) or from a computational point of view (e.g., estimation).

- **Statistics:** Instead of $\boldsymbol{X}$, we now have $X = (\boldsymbol{X}_1^\top, \dots, \boldsymbol{X}_n^\top)^\top \in \mathbb{R}^{n \times d}$.
  1) Given X, compute the *pseudo-observations* $U = (\boldsymbol{U}_1^\top, \dots, \boldsymbol{U}_n^\top)^\top \in \mathbb{R}^{n \times d}$ with

$$U_{ij} = \hat{F}_{n,j}(X_{ij}) = \frac{1}{n+1} \sum_{k=1}^{n} \mathbb{1}_{\{X_{kj} \le X_{ij}\}} = \frac{R_{ij}}{n+1},$$

     where $R_{ij} = \mathrm{rank}(X_{ij})$ among the component sample $X_{1j}, \dots, X_{nj}$.





  2) Based on U, one needs to fit, test and select an adequate copula C.
  3) Simulate $\boldsymbol{U}_1, \dots, \boldsymbol{U}_{n_{\text{gen}}}$ from the fitted C and estimate, for example:

- $F_S^{-1}(u)$ where $S = X_1 + \cdots + X_d = F_1^{-1}(U_1) + \cdots + F_d^{-1}(U_d)$

  Examples (QRM):

  – Value-at-risk $\mathrm{VaR}_\alpha(S) = F_S^{-1}(\alpha)$ of the total loss S

  – Expected shortfall $\mathrm{ES}_\alpha(S) = \mathbb{E}(S \,|\, S > F_S^{-1}(\alpha))$

- Expectations $\mu = \mathbb{E}(\Psi_0(\boldsymbol{X})) = \mathbb{E}(\Psi(\boldsymbol{U}))$ where $\Psi(\boldsymbol{u})$ is given by $\Psi_0(F_1^{-1}(u_1), \dots, F_d^{-1}(u_d))$ and so $\mu \approx \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} \Psi(\boldsymbol{U}_i)$.

  Examples (MC):

  – With $\Psi(\boldsymbol{u}) = \mathbb{1}_{\{u_1 > u, \dots, u_d > u\}}$ we obtain that

  $$\mu = \mathbb{E}(\Psi(\boldsymbol{U})) = \mathbb{P}(U_1 > u, \dots, U_d > u),$$

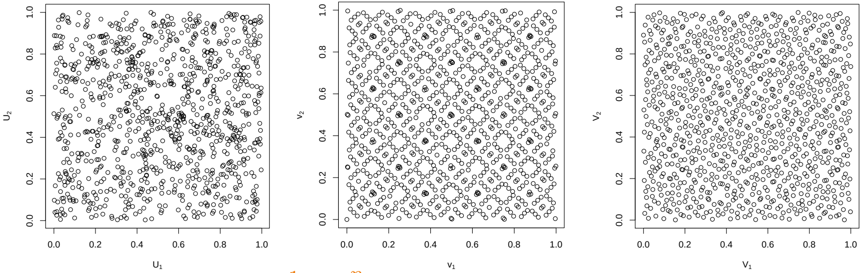
  so exceedance probabilities, e.g., the probability of a flood over multiple dikes or joint losses in a stock portfolio.

  – With $\Psi(\boldsymbol{u}) = \max\{(F_1^{-1}(u_1) + \cdots + F_d^{-1}(u_d))/d - K, 0\}$ we obtain that $\mu = \mathbb{E}(\Psi(\boldsymbol{U}))$ is the expected payoff of a European basket call option.

- **Problems (with this classical modeling approach):**
  1) Step 2) above: Finding an adequate copula model C for the pseudo-observations, especially in more than two dimensions.
  2) Large variance $\mathrm{var}(\frac{1}{n_{\text{gen}}}\sum_{i=1}^{n_{\text{gen}}}\Psi(\boldsymbol{U}_i))$ of the Monte Carlo estimator under rare-event simulation.
- **Ideas:**
  1) Use (specific) neural networks (NNs) ($\Rightarrow$ flexible dependence).
  2) Use quasi-random sampling for such NNs ($\Rightarrow$ variance reduction).
- **Outline:**
  - We first address 2), under dependence.
  - We then study how NNs can be random number generators (RNGs) from copulas, so 1).
  - We then address how NNs can be quasi-RNGs (QRNGs).
  - For simplicity, we focus on the case where $F_1 = \cdots = F_d$ are $\mathrm{U}(0,1)$ in what follows (except for an ES_α example).

2 Quasi-random numbers for copulas

- If $C = \Pi$ (independence), pseudo-random numbers (PRNs) can be replaced by quasi-random numbers (QRNs) to reduce the variance.
- QRN sequences are low-discrepancy sequences $P_n = \{\boldsymbol{v}_i\}_{i=1}^n$ (middle) with $D^*(P_n) = \sup_{\boldsymbol{z}\in(0,1]^d} \left| \frac{\#\{i:\ \boldsymbol{v}_i\in[\boldsymbol{0},\boldsymbol{z})\}}{n} - \lambda([\boldsymbol{0},\boldsymbol{z}))\right| \in O(n^{-1}\log^d n)$.
- **Example:** $\boldsymbol{U}_1,\dots,\boldsymbol{U}_{n_{\text{gen}}}$ (left), randomized Sobol' $\boldsymbol{V}_1,\dots,\boldsymbol{V}_{n_{\text{gen}}}$ (right):



- The *RQMC estimator* $\frac{1}{n_{\text{gen}}}\sum_{i=1}^{n_{\text{gen}}} \Psi(\boldsymbol{V}_i)$ is unbiased, fast and its variance can be estimated (from repeated randomizations).

- **Question:** How can we obtain QRNs from a general copula C?
- **Idea:** Could define $D^*_C(P_n) = \sup_{\boldsymbol{z}\in(0,1]^d}\left|\frac{\#\{i:\ \boldsymbol{v}_i\in[\boldsymbol{0},\boldsymbol{z})\}}{n} - \mathbb{P}(\boldsymbol{U}\in[\boldsymbol{0},\boldsymbol{z}))\right|$, but this does not lead to a construction principle. However, one can transform $\boldsymbol{V}_1,\dots,\boldsymbol{V}_{n_{\text{gen}}}$ to samples from C; see Cambou et al. (2017).
- **Inverse Rosenblatt transform $\mathcal{R}^{-1}$:** Bijection to transform $\boldsymbol{U}'\sim \mathrm{U}(0,1)^d$ to $\boldsymbol{U}\sim C$ (known as *conditional distribution method (CDM)* for sampling, a generalization of the inversion method to $d>1$):

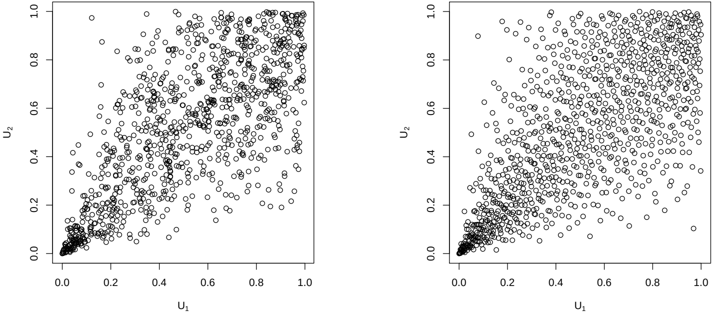
$$\begin{aligned}
U_1 &= U_1',\\
U_2 &= C^{-1}_{2|1}(U_2'\,|\,U_1), \quad \text{(compare with } X_2 = F_2^{-1}(U_2')\sim F_2\text{)}\\
U_j &= C^{-1}_{j|1,\dots,j-1}(U_j'\,|\,U_1,\dots,U_{j-1}), \quad j\in\{2,\dots,d\}.
\end{aligned}$$

- Formula for the implementation (which needs to be inverted!):

$$C_{j|1,\dots,j-1}(u_j\,|\,u_1,\dots,u_{j-1}) = \frac{\mathrm{D}_{j-1,\dots,1}\,C_{1,\dots,j}(u_1,\dots,u_j)}{\mathrm{D}_{j-1,\dots,1}\,C_{1,\dots,j-1}(u_1,\dots,u_{j-1})}.$$

$\Rightarrow$ Known analytically tractable $C^{-1}_{j|1,\dots,j-1}$: Normal, t, Clayton.

- **Example:** 1000 PRNs (left) and QRNs (right) from a Clayton copula.

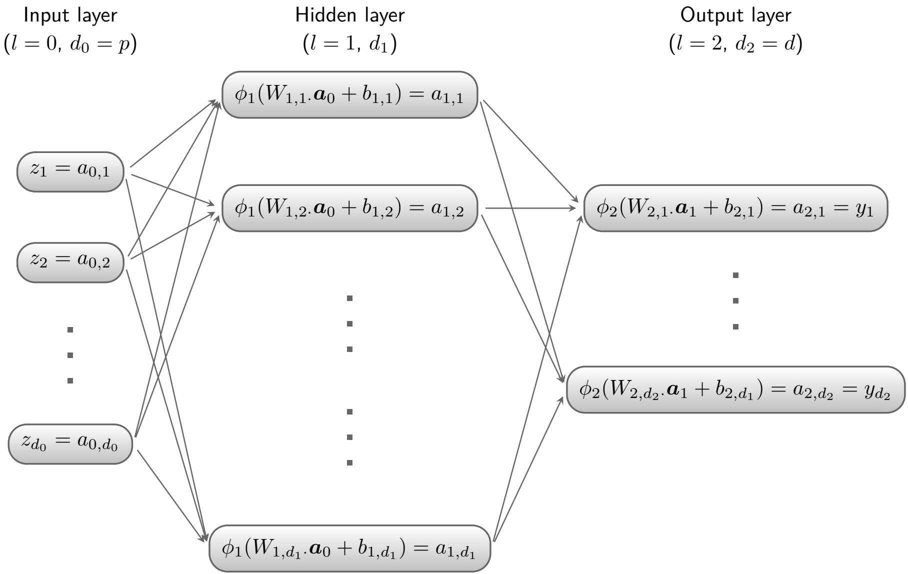


- **Disclaimer:** Even if not visible, there may be a variance reduction effect.
- For large j, evaluating $C^{-1}_{j|1,\dots,j-1}$ is time-consuming even for normal, t, Clayton.
- For Gumbel copulas, $C^{-1}_{j|1,\dots,j-1}$ is not tractable.
- See Cambou et al. (2017) for a different approach, but conceptually (first component $\Rightarrow$ frailty) and numerically challenging (α-stable qf).

3 PRNs for copulas via GMMNs

- **Idea**: To address Problem 1), we consider
$$\mathbb{E}(\Psi(\boldsymbol{U})) \approx \frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} \Psi(\boldsymbol{U}_i), \quad (1)$$
$$\boldsymbol{U}_i = f_{\hat{\boldsymbol{\theta}}}(F_{\boldsymbol{Z}}^{-1}(\boldsymbol{U}_i')), \quad i = 1, \dots, n_{\text{gen}}, \quad (2)$$
where
  - $\boldsymbol{U}_1', \dots, \boldsymbol{U}_{n_{\text{gen}}}' \overset{\text{ind.}}{\sim} \mathrm{U}(0,1)^p$ (later: $\boldsymbol{V}_1, \dots, \boldsymbol{V}_{n_{\text{gen}}}$; randomized Sobol');
  - $F_{\boldsymbol{Z}}^{-1}(\boldsymbol{u}) = (F_{Z_1}^{-1}(u_1), \dots, F_{Z_p}^{-1}(u_p))$ maps to a *prior distribution* (Φ);
  - $f_{\hat{\boldsymbol{\theta}}}$ is a trained generative neural network $f_{\boldsymbol{\theta}}$ (GMMNs).
- Training of $f_{\boldsymbol{\theta}}$ learns a map from pseudo-random numbers from $\boldsymbol{Z}$ (here: $\mathrm{N}_p(\boldsymbol{0}, I_p)$) to pseudo-random numbers from $\boldsymbol{U} \sim C$.
- This is similar to $\mathcal{R}^{-1}$, but computationally simpler to evaluate.
- **Errors:**
  1) Monte Carlo error (1) (can be made arbitrarily small by the SLLN)
  2) Neural network (NN) "bottleneck" (2) (small if NN trained correctly)

3.1 What are NNs?



- **Hyperparameters** (fixed before training):
  - The ϕ_l's are the *activation functions* (e.g., *ReLU* $\phi_l(x) = \max\{0, x\}$, *sigmoid* $\phi_l(x) = 1/(1 + e^{-x})$).
  - Number of layers, number of neurons per layer, number of epochs in training and batch size (more later).
- **Parameter vector:**
  $\boldsymbol{\theta} = (W_1, \dots, W_{L+1}, \boldsymbol{b}_1, \dots, \boldsymbol{b}_{L+1})$ consists of *weight matrices* and *biases* (initialized randomly and with zeros, resp.).
- $\boldsymbol{\theta}$ is fitted (or: the NN is *trained*) by stochastic gradient descent based on a *cost function* $E(U, Y)$ computed between (a subset of)
  - the training data $U = (\boldsymbol{U}_1^\top, \dots, \boldsymbol{U}_{n_{\text{trn}}}^\top)^\top$ from C (= target) and
  - the outputs $Y = f_{\boldsymbol{\theta}}(Z)$ of the NN from the prior sample $Z = (\boldsymbol{Z}_1^\top, \dots, \boldsymbol{Z}_{n_{\text{trn}}}^\top)^\top$.
- Often E is taken as the scaled MSE $E(U, Y) = \frac{1}{2n_{\text{trn}}} \sum_{i=1}^{n_{\text{trn}}} \|\boldsymbol{u}_i - \boldsymbol{y}_i\|_2^2$
  $\Rightarrow$ Fails to learn the map from Z to U properly.

3.2 What are GMMNs?

- *Generative moment matching networks (GMMNs)* use as $E(U, Y)$ the sample *maximum mean discrepancy* $\mathrm{MMD}(U, Y)$

$$\sqrt{\frac{1}{n_{\mathsf{trn}}^2}\sum_{i_1=1}^{n_{\mathsf{trn}}}\sum_{i_2=1}^{n_{\mathsf{trn}}}\bigl(K(\boldsymbol{U}_{i_1},\boldsymbol{U}_{i_2})-2K(\boldsymbol{U}_{i_1},\boldsymbol{Y}_{i_2})+K(\boldsymbol{Y}_{i_1},\boldsymbol{Y}_{i_2})\bigr)}$$

where K is a mixture of Gaussian kernels of different bandwidths. After experimentation, we chose

$$K(\boldsymbol{u},\boldsymbol{y})=\sum_{i=1}^{6}e^{-\frac{\|\boldsymbol{u}-\boldsymbol{y}\|_2^2}{2\sigma_i^2}}\qquad \boldsymbol{\sigma}=(0.001,0.01,0.15,0.25,0.50,0.75).$$

- Intuitively, MMD takes into account all pairs of observations between $\boldsymbol{U}_{i_1}$ and $\boldsymbol{Y}_{i_2}$ (desirable, but costly $\Rightarrow$ mini-batch optimization).
- For the population version, one can show that $\mathrm{MMD}(\boldsymbol{U},\boldsymbol{Y})=0$ if and only if $\boldsymbol{U}\overset{\mathrm{d}}{=}\boldsymbol{Y}$.

3.3 How are GMMNs trained?

- **Training algorithm** (*mini-batch optimization*):
  1) Initialize $\boldsymbol{\theta}$ (weights W_1, W_2: uniform entries; biases $\boldsymbol{b}_1, \boldsymbol{b}_2$: $\mathbf{0}$)
  2) Partition the (n_{trn}, d)-data matrix U and prior (n_{trn}, d)-matrix Z into *batches* ($\approx n_{\text{trn}}/n_{\text{bat}}$ blocks of n_{bat} (batch size) rows each).
  3) For each batch, update $\boldsymbol{\theta}$ by a stochastic gradient step (Adam).
  4) After $\approx n_{\text{trn}}/n_{\text{bat}}$ gradient steps, U and Z are exhausted and one *epoch* is completed. Shuffle their rows and go to Step 2).
  5) Training finishes after n_{epo} epochs.
- **Setup** in our experiments:
  - $p = d$ (inspired from $\mathcal{R}^{-1}$ in the CDM);
  - single hidden layer (*universal approximation theorem*: given suitable activation functions, single hidden layer NNs with $d_1 < \infty$ can approximate any continuous function on a compact subset of $\mathbb{R}^d$);
  - $d_1 = 300$ neurons in the hidden layer;

- ϕ_1 to be ReLU (fast) and ϕ_2 to be sigmoid (maps to $(0,1)$);
- batch size $n_{\text{bat}} = 5000$ (trade-off: large enough for "bottleneck" error to be small; small enough to not be memory-prohibitive);
- number of epochs $n_{\text{epo}} = 300$; and
- $n_{\text{trn}} = 60\,000$ pseudo-samples from C (available for all models).

3.4 How can GMMNs generate QRNs from C?

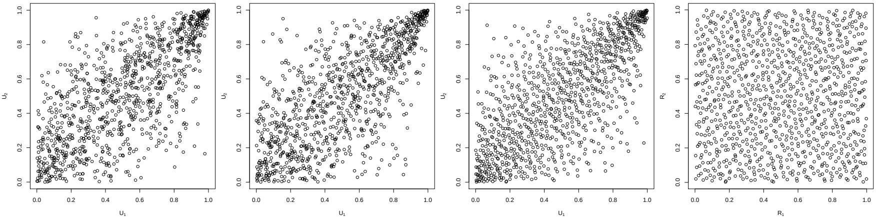
Algorithm 3.1 (GMMN quasi-random sampling)

1) Compute the RQMC points $\boldsymbol{V}_1, \dots, \boldsymbol{V}_{n_{\text{gen}}}$ (e.g., randomized Sobol').
2) Compute the prior samples $\boldsymbol{Z}_i = F_{\boldsymbol{Z}}^{-1}(\boldsymbol{V}_i)$, $i = 1, \dots, n_{\text{gen}}$.
3) Return the pseudo-observations of $\boldsymbol{Y}_i = f_{\hat{\boldsymbol{\theta}}}(\boldsymbol{Z}_i)$, $i = 1, \dots, n_{\text{gen}}$.

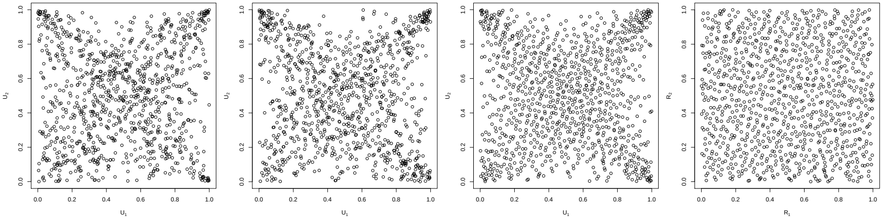
GMMNs are fast to evaluate and sufficiently smooth to preserve low discrepancy. If all the mixed partial derivatives of $h = \Psi \circ f_{\hat{\boldsymbol{\theta}}} \circ F_{\boldsymbol{Z}}^{-1}$ exist a.e. and are continuous, then under Owen-type scrambling, $\mathrm{var}\Big(\frac{1}{n_{\text{gen}}} \sum_{i=1}^{n_{\text{gen}}} h(\boldsymbol{V}_i)\Big) = O(n_{\text{gen}}^{-3}(\log n_{\text{gen}})^{p-1})$.

3.5 Do GMMNs generate samples from C well?

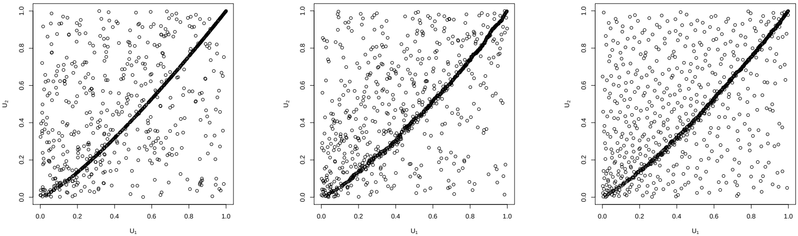
Gumbel PRNs, GMMN PRNs, GMMN QRNs, $\mathcal{R}$-transformed GMMN QRNs:



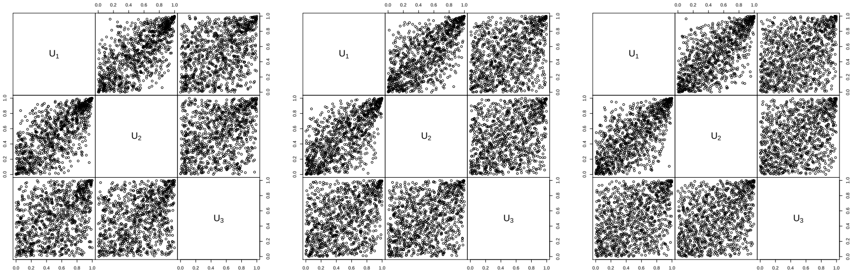
Gumbel-rotated t_4 mixture:



Marshall–Olkin copula $C(u_1, u_2) = \min\{u_1^{1-\alpha_1} u_2, u_1 u_2^{1-\alpha_2}\}$ (without $\mathcal{R}$):



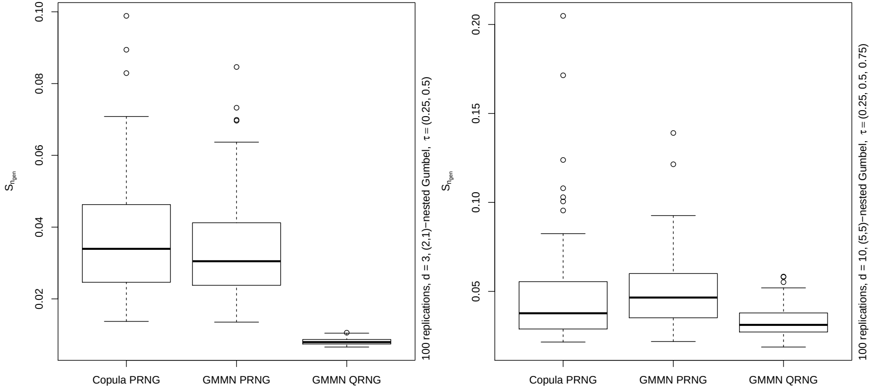
Nested Gumbel copula $C(\boldsymbol{u}) = C_0(C_1(u_1, u_2), u_3)$ ($\tau \in \{0.25, 0.5\}$):



One can consider box plots of realization of the *Cramér–von Mises statistic*

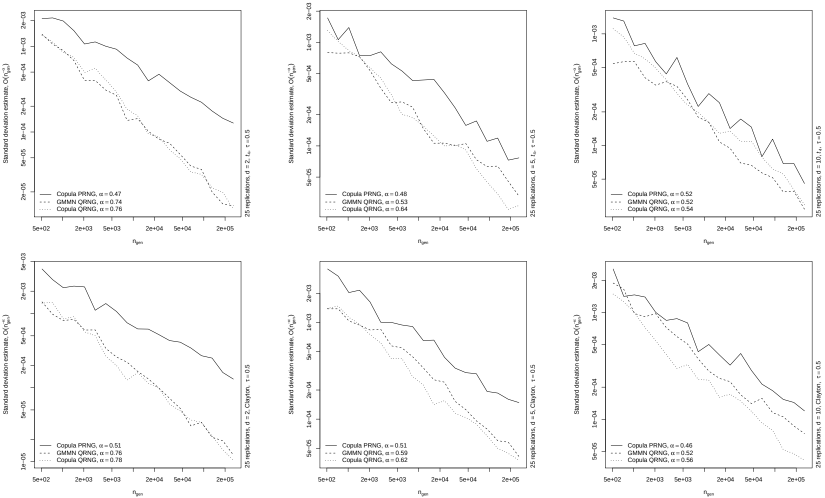
$$S_{n_{\text{gen}}} = \int_{[0,1]^d} n_{\text{gen}}(C_{n_{\text{gen}}}(\boldsymbol{u}) - C(\boldsymbol{u}))^2 \, \mathrm{d}C_{n_{\text{gen}}}(\boldsymbol{u}),$$

where $C_{n_{\text{gen}}}$ is the *empirical copula* (ecdf of the pseudo-obs.) of $n_{\text{gen}} = 1000$ PRNs, GMMN PRNs and GMMN QRNs ($d \in \{3, 10\}$):

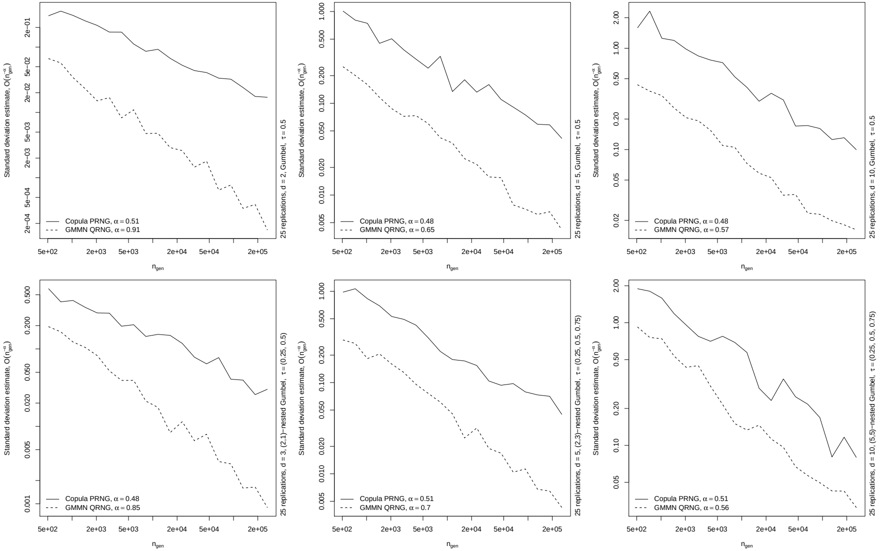


3.6 Variance-reduction effect?

Sample standard deviation of the estimator of $\mathbb{P}(\boldsymbol{U} > \mathbf{0.99})$ computed from PRNs, GMMN QRNs and QRNs:



Sample standard deviation of the estimator of $\mathbb{E}(S \,|\, S > F_S^{-1}(0.99))$ $(\mathrm{N}(0,1)$ margins) computed from PRNs and GMMN QRNs (no QRNs):



Summary

- We can learn a QRNG from any joint model based on a PRNG for C.
- If generated from a known C, this is easy. If only a sample is available, n_{trn} needs to be sufficiently large (depending on the application).
- **Gain:** Universality (all models), computability (robustness, run time), especially useful for real data (where the true model is unknown).
- **Challenges:**
  1) Kendall's tau near 1;
  2) training needs a GPU server (evaluation "needs" TensorFlow);
  3) joint tail behavior;
  4) $d \gg 10$.
- **Open problem:** For $d \gg 10$, must training be improved (as distributions become harder to learn) or do RQMC point sets generally deteriorate? (GMMNs are still smooth). One could try (t, m, s)-nets other than Sobol' (but no related R package or standalone C code available).

Outlook: Application to time series

- **Copula–GARCH model:** $X_{t,j} = \mu_{t,j} + \sigma_{t,j} Z_{t,j}$, $j = 1, \dots, d$, and $\boldsymbol{Z}_t = (Z_{t,1}, \dots, Z_{t,d}) \overset{\text{ind.}}{\sim} H$ with copula C.
- **GMMN–GARCH model:** $X_{t,j} = \mu_{t,j} + \sigma_{t,j} Z_{t,j}$, $j = 1, \dots, d$, and $\boldsymbol{Z}_t = (Z_{t,1}, \dots, Z_{t,d}) \overset{\text{ind.}}{\sim} H$ with GMMN for C.
- **Goal:** Improving empirical predictive distributions (here: 1-step ahead).
- **Applications:**
  1) **ZCB yield curves** (term structure of interest rates)
     - US ($d = 30$ annual times to maturity) and CA ($d = 120$ quarterly times to maturity) ZCB yield curves (training: 1995–2015)
     - Marginal deARMA–GARCHing ⇒ standardized residuals $(\hat{\boldsymbol{Z}}_t)_t$
     - PCA ⇒ $d = 3$ (US data) and $d = 4$ (CA data) account for $\geq 95\%$ of the total variance
     - **Performance** evaluation on 1y (CA) and 1/2y (US) test data:

- Dependence: $\mathrm{AMMD} = \frac{1}{100}\sum_{i=1}^{100} \mathrm{MMD}(U^{(i)}, \hat{U}; K_{\mathsf{tst}})$, where $U^{(i)}$ is the ith matrix of generated data and $\hat{U}$ are the pseudo-observations of the test data.
- Better one-day ahead empirical predictive distributions:

$$\mathrm{AVS}^{1/4} = \frac{1}{T-\tau}\sum_{t=\tau+1}^{T}\sum_{j_1=1}^{d}\sum_{j_2=1}^{d}\left(|X_{t,j_1} - X_{t,j_2}|^{1/4} - \frac{1}{1000}\sum_{i=1}^{1000}|\hat{X}_{t,j_1}^{(i)} - \hat{X}_{t,j_2}^{(i)}|^{1/4}\right)^2$$

For both datasets, the GMMN–GARCH models better matched the cross-sectional dependence on the test datasets.

2) **Exchange rates w.r.t. USD and w.r.t. GBP**

- CAD, GBP, EUR, CHF, JPY w.r.t. USD and of CAD, USD, EUR, CHF, JPY, CNY w.r.t. GBP (training: 2000–2015; test: 2015)
- AMMD and $\mathrm{AVS}^{1/4}$ as before.
- $\mathrm{VEAR}_\alpha = \left|\alpha - \frac{1}{T-\tau}\sum_{t=\tau+1}^{T} \mathbb{1}_{\{S_t < \widehat{\mathsf{VaR}}_\alpha(\hat{S}_t)\}}\right|$ (VaR ex. abs. err.)
- For both datasets, the GMMN–GARCH models produced better daily $\mathrm{VaR}_{0.05}(S_t)$ forecasts.

References

Cambou, M., Hofert, M., and Lemieux, C. (2017), Quasi-random numbers for copula models, *Statistics and Computing*, 27(5), 1307–1329, doi:10.1007/s11222-016-9688-4.

Hofert, M., Kojadinovic, I., Mächler, M., and Yan, J. (2018), Elements of Copula Modeling with R, Springer Use R! Series, ISBN 978-3-319-89635-9, doi:10.1007/978-3-319-89635-9, http://www.springer.com/de/book/9783319896342 (2018-03-15).

Hofert, M., Prasad, A., and Zhu, M. (2020), Quasi-random sampling for multivariate distributions via generative neural networks, *Journal of Computational and Graphical Statistics*, doi:10.1080/10618600.2020.1868302.