Not a matter of luck –
The smart choice of Monte Carlo samples

November, 19th 2021

Point of departure: A small(?!) problem

- Project: Support a client in extending its internal model for a new risk category
  - Complex model: high dimensional due to lots of risk factors, large losses, significant tail dependencies
- After implementation and validation the only task remaining was integration into the separate overall model of the client's group
- The problem: The group's model was only able to process 10'000 scenarios … our model required 1'000'000
  - The precision of Monte Carlo simulations is proportional to the size of the sample.
  - A factor 100 in size directly translates into a factor 100 in the variance of results
- What to do?

Solution: Find a “good” sample

- Wanted is a “good” small sample (with $N = 10'000$) which reflects the large sample (with $N = 1'000'000$) in the best possible way.
  - Two questions: 1) What is the meaning of “good” exactly? 2) How to find such a “good” sample?

1. A small sample is “good” if it reproduces the important *features* of a large sample with small deviation
   - Features are functions of the samples, they can be statistical such as means or correlations or business related such as the probability of a large loss in a legal entity.
   - The feature is calculated for the large and small sample, the squared difference is the deviation respectively the error.
2. One simple way of finding such a small yet good sample is by Trial & Error
   - Choose randomly a small sample from the large one
   - Calculate the features of this small sample and determine the error
   - Repeat this for a number of trials, say 1’000 times
   - Retain that sample from all trials with the smallest error

Can you go further?

- With this approach the original problem could be solved
- But this application is somewhat atypical since all interesting outputs of the model were already known.
  - Normally a Monte Carlo is done to determine values which are NOT known in the first place
- Question: Does this approach for sample selection has benefits even for the calculation of unknown targets of interest?
  - Features have to be easy to calculate to be usable for the optimisation procedure
  - If a target function is very hard to calculate or - worse - not even described explicitly it cannot serve as feature
  - Question: Does controlling for features reduces error also for difficult/unknown targets?
  - Question: How to rule out that selecting for features actually increases the error of the target functions?

The Hull-White model

- Suggestion: Just try it out in a laboratory model!
- Hull-White: Stochastic interest rate model
  - Risk factors x consist of annual short rates $r(1), \dots, r(T)$ and integrated short rates $Y(1), \dots, Y(T)$ with $T = 30$
  - $x = (r, Y)$ has a multivariate normal distribution
  - Vector of means and covariance matrix are results of calibration (according to DAV) and known
  - Field of application: Determination of prices and market consistent values of interest rate sensitive instruments
- The price respectively the market consistent values of a cash flow is the expectation of the cash flow f under the distribution of the Hull-White model

$$\text{Price} = \mathrm{E}_{HW \sim X}[f(X)] = \int_{\mathbb{R}^{60}} f(x)\, dHW(x)$$

Integrals and designs

- Even in simple models there is no explicit solution to most integrals.
- Often the explicit form of the target function itself is not known.
- Solution: Approximation of the continuous measure by empirical measures

Points:	Measures:	Integrals:
All $x \in \mathbb{R}^{60}$	Hull-White measure: $dHW(x)$	$\int_{\mathbb{R}^{60}} f(x)\, dHW(x)$
⇓	⇓	⇓
Design: $x_1, \dots, x_n$	empirical measure: $\frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$	$\frac{1}{n}\sum_{i=1}^{n} f(x_i)$

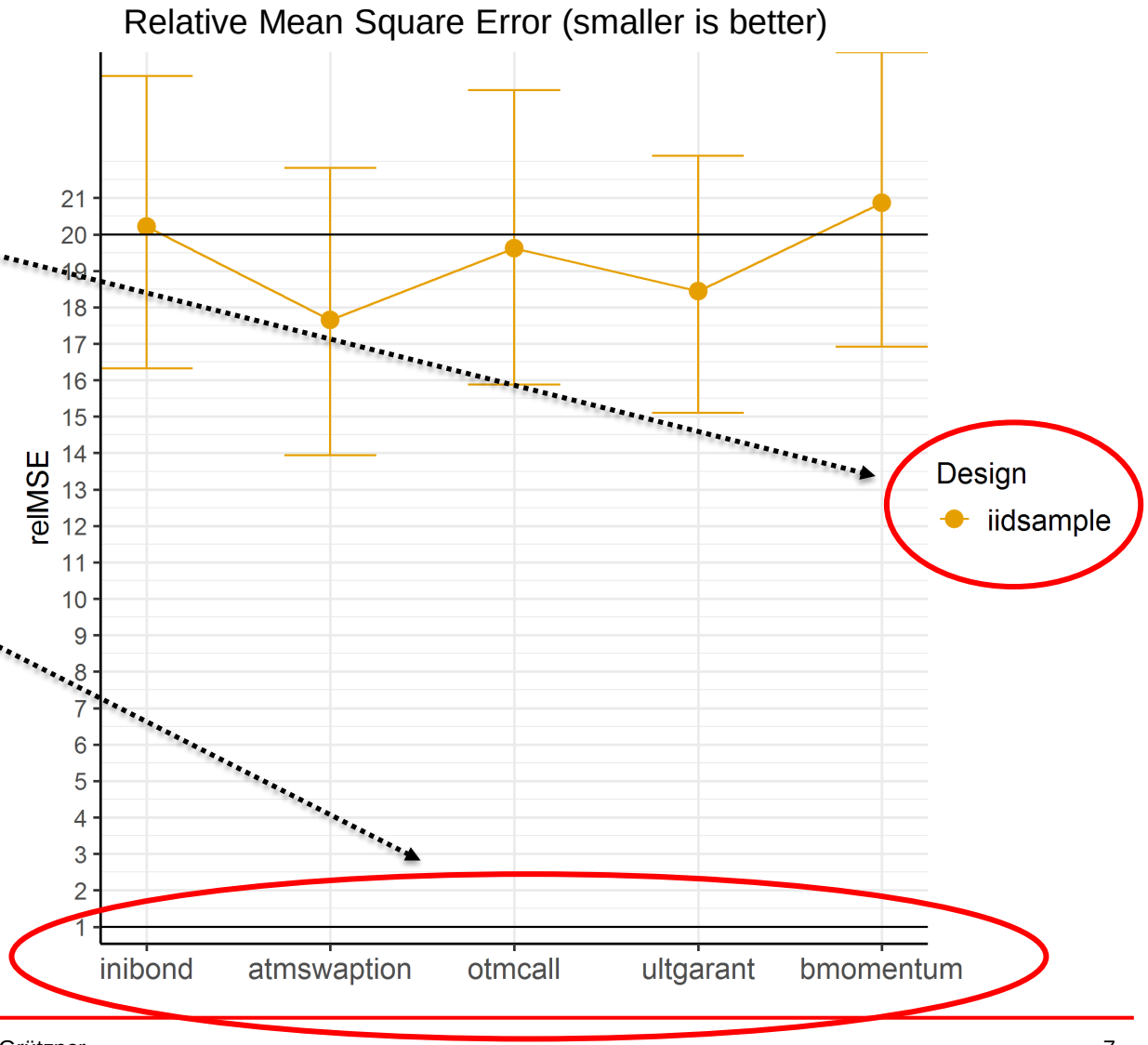
- A *Design* is a set of points/scenarios for integration.

The first experiment as warm-up

- Design:
  - iidsample Monte Carlo Sample with n=250

Target function:

- inibond: 30-year zerobond
- atmswaption: Option on an at-the-money swap
- otmcall: Out-of-the-money call on bond (with 90% chance of zero expiry)
- ultgarant: Guarantee (0%) on the ultimate value of the bank account after 30 years.
- bmomentum: Self-financing bond trading strategy: Invest each year into the bond with larger return in the prior year.



The first experiment as warm-up

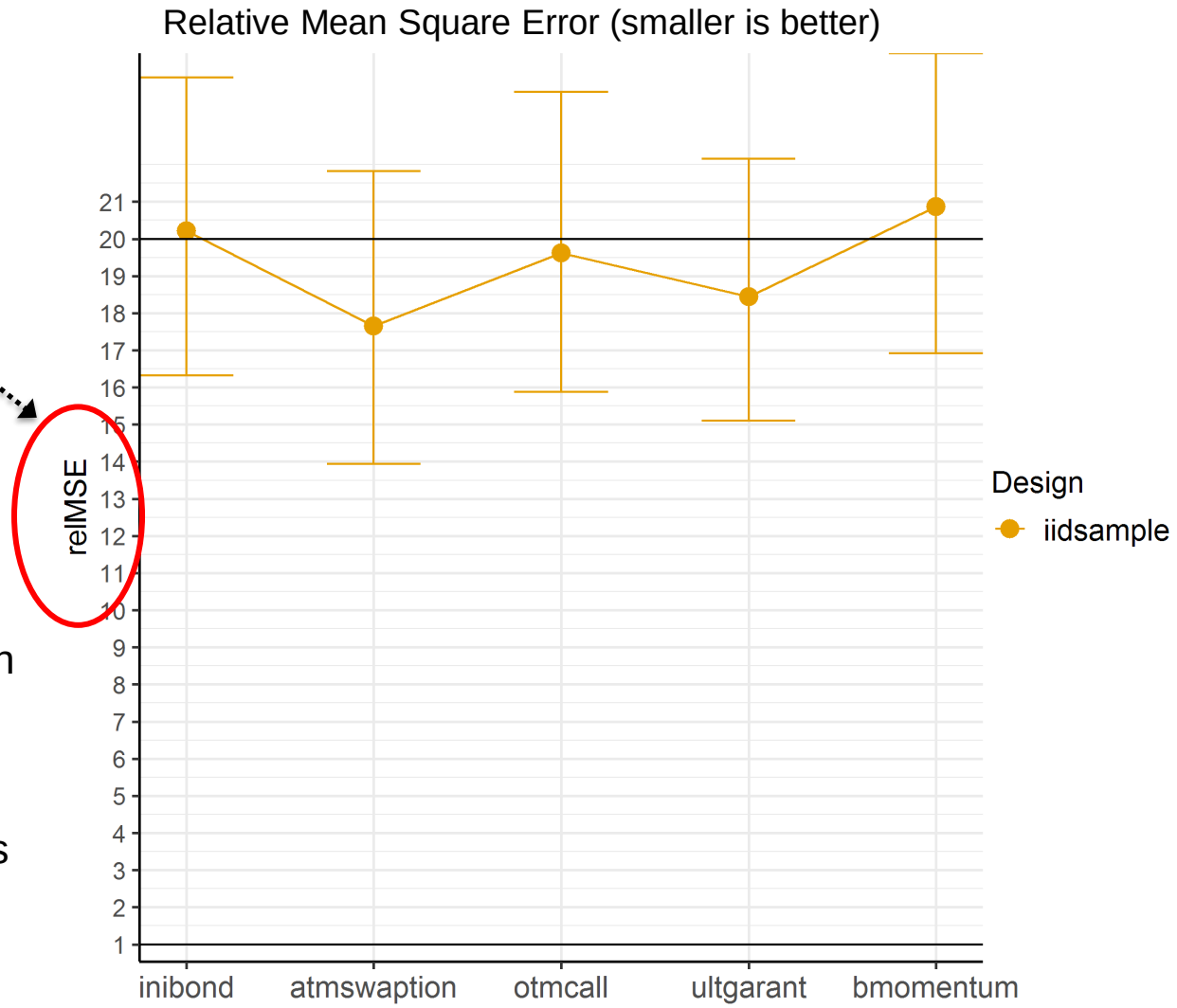
Indicator for the size of the error

- relMSE: relative Mean Square Error
- Square Error:

$$SE = \left(\text{Exact} - \frac{1}{250} \sum_{i=1}^{250} f(x_i) \right)^2$$

Exact: Analytic value or very large Monte-Carlo estimate

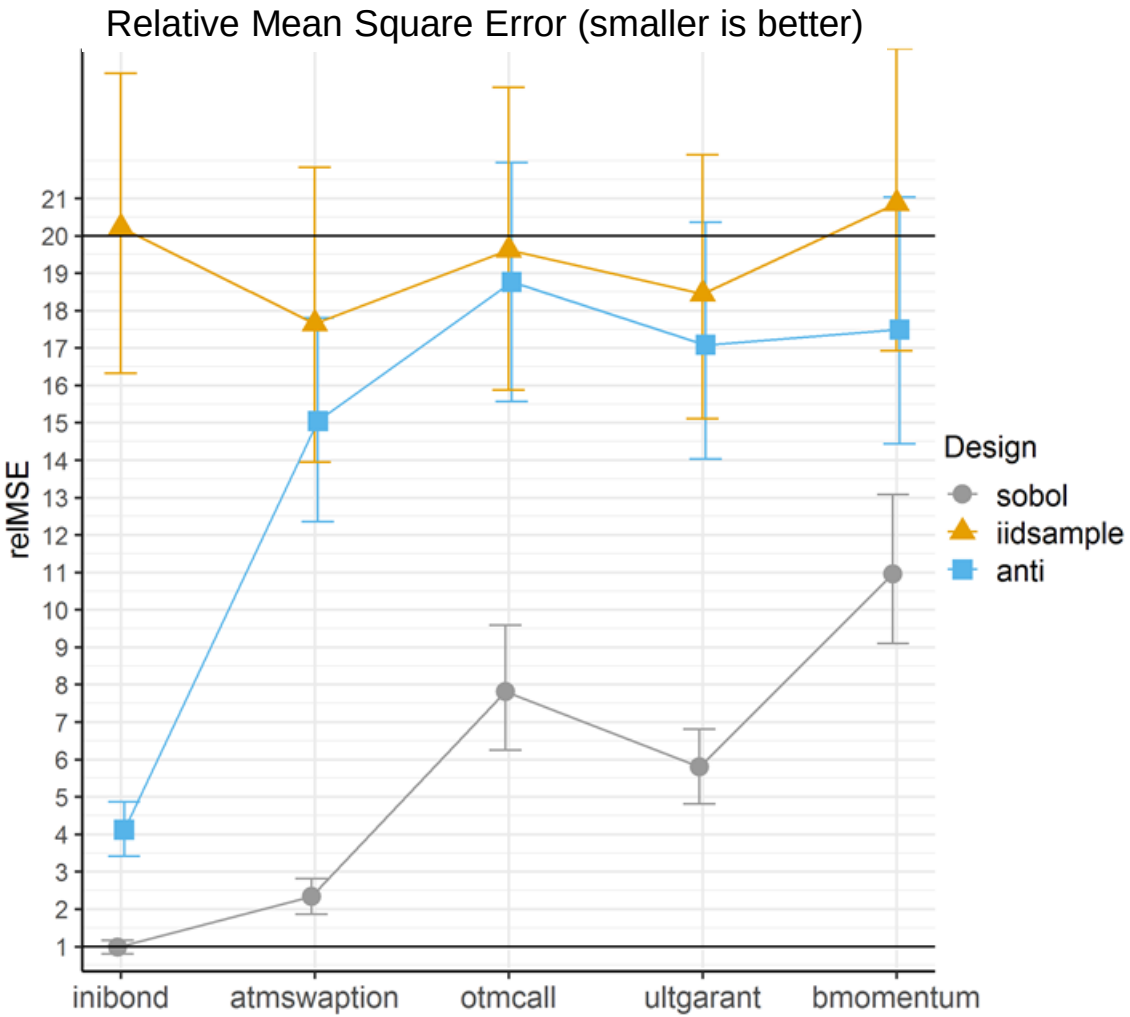
- Mean: averaged over a number of runs
- relative: Mean Square Error relative to a large sample with N=5'000.
- relMSE=20 means:
  On average the squared error of integration is 20 times as large as the error of an iid-Monte Carlo sample with N=5'000.



The first experiment

Comparison of three designs

- iidsample:
- Plain vanilla Monte-Carlo sample
- relMSE for all targets roughly equal to the ratio of the size of the samples, i.e. 5'000 / 250 = 20.
- anti:
- Antithetic sample
- Some improvement for inibond but in all other cases just like iidsample
- sobol:
- Sobol[1)] Quasi Monte Carlo, randomised.
- Significant improvement of error.
- Efficiency depends on target.



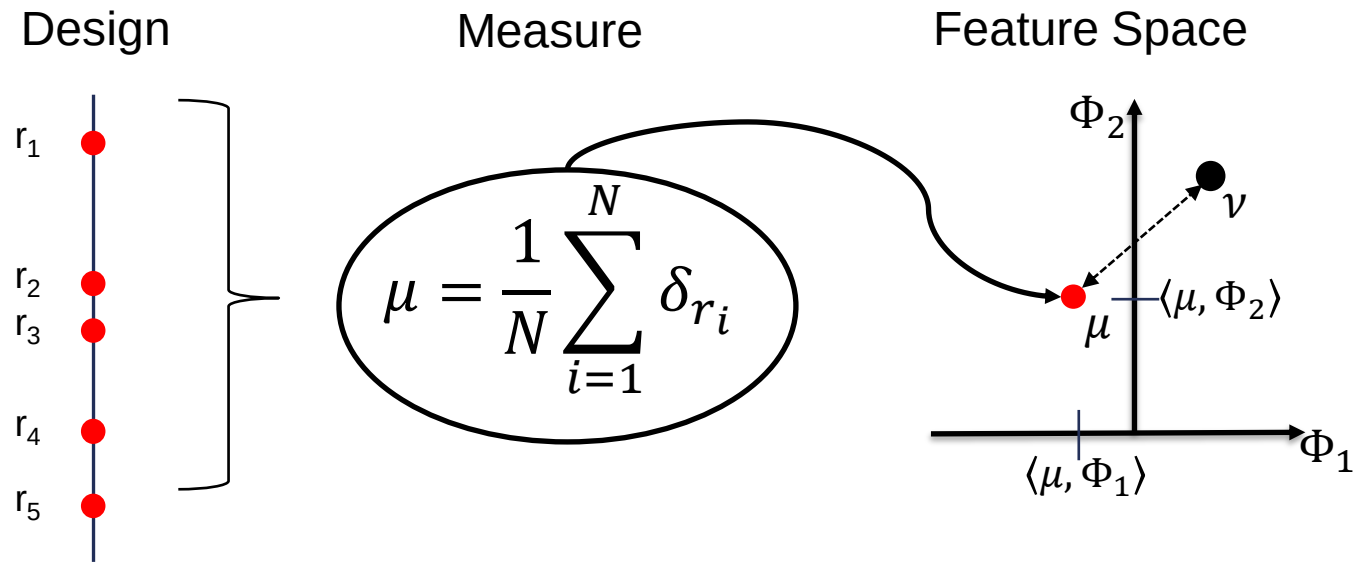
1) Function `sobol` in R-library «`randtoolbox`»

Introducing features

- First example: Two simple features
  - Focus on a single risk factor r (e.g. short rate in year 15)
  - Definition of the two Features: $\Phi_1(\mathrm{r}) = \mathrm{r}$ und $\Phi_2(\mathrm{r}) = \mathrm{r}^2$
- Every design $r_1, \ldots, r_N$ can be mapped to the values of its features
  - A design acts by integrating with its empirical measure $\mu = \frac{1}{N}\sum \delta_{r_i}$
  - $\langle \mu, \Phi_1 \rangle = \frac{1}{N}\sum r_i$ is the empirical mean of the design
  - $\langle \mu, \Phi_2 \rangle = \frac{1}{N}\sum r_i^2$ is the second empirical moment of the design
- Per values of their features any two designs can be compared
  - e.g. «large» design μ with «small» design ν having points $r_1', \ldots, r_n'$.
  - The *discrepancy* (or error) between μ und ν is the square root of

$$\left(\langle \mu, \Phi_1 \rangle - \langle \nu, \Phi_1 \rangle \right)^2 + \left(\langle \mu, \Phi_2 \rangle - \langle \nu, \Phi_2 \rangle \right)^2$$

The feature space



- The *Feature Space* is spanned by the features, it is a linear space of functions
  - The integrals of features are the coordinates in feature space
  - The mapping of a measure to a point in feature space is called the *Kernel Mean Embedding*
- The *discrepancy* between two measures or two designs is their distance in feature space

The Kernel

- The *Kernel* of a feature space is the inner product of the point masses respectively the features
- In the example:

$$K(r, r') = \langle \delta_r , \delta_{r'} \rangle = \Phi_1(r) \cdot \Phi_1(r') + \Phi_2(r) \cdot \Phi_2(r')$$
$$= r \cdot r' + r^2 \cdot r'^2$$

- The kernel encodes all properties of the feature space
- Spaces of function which allow a kernel are called *Reproducing Kernel Hilbert Spaces*.

How many features are possible

- The first example provided two features one for the first and one for the second moment

$$\Phi_1(r) = r \qquad \Phi_2(r) = r^2$$

- But why shouldn't we control ALL moments?
- Kernel for two moments: $K(r, r') = r \cdot r' + r^2 \cdot r'^2$
- Kernel for ALL moments:

$$K(r, r') = 1 + r \cdot r' + \frac{1}{2!} r^2 \cdot r'^2 + \frac{1}{3!} r^3 \cdot r'^3 + \cdots$$

$$= \sum_{m=0}^{\infty} \frac{(r \cdot r')^m}{m!} = \exp(r \cdot r')$$

- We just need weights such as $\frac{1}{m!}$ to ensure convergence.

The Kernel-Trick

- "Infinitely many features" means the feature space is infinite dimensional
  - But actually this does not matter for practical calculations!
- The discrepancy between designs $r_1, \dots, r_N$ (μ) und $r_1', \dots, r_n'(\nu)$ with respect to all moments can be obtained simply by sums over expressions involving the kernel

$$\|\mu - \nu\|^2 = \langle \mu - \nu, \mu - \nu \rangle = \langle \mu, \mu \rangle - 2\langle \mu, \nu \rangle + \langle \nu, \nu \rangle$$

$$\langle \mu, \mu \rangle = \left\langle \frac{1}{N}\sum_{i=1}^{N} \delta_{r_i}, \frac{1}{N}\sum_{j=1}^{N} \delta_{r_j} \right\rangle = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \left\langle \delta_{r_i}, \delta_{r_j} \right\rangle = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \exp\left(r_i\, r_j\right)$$

$$\langle \mu, \nu \rangle = \frac{1}{Nn}\sum_{i=1}^{N}\sum_{j=1}^{n} \exp\left(r_i\, r_j'\right) \qquad \langle \nu, \nu \rangle = \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} \exp\left(r_i'\, r_j'\right)$$

- Conclusion: The discrepancy between designs can be calculated easily and efficiently even for infinite dimensional feature spaces!
  - See code example in https://github.com/QuantAkt/minimal-working-example

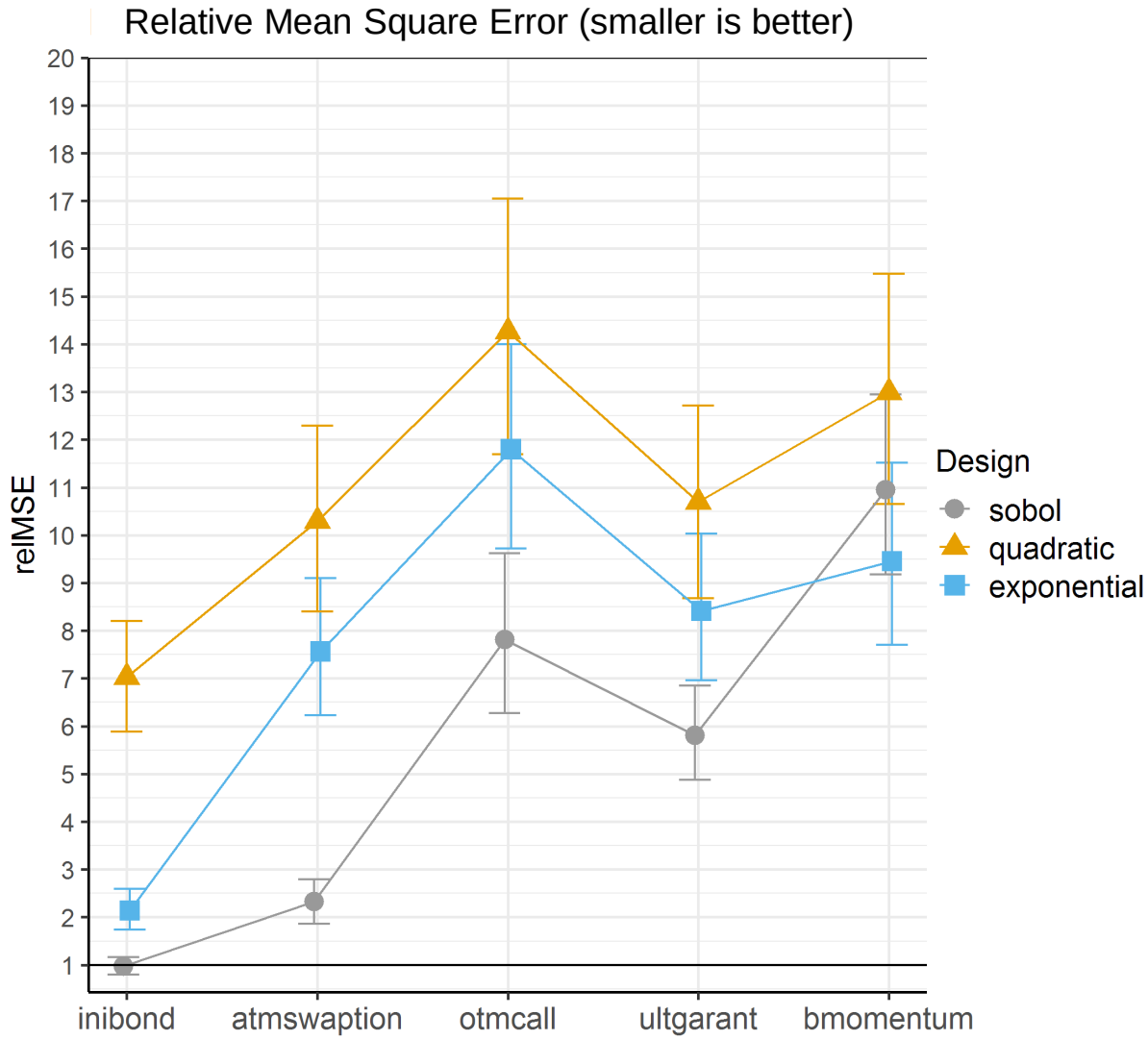
Moments as features

- quadratic:
- The kernel is $K(x, y) = (1 + x^T y)^2$
- Contains pairwise interactions between risk factors.

- exponential:
- The kernel is $K(x, y) = \exp(x^T y)$

Both kernels are defined on all risk factors (i.e. in the example 60).

Selection of designs is again by «Trial&Error».



MMD: Maximum Mean Discrepancy

- The distance in the feature space $\mathcal{F}$ is the Worst Case Integration Error!

$$\|\mu - \nu\| = \max_{\substack{f \in \mathcal{F} \\ \|f\| \leq 1}} \int f \, d\mu - \int f \, d\nu$$

- This explains why this distance is often called *Maximum Mean Discrepancy (MMD).*
- Because the Worst Case bounds the integration error for ALL functions in feature space, MMD permits a clear answer to the question “Which functions can be controlled by a given kernel/feature?”

For all $f \in \mathcal{F}$: $\left| \int f \, d\mu - \int f \, d\nu \right| \leq \|f\| \cdot MMD(\mu, \nu)$

Aggressive optimisation and kernel engineering

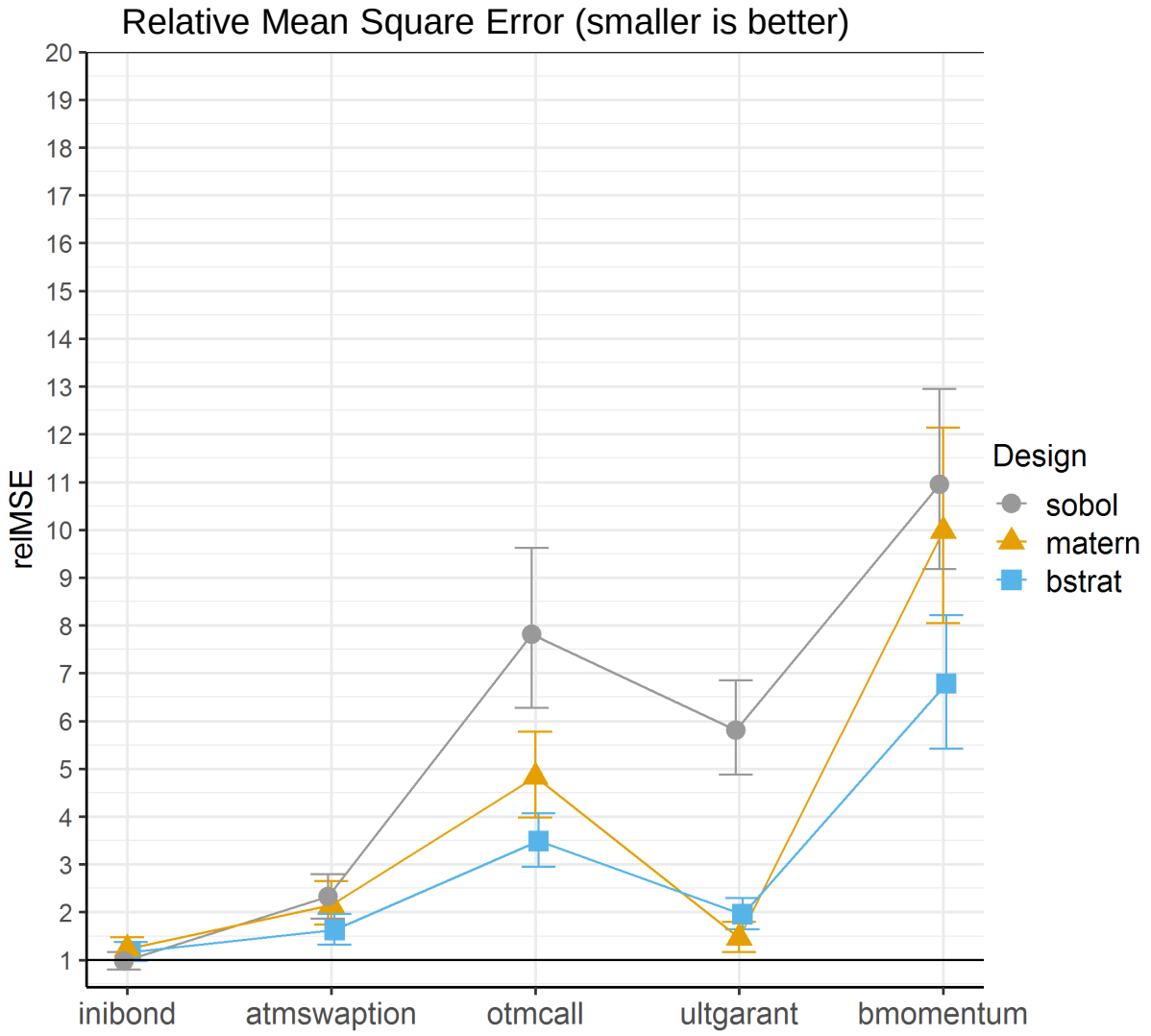
- Kernel-Engineering: Choice of kernel motivated by the feature space.
- In addition: «Greedy» selection of designs.

- matern:

- Well known standard kernel[1)] can be used out-of-the-box
- Feature space is quite large and contains continuous, bounded functions.

- bstrat:

- Feature space spanned by bond trading strategies.



1) e.g. found in sklearn.gaussian_process.kernels.Matern

Conclusions

- The selection by features respectively using kernels can improve the quality of samples/designs significantly!
- The method is completely generic, it works for all distributions and models.
- Since the selection starts simply with a large sample, no modification of the stochastic models or economic scenario generators is required.
- Prior domain knowledge about the targets can be included by engineering the kernels appropriately.
- Error estimates allow statements on convergence and put the approach on sound mathematical grounds.
- There are synergies in methods and techniques with other kernel based machine learning approaches such as Gaussian Process Regression or Bayesian Optimisation.

Literature

This is just a first overview. I am happy to provide more sources on request.

- Wikipedia
  - Kernel Method https://en.wikipedia.org/wiki/Kernel_method
  - Kernel Mean Embedding https://en.wikipedia.org/wiki/Kernel_embedding_of_distributions
- Overview article on Kernel Mean Embeddings
  - Kernel Mean Embedding of Distributions: A Review and Beyond https://arxiv.org/abs/1605.09522
- Scientific papers
  - «Super-Samples from Kernel Herding» von Chen, Yutian and Welling, Max and Smola, Alex in Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence oder als https://arxiv.org/abs/1203.3472
  - “Construction of Optimal Cubature Algorithms with Applications to Econometrics and Uncertainty Quantification” von J.Oettershagen, PhD thesis, University of Bonn, 2017. https://ins.uni-bonn.de/media/public/publication-media/diss_oettershagen.pdf
- Code (minimal working example)
  - https://github.com/QuantAkt/minimal-working-example

Contact

guido.gruetzner@quantakt.com

https://www.linkedin.com/in/guido-gruetzner

- Do not hesitate to contact me with any questions you might have.
- When you try this out, make sure to share your experience. I would be glad to hear from you.