Efficient, consistent and flexible Credit Default simulation with TRNG and RcppParallel

Riccardo Porreca

Roland Schmid

Mirai Solutions GmbH
Tödistrasse 48
CH-8002 Zurich
Switzerland

info@mirai-solutions.com
www.mirai-solutions.com

Outline

- Motivation and challenges
- Integrated Market and Default Risk model
- Parallel Random Number Generation techniques with TRNG
- Examples and results
- Summary

Motivation and challenges

- Monte Carlo approach to the simulation of rare and correlated credit default events in large portfolios
- Desire for efficient, consistent, flexible MC simulation
  - parallel execution on multicore architectures
  - simulation of a subset of the variables of interest (e.g. sub-portfolios)
  - granular details/insight for given scenarios of interest (e.g. individual contributions)
  - exact reproducibility of full-simulation results (*fair-playing*)
    => control and isolate Monte Carlo effect
- General flexibility of performing fast ad-hoc consistent simulations
- Limitation: (Pseudo)Random Number Generators (RNGs) used to draw random numbers in Monte Carlo simulations are intrinsically sequential

Integrated market and default risk model

- Portfolio of defaultable securities issued by a set of counterparties
- Model market risk in correlation with credit default risk using an integrated approach
  - Market and default risk are intrinsically related
  - Dependency must be properly taken into account
- Simplifying assumptions
  - Default occurrence determined at counterparty level
  - Exactly one security per each counterparty in the portfolio
  - We ignore non-defaultable securities subject to market risk only

Integrated market and default risk model

- State of the credit environment driving the default of counterparty j in [1,J]

$$Y_j = \underbrace{\beta_j Z_j}_{\substack{\text{systemic component} \\ \text{(reflects the state of the world)}}} + \underbrace{\sigma_j \varepsilon_j}_{\substack{\text{specific component} \\ \text{(idiosyncratic return)}}}, \text{ i.i.d. } \varepsilon_j \sim N(0,1)$$

- Return r_j drives the market value at horizon (based on the state of the world)

$$V_j = V_j^0(1 + r_j)$$

- Default indicator

$$D_j = \begin{cases} 1, & Y_j < \theta_j \\ 0, & Y_j \geq \theta_j \end{cases}, \quad \theta_j : \mathrm{P}(Y_j < \theta_j) = P_j^D$$

- Loss (including occurrence of defaults)

$$L_j = V_j^0 - \left[(1 - D_j)V_j + D_j R_j\right], \quad 0 \leq R_j \leq V_j^0$$

- Default events D_j and losses L_j inherit the correlation structure of r_j and Z_j with other counterparties

Integrated market and default risk model

- Assumption: M scenarios of the state of the world are available

$$\left\{Z_j^{(m)}, r_j^{(m)}\right\}_{m=1}^{M}$$

  - Monte Carlo realizations from a given market risk model, which we extend by the occurrence of defaults
  - we also assume (WLOG): $Z_j \sim N(0,1)$, $\sigma_j = \sqrt{1-\beta_j^2} \Rightarrow \theta_j = \Phi^{-1}\left(P_j^D\right)$

$$Y_j = \beta_j Z_j + \sqrt{1-\beta_j^2}\,\varepsilon_j$$

$$V_j = V_j^0(1+r_j)$$

$$D_j = \begin{cases} 1, & Y_j < \Phi^{-1}\left(P_j^D\right) \\ 0, & Y_j \geq \Phi^{-1}\left(P_j^D\right) \end{cases}$$

$$L_j = V_j^0 - \left[\left(1-D_j\right)V_j + D_j\,R_j\right]$$

Integrated market and default risk model

- Assumption: M scenarios of the state of the world are available

$$\left\{Z_j^{(m)}, r_j^{(m)}\right\}_{m=1}^{M}$$

  - Monte Carlo realizations from a given market risk model, which we extend by the occurrence of defaults
  - we also assume (WLOG): $Z_j \sim N(0,1),\ \ \sigma_j = \sqrt{1-\beta_j^2} \Rightarrow \theta_j = \Phi^{-1}\left(P_j^D\right)$

- Monte Carlo approach for simulating the integrated model:
  - combine V_j and Z_j for the available scenarios with independent realizations of the idiosyncratic returns ε_j
  - for each scenario m in [1,M], generate K samples of ε_j to obtain M*K realizations of the credit environment return Y_j
  - combined simulation size M*K high enough to capture the rare nature of default events

Efficient, consistent, flexible MC simulation

- Exact reproducibility of full-simulation results
  - parallel execution on multicore architectures
  - simulation of a subset of the variables of interest (e.g. sub-portfolios)
  - granular details/insight for given scenarios of interest (e.g. individual contributions)

- Tina's Random Number Generator Library (TRNG)

“*state of the art C++ pseudo-random number generator library for sequential and parallel Monte Carlo simulations*”

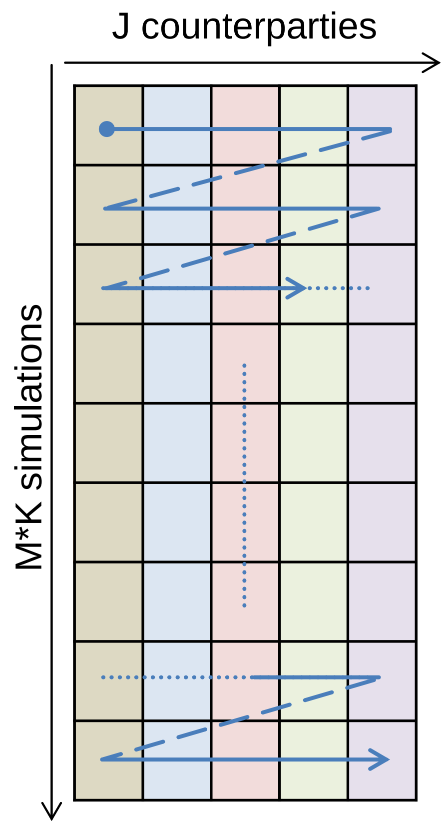
[H. Bauke, http://numbercrunch.de/trng]

Efficient, consistent, flexible MC simulation

- TRNG provides a collection of parallel-oriented random number engines
  - simple structure (LFSR sequences)
  - strong mathematical properties
  - possible to manipulate the internal state
  - equipped with powerful operations
    - jump
    - split

- Available to the R community via rTRNG package
  - being developed at Mirai Solutions

  => `install_github("miraisolutions/rTRNG")`

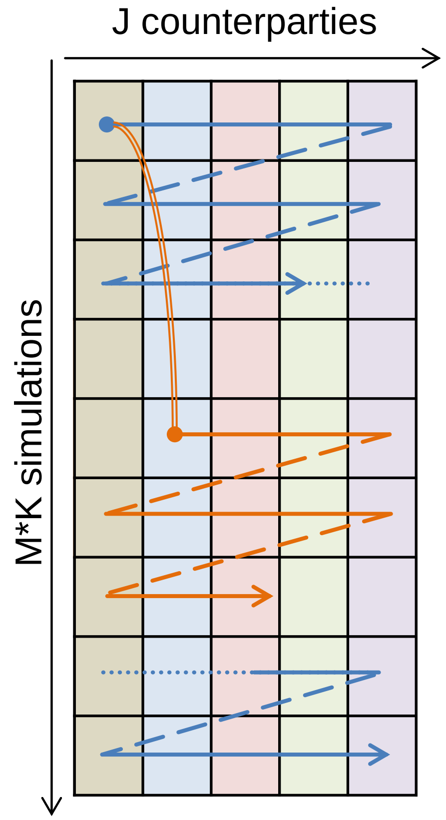


Efficient, consistent, flexible MC simulation

- TRNG provides a collection of parallel-oriented random number engines
  - simple structure (LFSR sequences)
  - strong mathematical properties
  - possible to manipulate the internal state
  - equipped with powerful operations
    - jump
    - split

- Available to the R community via rTRNG package
  - being developed at Mirai Solutions

```
=> install_github("miraisolutions/rTRNG")
```

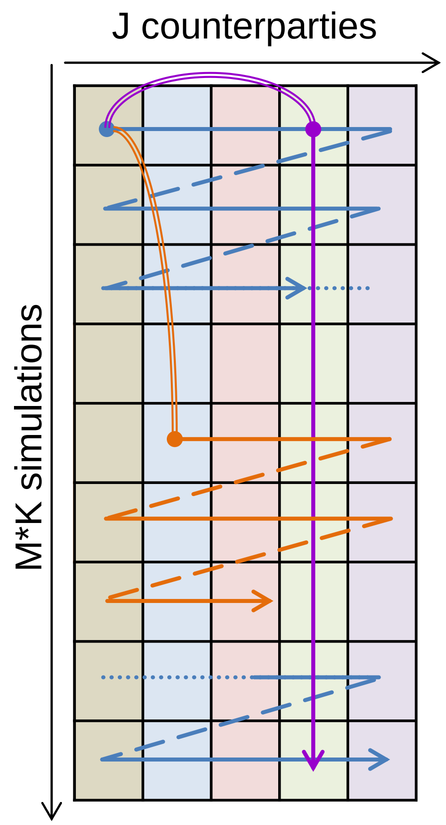


Efficient, consistent, flexible MC simulation

- TRNG provides a collection of parallel-oriented random number engines
  - simple structure (LFSR sequences)
  - strong mathematical properties
  - possible to manipulate the internal state
  - equipped with powerful operations
    - jump
    - split

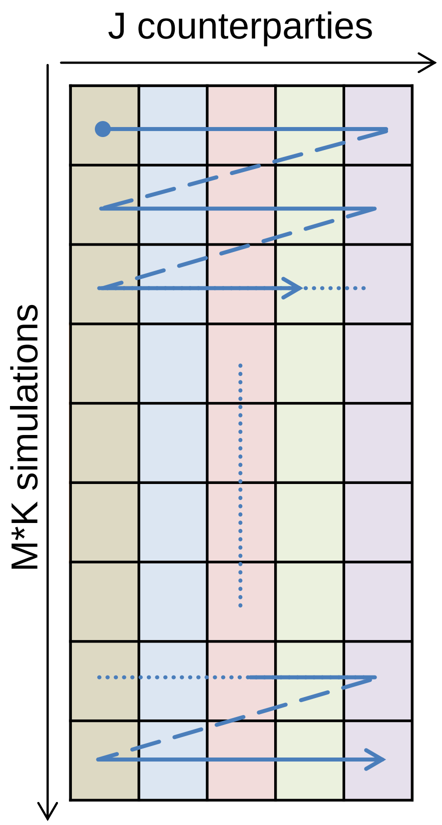
- Available to the R community via rTRNG package
  - being developed at Mirai Solutions

=> install_github("miraisolutions/rTRNG")



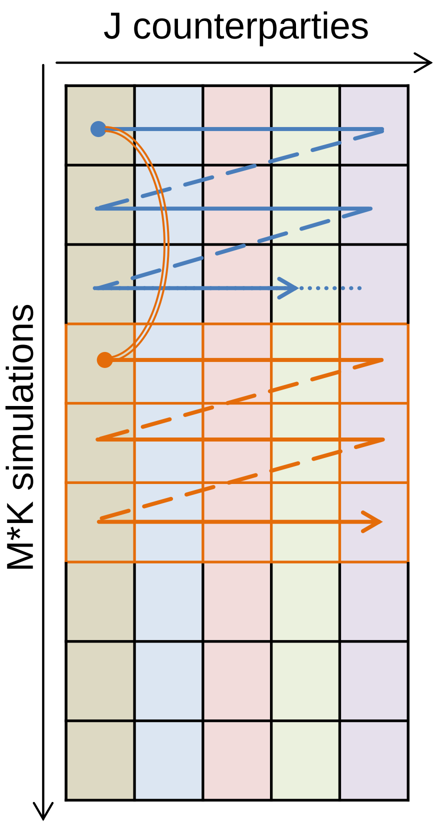
Efficient, consistent, flexible MC simulation

- Parallel execution
  - Jump to the beginning of a given chunk of simulations (*block splitting*)
- Sub-portfolio simulation
  - Split and simulate only the relevant counterparties
- Insight for given scenarios of interest
  - Jump to individual simulations
- Any combination of the above



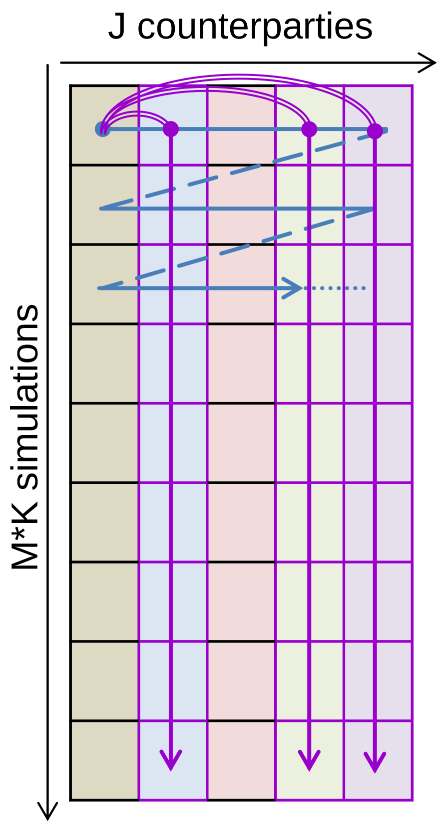
Efficient, consistent, flexible MC simulation

- Parallel execution
  - Jump to the beginning of a given chunk of simulations (*block splitting*)
- Sub-portfolio simulation
  - Split and simulate only the relevant counterparties
- Insight for given scenarios of interest
  - Jump to individual simulations
- Any combination of the above



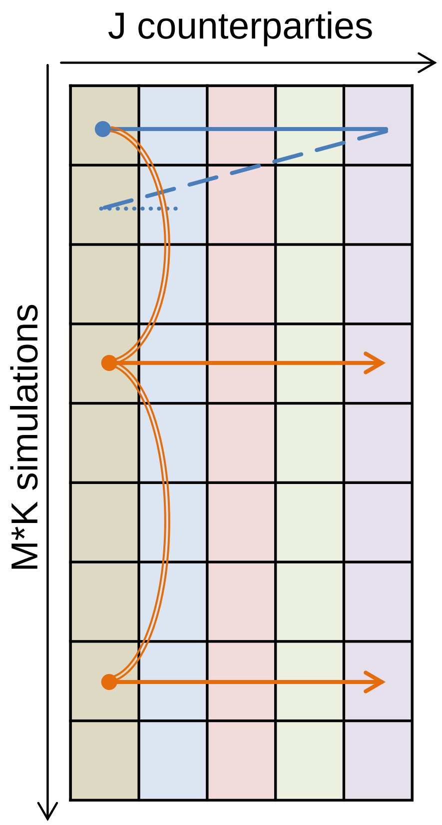
Efficient, consistent, flexible MC simulation

- Parallel execution
  - Jump to the beginning of a given chunk of simulations (*block splitting*)
- Sub-portfolio simulation
  - Split and simulate only the relevant counterparties
- Insight for given scenarios of interest
  - Jump to individual simulations
- Any combination of the above



Efficient, consistent, flexible MC simulation

- Parallel execution
  - Jump to the beginning of a given chunk of simulations (*block splitting*)
- Sub-portfolio simulation
  - Split and simulate only the relevant counterparties
- Insight for given scenarios of interest
  - Jump to individual simulations
- Any combination of the above



R(cpp) simulation kernel

- Efficient, consistent, flexible, parallel simulation kernel
  - Fast C++ simulation core with Rcpp and RcppParallel
    - seamless R and C++ integration
    - in memory, thread-safe access to R objects
  - TRNG C++ headers and library from rTRNG
  - simple yet powerful multi-purpose simulation core

- Assessment on a test portfolio
  - J = 6’000 counterparties
  - M = 10’000 available market scenario simulations
  - K = 100 idiosyncratic simulations for a given market scenario
  - fixed $\beta_j = \sqrt{0.5}$

Data and interface

$$Y_j = \beta_j Z_j + \sqrt{1-\beta_j^2}\,\varepsilon_j$$

$$V_j = V_j^0 (1 + r_j)$$

$$D_j = \begin{cases} 1, & Y_j < \Phi^{-1}\left(P_j^D\right) \\ 0, & Y_j \geq \Phi^{-1}\left(P_j^D\right) \end{cases}$$

$$L_j = V_j^0 - \left[\left(1 - D_j\right)V_j + D_j\, R_j\right]$$

```
simulationKernel(pf, Z, r,
                 J,
                 K, mk = seq_len(K * nrow(Z)),
                 agg = factor(rep("PF", nrow(pf))),
                 seed)
```

Data and interface

j	V0	R	beta	PD	rtng	...
2	32606970	910000	0.707	0.0025	BBB	.
3	8932752	92800	0.707	0.0010	A	.
6	564931	335675	0.707	0.1000	CCC	.
7	3494502	82000	0.707	0.0400	B	.
9	6679886	1000500	0.707	0.0100	BB	.

j

m	1	.	.	.	J
1	0.5241	-0.484	-0.402	-0.774	-0.702
.	-2.2608	-0.666	-1.003	0.423	0.683
.	-0.0197	-0.174	-0.178	-0.607	-0.858
.	0.1831	-1.011	-0.488	0.209	0.368
M	-0.3614	0.740	0.928	-0.777	-1.430

j

m	1	.	.	.	J
1	-0.512	-0.376	-1.059	-0.165	0.511
.	-0.268	-1.135	-1.318	1.760	-3.088
.	-0.199	0.609	-0.204	1.612	0.474
.	0.857	0.566	-0.601	0.803	0.626
M	-0.167	-0.736	0.661	1.265	0.981

```
simulationKernel(pf, Z, r,
                 J,                                  total nr. of counterparties
                 K, mk = seq_len(K * nrow(Z)),
                 agg = factor(rep("PF", nrow(pf))),
                 seed)                               initial RNG state
```

$$Y_j = \beta_j Z_j + \sqrt{1-\beta_j^2}\,\varepsilon_j$$

$$V_j = V_j^0(1+r_j)$$

$$D_j = \begin{cases} 1, & Y_j < \Phi^{-1}\left(P_j^D\right) \\ 0, & Y_j \geq \Phi^{-1}\left(P_j^D\right) \end{cases}$$

$$L_j = V_j^0 - \left[\left(1-D_j\right)V_j + D_j\, R_j\right]$$

Data and interface

j	v0	R	beta	PD	rtng	...
2	32606970	910000	0.707	0.0025	BBB	.
3	8932752	92800	0.707	0.0010	A	.
6	564931	335675	0.707	0.1000	CCC	.
7	3494502	82000	0.707	0.0400	B	.
9	6679886	1000500	0.707	0.0100	BB	.

j

m	1	.	.	.	J
1	0.5241	-0.484	-0.402	-0.774	-0.702
.	-2.2608	-0.666	-1.003	0.423	0.683
.	-0.0197	-0.174	-0.178	-0.607	-0.858
.	0.1831	-1.011	-0.488	0.209	0.368
M	-0.3614	0.740	0.928	-0.777	-1.430

j

m	1	.	.	.	J
1	-0.512	-0.376	-1.059	-0.165	0.511
.	-0.268	-1.135	-1.318	1.760	-3.088
.	-0.199	0.609	-0.204	1.612	0.474
.	0.857	0.566	-0.601	0.803	0.626
M	-0.167	-0.736	0.661	1.265	0.981

Usage

```
simulationKernel(pf, Z, r,
                 J,
                 K, mk = seq_len(K * nrow(Z)),
                 agg = factor(rep("PF", nrow(pf))),
                 seed)
```

total nr. of counterparties

initial RNG state

Value

aggregation criterion

agg

mk	1	.	.	A
1	2942986	2144142	2551616	2128115
.	3269994	1886979	2881280	1447524
.	3874471	1711273	2544507	3696855
.	2809653	3757694	1816909	3071337
.	4100697	2123775	2544716	1878057
.	2106241	4758459	4014828	3573142
M*K	2045032	2990315	4636829	2588612

simulations of interest

Examples and results

- Full simulation (multi-threaded)
  - Aggregation criterion: rating (credit quality)

```
L_rtng <- simulationKernel(pf, Z, r, J, K,
                           agg = pf$rtng, seed = s)
```

  - ES99: average loss in the 1% worst scenarios

```
ES99_rtng <- ES99(L_rtng)
##        BBB         AA        AAA          A         BB ...
## 9878920788 8620051762 7468245838 4796596354 1190520347 ...
```

- Consistent simulation for the sub-portfolio with BBB rating

```
L_BBB <- simulationKernel(pf %>% filter(rtng == "BBB"),
                          Z, r, J, K, seed = s)
all.equal(c(L_BBB), L_rtng[, "BBB"], check.attributes = FALSE)
## [1] TRUE
ES99_BBB <- ES99(L_BBB)
## 9878920788
```

Examples and results

- Risk insight for BBB
  - Contribution of individual counterparties to the BBB total ES99

```
pfBBB <- pf %>% filter(rtng == "BBB")
L_jBBBtail <- simulationKernel(pfBBB, Z, r, J, K, agg = pfBBB$j,
                               mk = tail99(L_BBB), seed = s)
ContrES99_jBBB <- colMeans(L_jBBBtail)
all.equal(sum(ContrES99_jBBB), ES99_BBB, check.attributes = FALSE)
## [1] TRUE
```

  - Focus on the top 3 counterparties (highest contribution)

```
pftop3BBB <- pfBBB %>% filter(j %in% top3jBBB)
L_top3BBB <- simulationKernel(pftop3BBB, Z, r, J, K,
                              agg = pftop3BBB$j, seed = s)
ES99_top3BBB <- ES99(L_top3BBB)
##  j         V0         R       ES99  ContrES99       Div   Contr/V0
##  2 9444041000 278250000 4720889738 4277383374 0.9060545 0.4529188
##  8 3260697000  91000000 1672364832  999636420 0.5977382 0.3065714
## 70  298111300  13483307  963482756  436598485 0.4531461 1.4645486
```

- What-if scenario
  - Top 3 BBB counterparties downgraded => PD from 0.0025 to 0.01

```
pfBBBwi <- pf %>% filter(rtng == "BBB") %>%
  mutate(PD = replace(PD, j %in% top3jBBB, 0.01))
pftop3BBBwi <- pfBBBwi %>% filter(j %in% top3jBBB)
L_top3BBBwi <- simulationKernel(pftop3BBBwi, Z, r, J, K,
                                agg = pftop3BBBwi$j, seed = s)
```

  - Effect on the BBB total

```
L_BBBwi <- L_BBB + (rowSums(L_top3BBBwi) - rowSums(L_top3BBB))
ES99_BBBwi <- ES99(L_BBBwi)
##  ES99_BBBwi    ES99_BBB
## 11943875892 9878920788
```

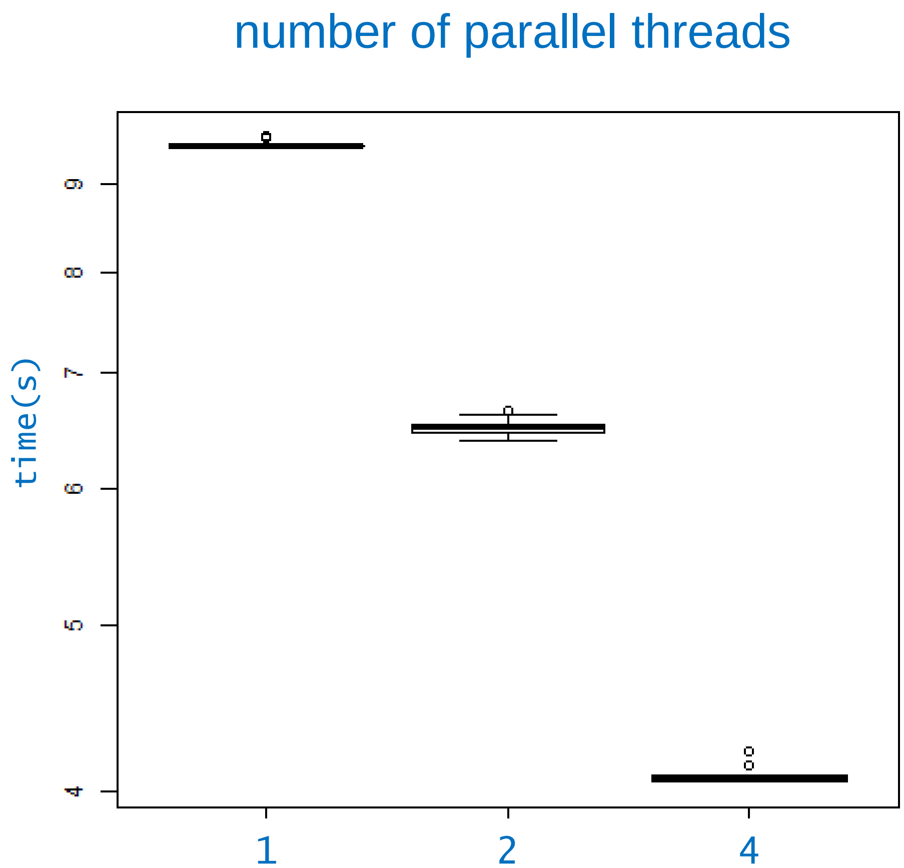
  - New contribution for the full BBB sub-portfolio

```
L_jBBBtailwi <- simulationKernel(pfBBBwi, Z, r, J, K, agg = pfBBBwi$j,
                                mk = tail99(L_BBBwi), seed = s)
```

All this achieved without re-simulating the whole BBB portfolio!

Performance benchmark

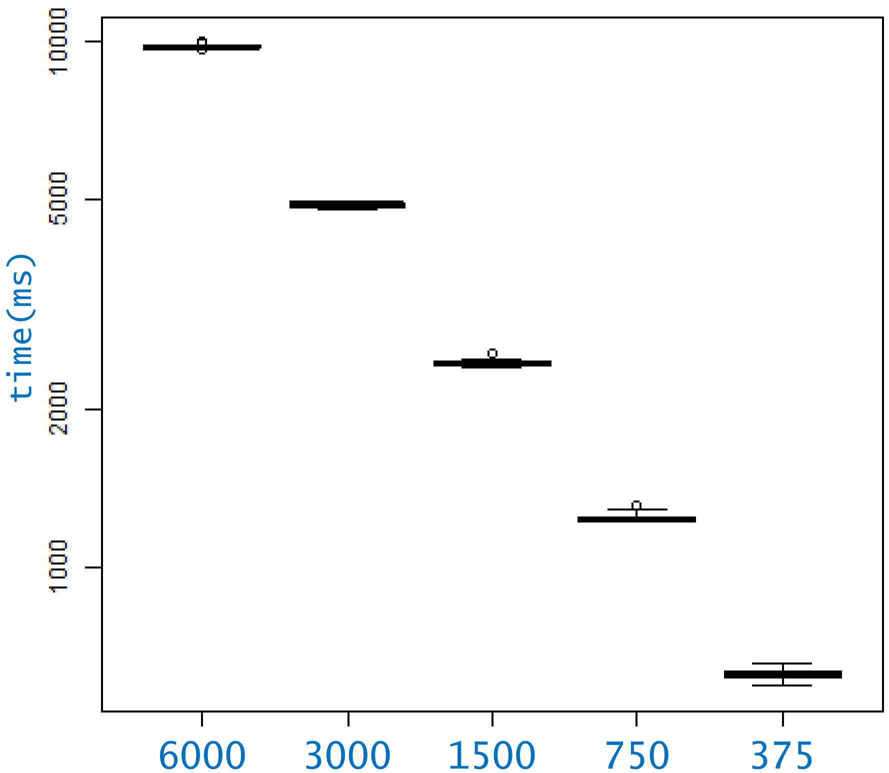
`microbenchmark` results (M=1000, K=10)



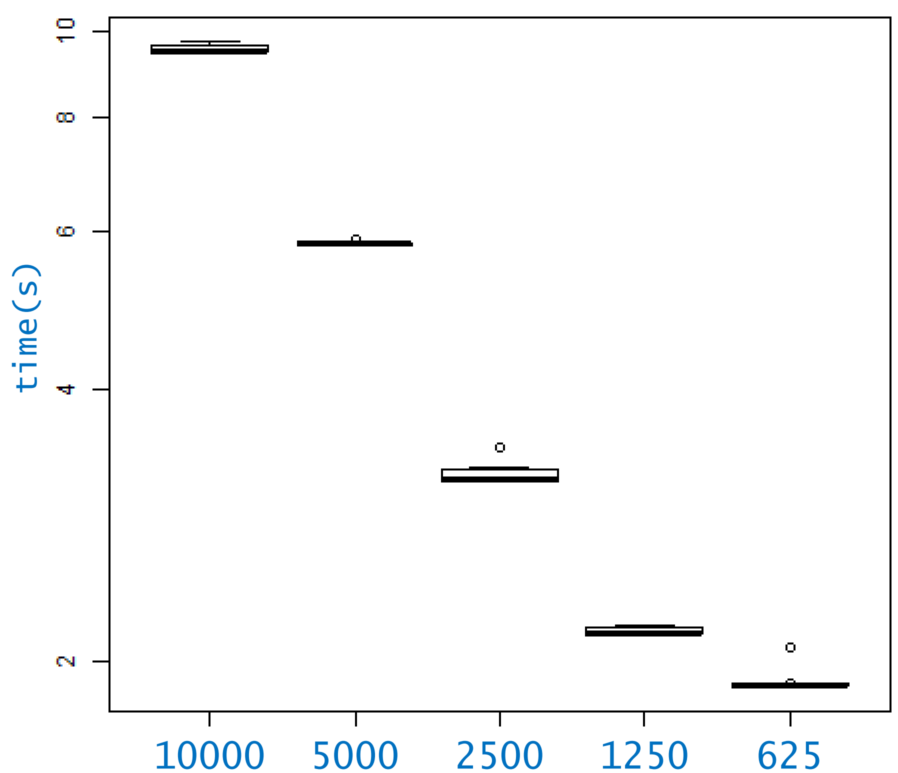
Performance benchmark

`microbenchmark` results (M=1000, K=10)

size of the sub-portfolio



number of sub-simulations



Summary

- Monte Carlo simulation of an integrated market and default risk model
  - Flexible, consistent, slim, multi-purpose simulation kernel
- TRNG state-of-the-art parallel random number generators
  - rTRNG for prototyping in R and broader usage in R/C++ projects
- Flexible and fast ad-hoc assessments on sub-portfolios, simulations of interest, what-if scenarios
- Incremental simulations and updates possible
- Can also be used for driver or change analysis, isolating away the MC variability

=> Achieve fast re-simulation instead of

- storing GBs or TBs of granular results
- using complex analytic approximation models that are hard to explain and understand