RNG Slides

Adam Kaplan

March 30, 2022

Random number generation with Futures

- **Motivating example:** Monte Carlo simulations.

Random number generation with Futures

- **Motivating example:** Monte Carlo simulations.
- To ensure replicability we would usually set a random seed using `set.seed(02135)` before running the simulations.

Random number generation with Futures

- **Motivating example:** Monte Carlo simulations.
- To ensure replicability we would usually set a random seed using `set.seed(02135)` before running the simulations.
- Unfortunately, when we parallelize, since each process is independent of each other, it is very likely that the random state *across* processes is no longer pseudo-random. In fact, it will likely be correlated.

Random number generation with Futures

- **Motivating example:** Monte Carlo simulations.
- To ensure replicability we would usually set a random seed using `set.seed(02135)` before running the simulations.
- Unfortunately, when we parallelize, since each process is independent of each other, it is very likely that the random state *across* processes is no longer pseudo-random. In fact, it will likely be correlated.
- Even the naive solution of setting a different seed for each iteration does not guarantee the desired level of randomness.

Random number generation with Futures

- **Motivating example:** Monte Carlo simulations.
- To ensure replicability we would usually set a random seed using `set.seed(02135)` before running the simulations.
- Unfortunately, when we parallelize, since each process is independent of each other, it is very likely that the random state *across* processes is no longer pseudo-random. In fact, it will likely be correlated.
- Even the naive solution of setting a different seed for each iteration does not guarantee the desired level of randomness.
- Thankfully, there are random number generation algorithms specifically designed for parallel computing. The default for `future` and most commonly used one is *L'Ecuyer CMRG algorithm*.

Random number generation with Futures

- **Motivating example:** Monte Carlo simulations.
- To ensure replicability we would usually set a random seed using `set.seed(02135)` before running the simulations.
- Unfortunately, when we parallelize, since each process is independent of each other, it is very likely that the random state *across* processes is no longer pseudo-random. In fact, it will likely be correlated.
- Even the naive solution of setting a different seed for each iteration does not guarantee the desired level of randomness.
- Thankfully, there are random number generation algorithms specifically designed for parallel computing. The default for `future` and most commonly used one is *L'Ecuyer CMRG algorithm*.
- Using it is as easy as in a sequential program.

Random number generation in future.apply

```
library(future.apply)
# Set the seed
set.seed(02135)
# Specify that you want to use random numbers
out1 <- future_sapply(1:10, function(x) runif(1),
  future.seed = TRUE
)
```

Random number generation in furrr

```r
library(furrr)
# Set the seed
set.seed(02135)
# Specify that you want to use random numbers
out2 <- future_map_dbl(1:10, ~ runif(1),
  .options = furrr_options(seed = TRUE)
)
# Check that they are the same
all(out1 == out2)
```

```
## [1] TRUE
```