TMA4300 Computer Intensive Statistical Methods

- Home page: https://wiki.math.ntnu.no/tma4300/2017v/
- Lecturer: Håkon Tjelmeland
- Teaching assistant: Xin Luo

Access to the computer lab

- For those who do not have access to Nullrommet 380A: please send me (haakont@math.ntnu.no) a mail as soon as possible with your name, NTNU username/e-mail address and study programme.
- You will get:
  - Physical access to the computer lab
  - User at the (linux) computers
- Don't wait, do it today!

Partner for the exercises

- The exercises should be done in groups of two persons. You can
  - find an exercise partner yourself. If so, send an email with (the two) names and email addresses to the teaching assistant (xin.luo@math.ntnu.no).
  - let the teaching assistant find exercise partner for you. If so, send an email with your name and email address to the teaching assistant (xin.luo@math.ntnu.no) and ask for this.
- Don't wait, do it today. Deadline: January 19th.

Reference group

- Three students should be in the reference group.
  - Preferably from different study programmes.
- Duties:
  - Stay in dialogue with the other students.
  - Participate in three meetings with the lecturer and the teaching assistant, and provide suggestions for improvement in exercises, lectures, home page, and so on.
  - Write a joint brief final report providing constructive feedback and evaluation of the course. This report will be published unedited in the course evaluation.
- Volunteers?

Course outline

- The course is divided in three topic blocks:

Part 1: Algorithms for stochastic simulation.
Part 2: Markov chains Monte Carlo methods
Part 3: Expectation-maximisation algorithms, bootstrap and classification methods

- Each part consists of:
  - A number of lectures
  - Exercises (obligatory)
  - Oral presentations (obligatory)

TMA4300: Learning outcome, Knowledge

- The student knows computational intensive methods for doing statistical inference.
- This includes direct and iterative Monte Carlo simulations, as well as the expectation-maximisation algorithm and the bootstrap.
- The student has basic knowledge in how hierarchical Bayesian models can be used to formulate and solve complex statistical problems.
- Finally, the student understands a number of classification techniques.

TMA4300: Learning outcome, Skills

- The student can apply computational methods, such as Monte Carlo simulations, the expectation-maximisation algorithm and the bootstrap, on simple applied problems.
- General competence. The student is able to give an oral presentation where he or she communicate his or her findings in a project.

Do you have experience with . . .

- Markov chains?
- The computer language `R`?
- Basic Bayesian inference?

The word simulation ...

...refers to the treatment of a real problem through reproduction in an environment controlled by the experimenter.

Gamerman & Lopes, Markov Chain Monte Carlo, 2nd Edition, Page 9

Motivation: Queueing problem

M/G/1 - queue:

- Customers arrive to a queueing system according to a Poisson process, i.e. interarrival times are exponentially distributed.
- One server
- Independent service times distributed according to $f(t)$.
- Queue system empty at time $t = 0$

$X(t)$ customers in queueing system at time t.

Motivation: Queueing problem

If service times are exponentially distributed, $X(t)$ is a Markov process and an explicit analytical formula for the limiting distribution is available. For general f, analytical solutions might not be available.

$\Rightarrow$ Idea: Simulate the queueing process on a computer and return the quality of interest, e.g. $\min\{t > 0 : X(t) \geq 7\}$.

Simulation, why do we need it?

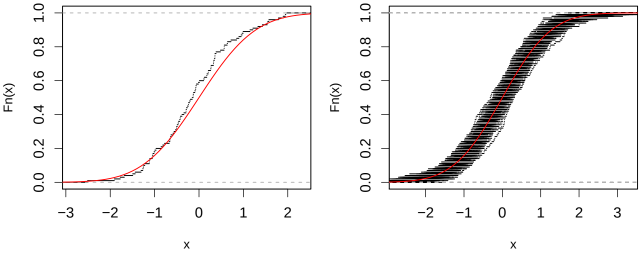
Necessity to produce chance on the computer:

- Evaluation of the behaviour of a complex system (Epidemics, weather forecast, networks, economic actions, etc)

Simulation, why do we need it?

Necessity to produce chance on the computer:

- Evaluation of the behaviour of a complex system (Epidemics, weather forecast, networks, economic actions, etc)
- Determine probabilistic properties of a novel statistical procedure or under an unknown distribution.

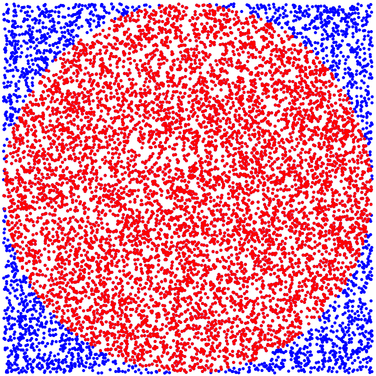


(Left: Estimation of CDF from a normal sample of 100 points,
Right: Variation of the estimation over 200 samples.)

Simulation, why do we need it?

- Approximation of an integral/area

```
n = 1000            ▷ (# of simulations)
m = 0               ▷ (# points in circle)
i = 1
while i < n do
    x = Rand(1), y = Rand(1)
    if x^2 + y^2 < 1 then
        m ← m + 1
    end if
    i = i + 1
end while
return 4 · m/n
```



$\hat{\pi} = 3.1353.$

Simulation, how do we do it?

Central building block of simulation: Always requires availability of uniform $\mathcal{U}(0,1)$ random variables.

```
> runif(10)
 [1] 0.96112670 0.83981815 0.77269526 0.03363567 0.89374420 0.25631906
 [7] 0.82222815 0.77355197 0.17093594 0.12035040
```

Pseudo-random generator

A pseudo-random generator is a deterministic function f which takes a uniform random bit string as input and outputs a bit string which cannot be distinguished from a uniform random string.

In more detail, this means that for starting value (seed) u_0 and any n, the sequence

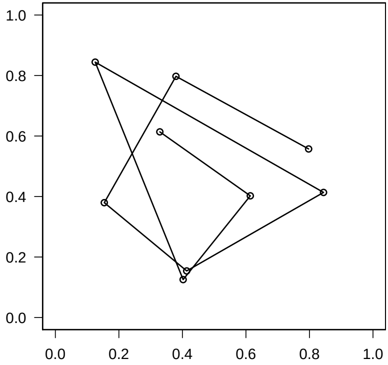
$$\{u_0, f(u_0), f(f(u_0)), f(f(f(u_0))), \ldots, f^n(u_0)\}$$

behaves statistically like an $\mathcal{U}(0,1)$ sequence (when appropriately scaled).

Pseudo-random generator

Illustration of first 10 (u_t, u_{t+1}) steps

```
> par(mfrow=c(1,1), mar=c(4,4,1,0), las=1, cex.lab=1.2, cex.axis=1.1, lwd=1.6)
> set.seed(2118735)
> a = runif(10)
> plot(a[1:9], a[2:10], type="o", xlab="", ylab="", xlim=c(0,1), ylim=c(0,1))
```



A standard uniform generator

The congruential random generator on $\{0, 1, \ldots, M-1\}$

$$f(x) = (a \cdot x + b) \mod M$$

has a period equal to M for proper choices (a, b) and becomes a generator on $[0, 1)$ when dividing by M.

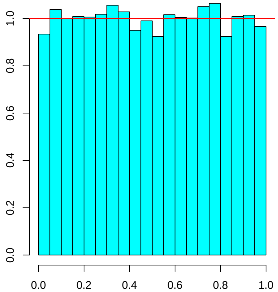
Example

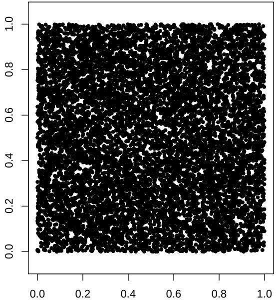
Take

$$f(x) = (69069069 \cdot x + 12345) \mod (2^{32})$$

and produce

$\ldots, 69081414, 2406887111, 1109307232, 2802677792, 3651430880, 806776992, \ldots$
i.e.
$\ldots, 0.01608427, 0.56039708, 0.25828072, 0.65254927, 0.85016500, 0.18784241, \ldots$





Random number generator

From now on, we assume to have a random generator on the unit interval available.