Introduction

TMA4300: Computer Intensive Statistical Methods
(Spring 2019)

Sara Martino

[1]

[1] Slides are based on lecture notes kindly provided by Håkon Tjelmeland and Andrea Riebler.

General information

Lectures:

- Lectures: Sara Martino, room 1140
  `sara.martino@math.ntnu.no`
- Tuesday 10.12-12 ,
  Thursday 12.15-14:

Computer exercises:

- Exercises:Jorge Sicacha Parada, room 1144,
  `jorge.sicacha@ntnu.no`
- Tuesday 10.12-12 ,
  Thursday 12.15-14:
- Extra hour: Friday 10.15-12

See course webpage regularly for time plan!

Access to computer lab Nullrommet 380A

For those who do not have access to Nullrommet 380A: please send me (`sara.martino@math.ntnu.no`) as soon as possible your name, student number, NTNU username/e-mail address and study programme.

TMA4300: Course webpage

`https://wiki.math.ntnu.no/tma4300/2019v/start`

Please check this website regularly!

- Messages
- Course information
- Curriculum
- Lecture plan
- Exercise classes
- Statistical software
- Reference group
- Exam

Quality ensurance: Reference group

Three members preferably from different study programmes, such as Industrial Mathematics, Erasmus Programme, others.

Duties:

- Stay in dialogue with all students.
- Participate in three meetings distributed over the semester.
- Give feedback to lecturer and teaching assistant about lectures and exercises, and provide suggestions for improvement.
- Write a joint final report providing constructive feedback and evaluation of the course. This report will be published unedited in course evaluation.

A certificate will prove participation in the reference group.

Volunteers?

Course outline

Reference book:

- Givens, G.H., Hoeting, J.A., 2013, *Computational Statistics*, 2nd edition, John Wiley & Sons.

  The book is freely available as e-book within the NTNU network from the library.

- Gamerman, D. Lopes, H.F. 2006. *Markov chain Monte Carlo - Stochastic Simulation for Bayesian Inference*, 2nd edition

- Extra references might be used.

Course Structure

The course is divided in three topic blocks:

Part 1: Algorithms for stochastic simulation

Course Structure

The course is divided in three topic blocks:

Part 1: Algorithms for stochastic simulation

Part 2: Markov chains Monte Carlo methods and INLA

Course Structure

The course is divided in three topic blocks:

Part 1: Algorithms for stochastic simulation

Part 2: Markov chains Monte Carlo methods and INLA

Part 3: Expectation-maximisation algorithms, bootstrap.

Exercises

- Each lecture block is followed by an exercise block.

Exercises

- Each lecture block is followed by an exercise block.
- Exercises are obligatory. If you did the exercises in a previous year and you were admitted to the exam, you will still be admitted to the exam this year. However, the points you obtained will not be transferred. Thus, you get credit only for exercise work made during this semester.

Exercises

- Each lecture block is followed by an exercise block.
- Exercises are obligatory. If you did the exercises in a previous year and you were admitted to the exam, you will still be admitted to the exam this year. However, the points you obtained will not be transferred. Thus, you get credit only for exercise work made during this semester.
- The exercises account for 30% of the final mark. The final exam counts for 70%. You must pass the final exam to pass the course (exercises + final exam).

Exercises

Each lecture block is followed by an exercise block

- The exercises MUST be done in groups of two persons.

Register your group until **18 January** by sending an email to (`jorge.sicacha@math.ntnu.no`). If you need help to find a team partner please send also an email to Jorge.

Exercises

Each lecture block is followed by an exercise block

- The exercises MUST be done in groups of two persons.

Register your group until **18 January** by sending an email to (`jorge.sicacha@math.ntnu.no`). If you need help to find a team partner please send also an email to Jorge.

- The exercises have to be done using the statistical package `R`.

Exercises

Each lecture block is followed by an exercise block

- The exercises MUST be done in groups of two persons.

Register your group until **18 January** by sending an email to (`jorge.sicacha@math.ntnu.no`). If you need help to find a team partner please send also an email to Jorge.

- The exercises have to be done using the statistical package `R`.
- The exercises account for 30% of the final mark.

Exercises

Each lecture block is followed by an exercise block

- The exercises MUST be done in groups of two persons.

Register your group until **18 January** by sending an email to (`jorge.sicacha@math.ntnu.no`). If you need help to find a team partner please send also an email to Jorge.

- The exercises have to be done using the statistical package `R`.
- The exercises account for 30% of the final mark.
- In an oral presentation each group will once present their finding on a selected part of the exercises.

Exam

- The exam will be on 5th June 2019
- Examination aids: C (to be decided on)
- The exam counts for 70% of the final mark and must be passed in order to pass the course.

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.

Statistical software R

R is available for free download at The Comprehensive R Archive Network (Windows, Linux and Mac).

- Rstudio `http://www.rstudio.org` is an integrated development environment (system where you have your script, your run-window, overview of objects and a plotting window).
- A nice introduction to R is the book P. Dalgaard: Introductory statistics with R, 2nd edition, Springer which is also available freely to NTNU students as an ebook.

Intro to R

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 (II)

What are we interested in:

- limiting distribution of $X(t)$, ie

$$\lim_{t \to \infty} P\{X(t) = i\}, i = 0, 1, \ldots$$

- Other quantities, for example

$$T = \min\{t > 0; X(t) > 7\}$$

Motivation: Queueing problem (III)

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. $\min\{t > 0 : X(t) \geq 7\}$.

Show code `Queue.R`

Simulation, why do we need it?

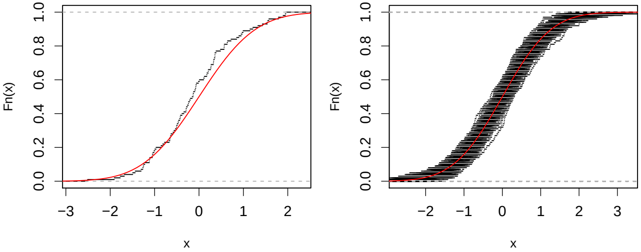
Necessity to produce chance on the computer:

- 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**

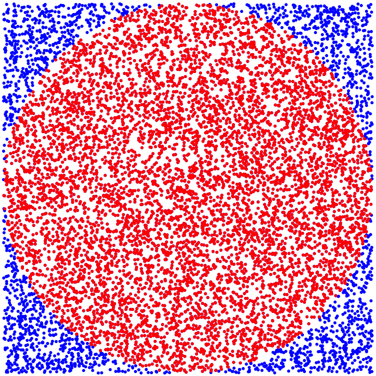
    $m \leftarrow m + 1$

  end if

  $i = i + 1$

end while

return $4 \cdot m/n$



$\hat{\pi} = 3.1353.$

Pseudo-random generator

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

```
[1] 0.6925103 0.2345950 0.7856542 0.3596010 0.2180535 0.2461511 0.7186286
[8] 0.7482534 0.6525447 0.5339895
```

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.

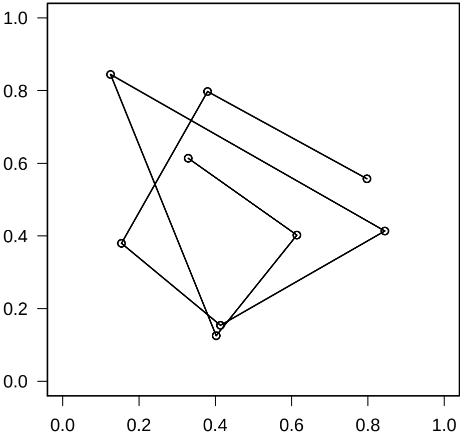
In more detail, this means that for starting value u_0 and any n, the sequence

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

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



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

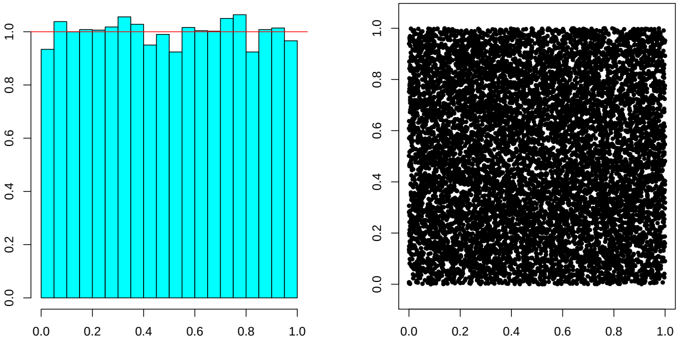
$$f(x) = (69069069 \cdot x + 12345) \mod (2^{32})$$

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.

Show code

Discrete and continuous random variables

Discrete

- Takes values in $\mathcal{N}$
- Probability mass func. $f(x)$

$$f(x) = \text{Prob}(X = x)$$

- Cum. distribution func. $F(x)$

$$F(x) = \sum_{u \leq x} f(u) = \text{Prob}(X \leq x)$$

Continuous

- Takes values in $\mathcal{R}$
- Probability density func. $f(x)$

$$\text{Prob}(a < X < b) = \int_a^b f(u)du$$

- Cum. distribution func. $F(x)$

$$\begin{aligned} F(x) &= \text{Prob}(X \leq x) \\ &= \int_{-\infty}^{x} f(u)du \end{aligned}$$

Normalising constant

A normalising constant c is a multiplicative term in $f(x)$, which does not depend on x. The remaining term is called core:

$$f(x) = c \underbrace{g(x)}_{\text{core}}$$

We often write $f(x) \propto g(x)$.

Examples of continuous distributions

- Uniform distribution $f(x) \propto 1$
- Exponential distribution $f(x) \propto e^{-\lambda x}$
- Normal distribution $f(x) \propto \exp\{-\frac{1}{\sigma^2}(x - \mu)^2\}$.
- ...

Generating from standard parametric families

We would like to find strategies to generate random variates from familiar distributions based on uniform random variates.

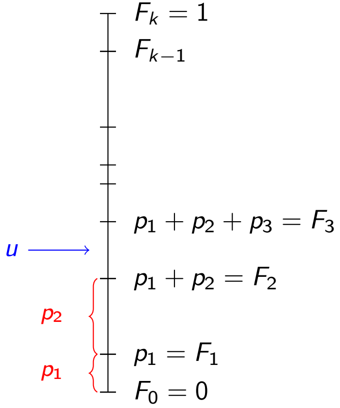
Discrete distributions

Let X be a stochastic variable with possible values $\{x_1, \ldots, x_k\}$ and $\mathrm{P}(X = x_i) = \mathrm{p}_i$. Of course $\sum_{i=1}^{k} p_i = 1$.

An algorithm for simulating a value for x is then:

$u \sim U[0, 1]$
for $i = 1, 2, \ldots, k$ **do**
  if $u \in (F_{i-1}, F_i]$ **then**
    $x \leftarrow x_i$
  end if
end for

Each interval $I_i = (F_{i-1}, F_i]$ corresponds to single value of x.



Proof & Note

Proof.

See Blackboard $\square$

Proof & Note

Proof.

See Blackboard $\square$

Note: We may have $k = \infty$

- The algorithm is not necessarily very efficient. If k is large, many comparisons are needed.
- This generic algorithm works for any discrete distribution. For specific distributions there exist alternative algorithms.

Bernoulli distribution

Let $S = \{0, 1\}$, $\mathrm{P}(X = 0) = 1 - p$, $\mathrm{P}(X = 1) = p$.

Thus $X \sim \mathrm{Bin}(1, p)$.

The algorithm becomes now:

$u \sim U[0, 1]$

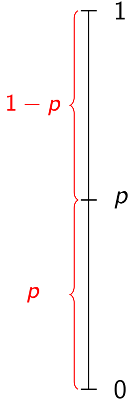
if $u \leq p$ **then**

$\quad x = 1$

else

$\quad x = 0$

end if



Binomial distribution

Let $X \sim \text{Bin}(n, p)$.

The generic algorithm from before can be used, but involves tedious calculations which may involve overflow difficulties if n is large.

An alternative is:

$x = 0$
for $i = 1, 2, \ldots, n$ **do**
  generate $u \sim U[0, 1]$
  if $u \leq p$ **then**
    $x \leftarrow x + 1$
  end if
end for
return x

Geometric and negative binomial distribution

The negative binomial distribution gives the probability of needing x trials to get r successes, where the probability for a success is given by p. We write $X \sim \mathrm{NB}(r, p)$.

The generic algorithm can still be used, but an alternative is:

$s = 0$ ▷ (# of successes)
$x = 0$ ▷ (# of tries)
while $s < r$ **do**
  $u \sim U[0, 1]$
  $x \leftarrow x + 1$
  if $u \leq p$ **then**
    $s \leftarrow s + 1$
  end if
end while
return x

Poisson distribution

Let $X \sim \text{Po}(\lambda)$, i.e. $f(x) = \frac{\lambda^x}{x!} e^{-\lambda}$, $x = 0, 1, 2, \ldots$.

An alternative to the generic algorithm is:

$x = 0$ ▷ (# of events)

$t = 0$ ▷ (time)

while $t < 1$ **do**

  $\Delta t \sim \text{Exp}(\lambda)$

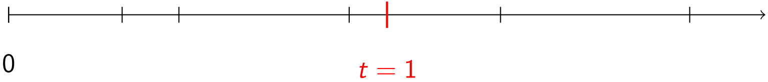
  $t \leftarrow t + \Delta t$

  $x \leftarrow x + 1$

end while

$x \leftarrow x - 1$

return x



It remains to decide how to generate $\Delta t \sim \text{Exp}(\lambda)$.

Sampling from continuous distribution functions

The inversion method (or probability integral transform approach) can be used to generate samples from an arbitrary continuous distribution with density $f(x)$ and CDF $F(x)$:

1. Generate random variable U from the standard uniform distribution in the interval $[0, 1]$.
2. Then is

   $$X = F^{-1}(U)$$

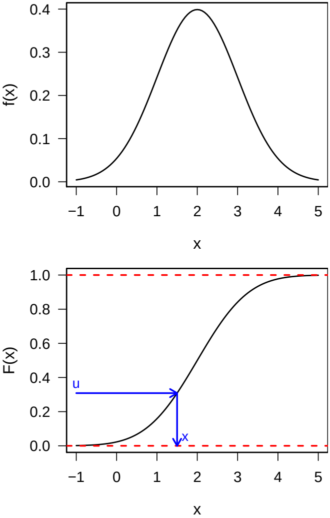
   a random variable from the target distribution.

Proof.

See blackboard □

Inverse cumulative distribution function (II)

Let X have density $f(x)$, $x \in \mathbb{R}$ and CDF $F(x) = \int_{-\infty}^{x} f(z)dz$:



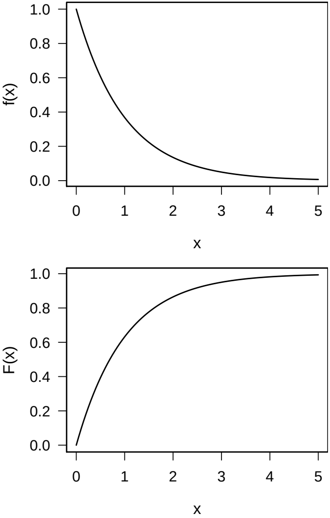
Simulation algorithm:

$u \sim U[0,1]$

$x = F^{-1}(u)$

return x

Example - Exponential Distribution



$$f(x) = \lambda \exp(-\lambda x) : \; x > 0$$

$$F(x) = 1 - \exp(-\lambda x)$$

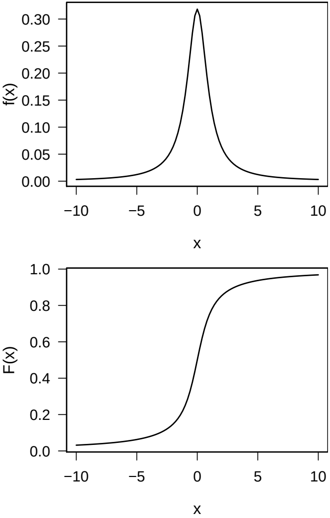
Simulation algorithm:

$u \sim U[0,1]$

$x = -\frac{1}{\lambda} \log(u)$

return x

Example - Standard Cauchy distribution



$$f(x) = \frac{1}{\pi} \cdot \frac{1}{1+x^2}$$
$$F(x) = \frac{1}{2} + \frac{\arctan(x)}{\pi}$$
$$F^{-1}(y) = \tan\left[\pi\left(y - \frac{1}{2}\right)\right]$$

Simulation algorithm:

$u \sim U[0,1]$

$x = \tan[\pi(U_i - \frac{1}{2})]$

return x

Inversion Sampling

Can we always use this algorithm??

- We have to be able to find $F^{-1}(u)$
- Not possible for many important distribution
  - Normal
  - Gamma
  - ...
- Need to know the normalizing constant