TMA4300: Computer Intensive Statistical Methods

Spring 2023

John Paige

Plan for today

- Course Info
- Why simulation?
- Pseudo random numbers
- Discrete and continuous RV
- Sample from standard parametric discrete distribution
- Sample from standard parametric continuous distribution

General information

- Course page on Blackboard
- Lecturers: John Paige
- Teaching Assistant: Guillermina Senn

Lectures and Exercise Sections:

- Lecture: Monday 12.15-14.00 in R54
- Lecture: Tuesday 8.15-10.00 in MA24
- Exercise: Friday 10.15-12 in R54

Poll: switch class from Tuesdays to Fridays?

Quality ensurance: Reference group

Two or 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.

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 modules:

Part 1: Algorithms for stochastic simulation

Part 2: Markov chains Monte Carlo methods and INLA

Part 3: Expectation-maximization algorithm, bootstrap.

TMA4300: Topics of the course

- Simulation & Bayesian inference:
  - Generation from standard parametric families
  - Inverse cumulative distribution function
  - Rejection sampling
  - ...
- Estimation of Bayesian models
  - Metropolis-Hastings
  - Gibbs
  - Implementation & Output analysis
  - Approximate Bayesian inference (INLA)
- Expectation Maximization (EM) algorithms, bootstrap

TMA4300 learning outcome: Knowledge

- The student will know computational intensive methods for statistical inference
- This includes direct and iterative Monte Carlo simulations
- The student has basic knowledge in how hierarchical Bayesian models can be used to formulate and solve complex statistical problems.
- Finally, the student understands the basics of the expectation maximization algorithm and the bootstrap

TMA4300 learning outcome: Skills

- Statistical ability. The student can apply computational methods, such as Monte Carlo simulations, the expectation maximization algorithm and the bootstrap, on simple applied problems.
- Group collaboration. The student can collaborate in groups to write reports providing exercise solutions.
- Oral presentation. The student is able to give an oral presentation where he or she communicate his or her findings in a project.

Exercises

- Each lecture module is followed by an exercise
- 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

- The exercises MUST be done in groups of two or three people.

Register your group by **January 25th** by sending an email to (`guillermina.senn@ntnu.no`). If you need help to find a team partner please send also an email to Guillermina.

- The exercises must be done with `R`. See the "Learning R" page on Blackboard
- The following will be considered when grading exercises: correctness, clarity of presentation, discussion of results.
- The exercises account for 30% of the final mark.
- In an oral presentation, each group will present their findings on a selected part of the exercises once throughout the semester.

Exam

- At the moment the exam is planned as a school exam
- The exam counts for 70% of the final mark and must be passed in order to pass the course.

Statistical software R

See the "Learning R" tab on Blackboard for more information on 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.
- An online course is available at:
  `https://digit.ntnu.no/courses/course-v1:NTNU+IMF001+2020/course/`

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.

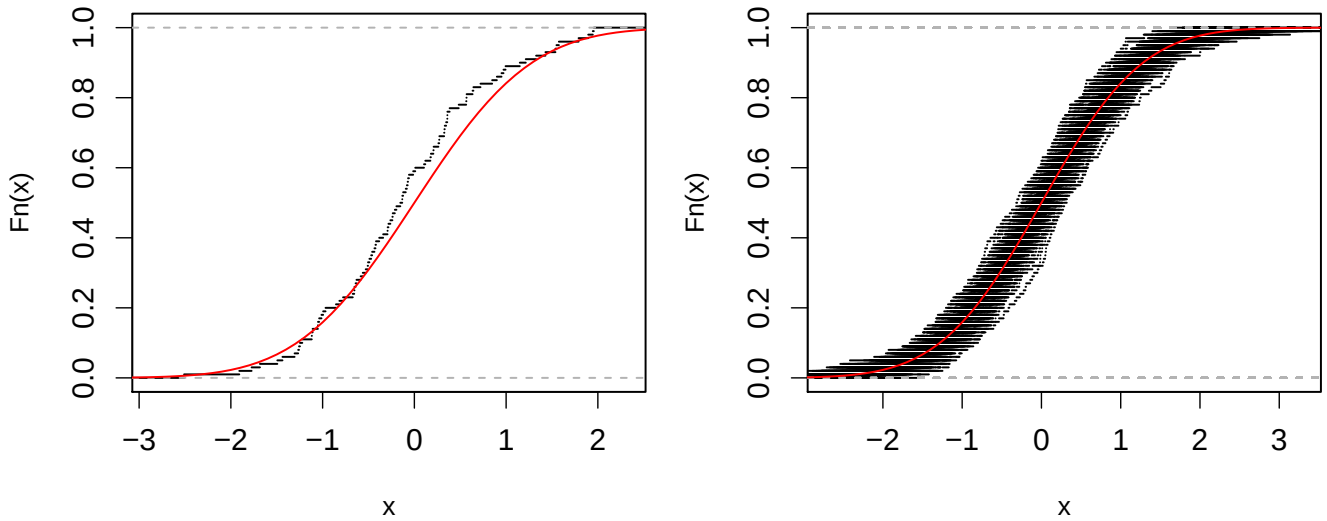
$\Rightarrow$ Idea: Simulate the queueing process on a computer and return the quality of interest, e.g. $\min\{t > 0 : X(t) \geq 7\}$.

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)
- 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 (Show code `Area.R`)

$n = 1000$ ▷ (# 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**

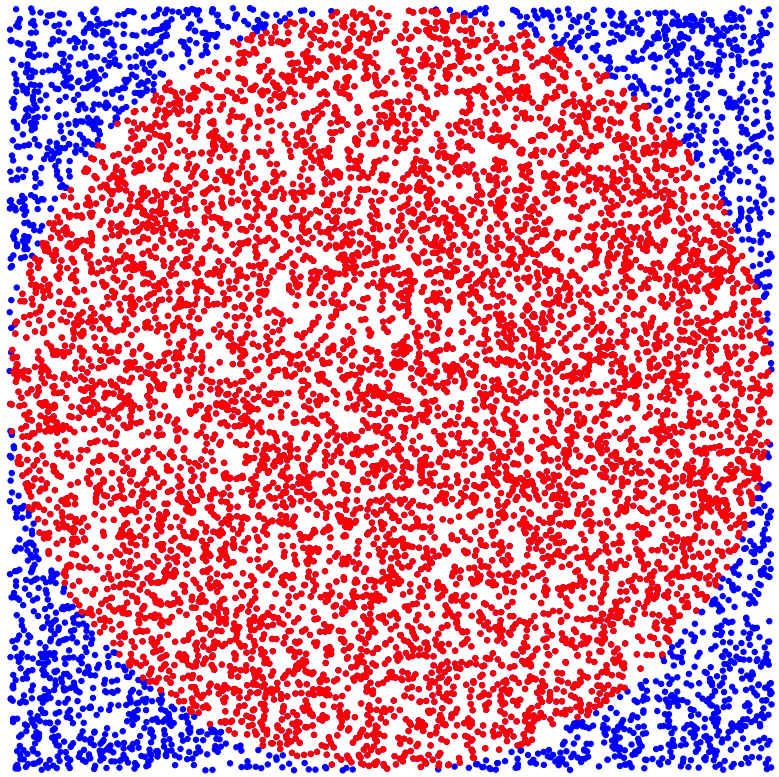
    $m \leftarrow m + 1$

  end if

  $i = i + 1$

end while

return $4 \cdot m/n$

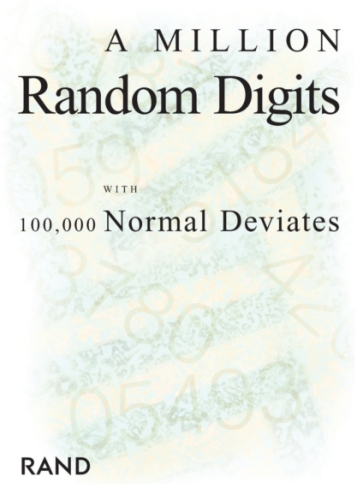


$\hat{\pi} = 3.1353.$

How to generate random number? (from uniform distribution..)

- A old need....
- Became more important in the 1940s when computer were invented

[1]



[2]

[1] Source:https://quatr.us/west-asia/dice-invented-history-dice.htm

[2] Source:https://www.amazon.it/Million-Random-Digits-Normal-Deviates/dp/0833030477

How to generate random number? (from uniform distribution..)

Real Random Numbers

- Based of physical devices
- problems with biaseness (hard to get independence)
- Often slow
- not reproducible

Pseudo Random Numbers

- Based on a deterministic algorithm that imitates randomness
- Fast
- Reproducible
- Problems with cycles

Pseudo-random generator

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

```
> runif(10)
 [1] 0.799261861 0.088982982 0.967558112 0.856558798 0.007653082 0.290359230
 [7] 0.701334139 0.303125717 0.384106228 0.124992917
```

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

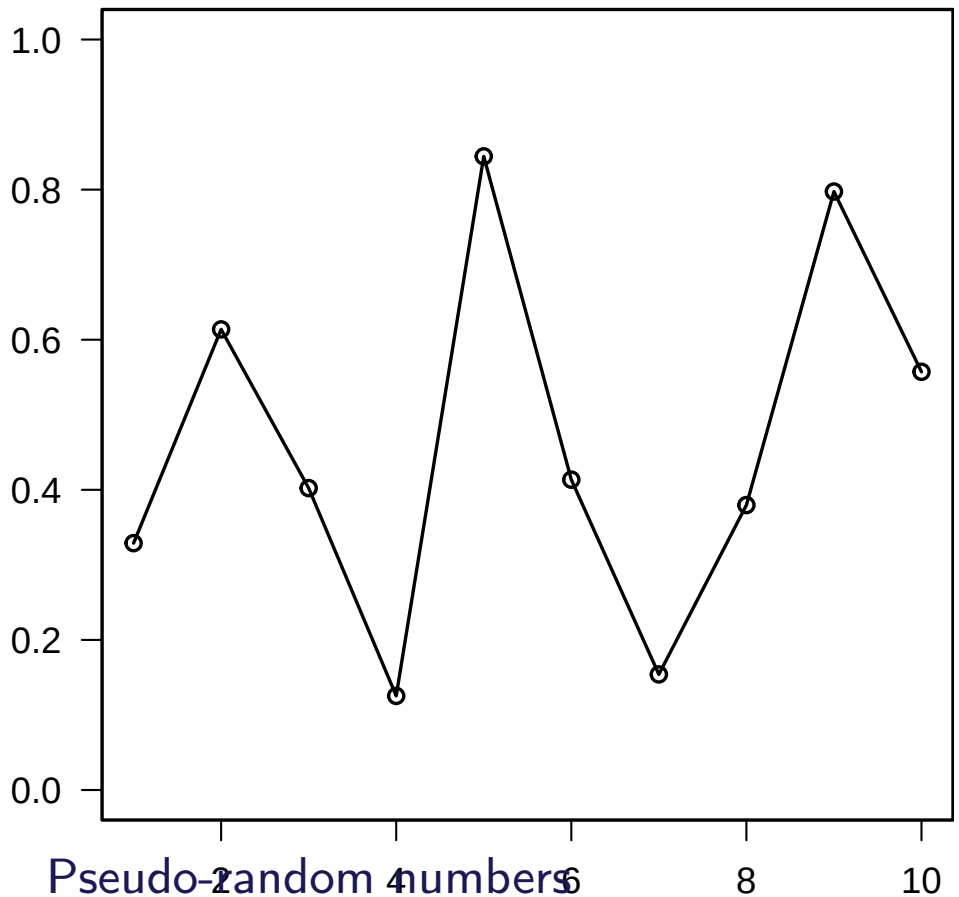
$$\{u_0, f(u_0), f(f(u_0)), f(f(f(u_0))), \ldots, f^n(u_0)\}$$

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

```
> 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, type="o", xlab="", ylab="", 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

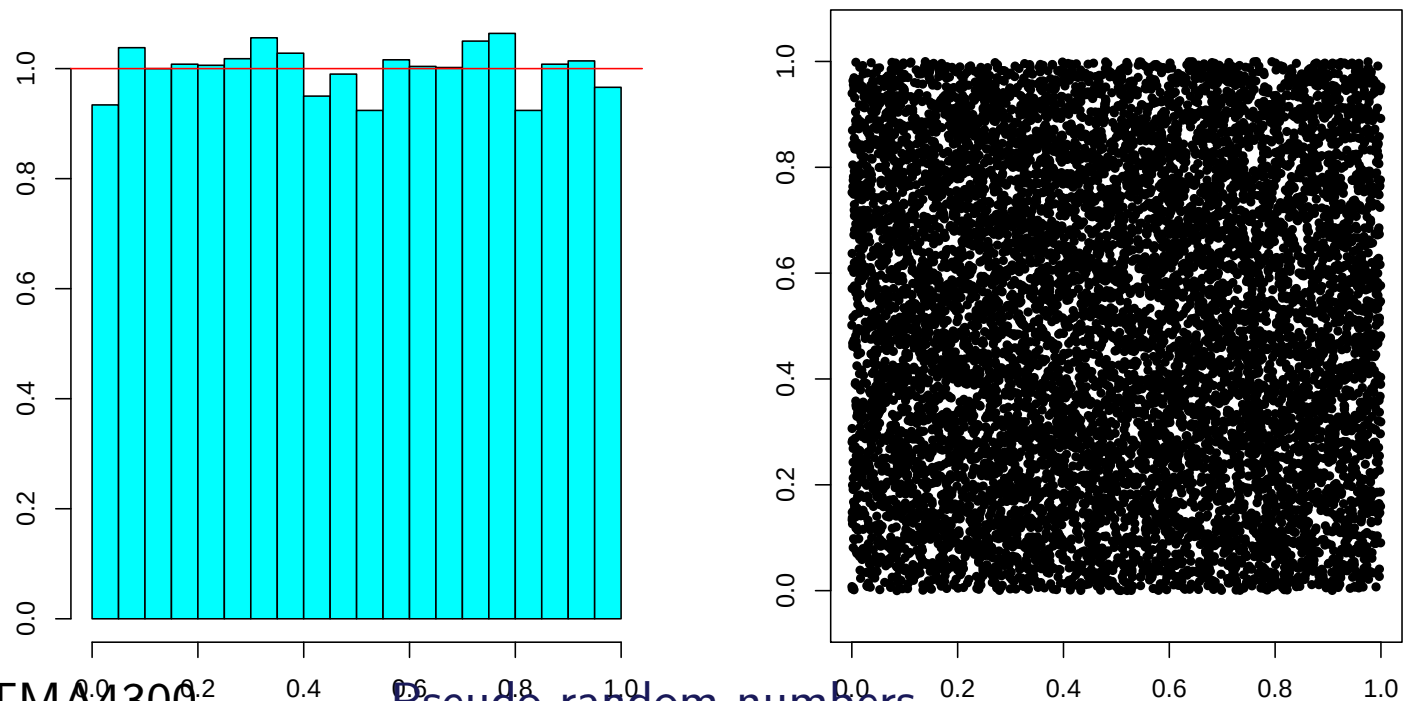
$$f(x) = (69069069 \cdot x + 12345) \mod (2^{32})$$

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.

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. $F(x)$

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

Examples of Discrete and continuous random variables

Discrete

- Bernoulli distribution
- Binomial distribution
- Geometric/Negative binomial distribution
- Poisson distribution
- ...

Continuous

- Uniform distribution
- Exponential distribution
- Normal distribution
- ...

Normalising constant

A normalising constant c is a multiplicative term in $f(x)$, which does not depend on x. The remaining term is called the 'kernel' or '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 $f(x) = \lambda e^{-\lambda x} \propto e^{-\lambda x}$
- Normal distribution

  $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{1}{\sigma^2}(x-\mu)^2\} \propto \exp\{-\frac{1}{\sigma^2}(x-\mu)^2\}.$
- ...

Generating from standard parametric families

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

Discrete and continuous random variables

Let X be a stochastic variable with possible values $\{x_1, \ldots, x_k\}$ and $P(X = x_i) = p_i$.

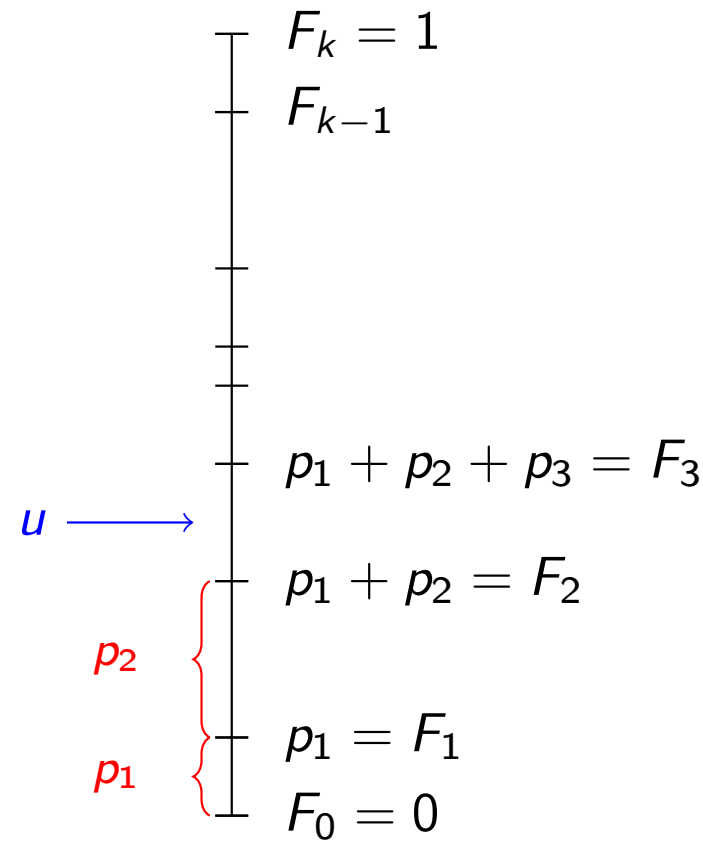
Of course $\sum_{i=1}^{k} p_i = 1$.

Discrete distributions

Let X be a stochastic variable with possible values $\{x_1, \ldots, x_k\}$ and $P(X = x_i) = p_i$. 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 □

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:

$u \sim U[0, 1]$

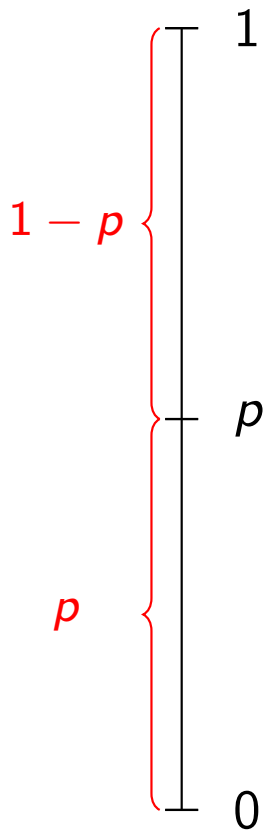
if $u \leq p$ **then**

$x = 1$

else

$x = 0$

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 $X \sim \text{NB}(r, p)$.

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**

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

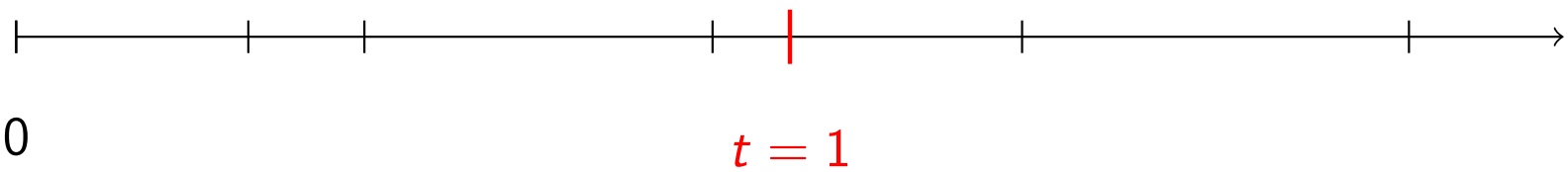
  $t \leftarrow t + \Delta t$

  $x \leftarrow x + 1$

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

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

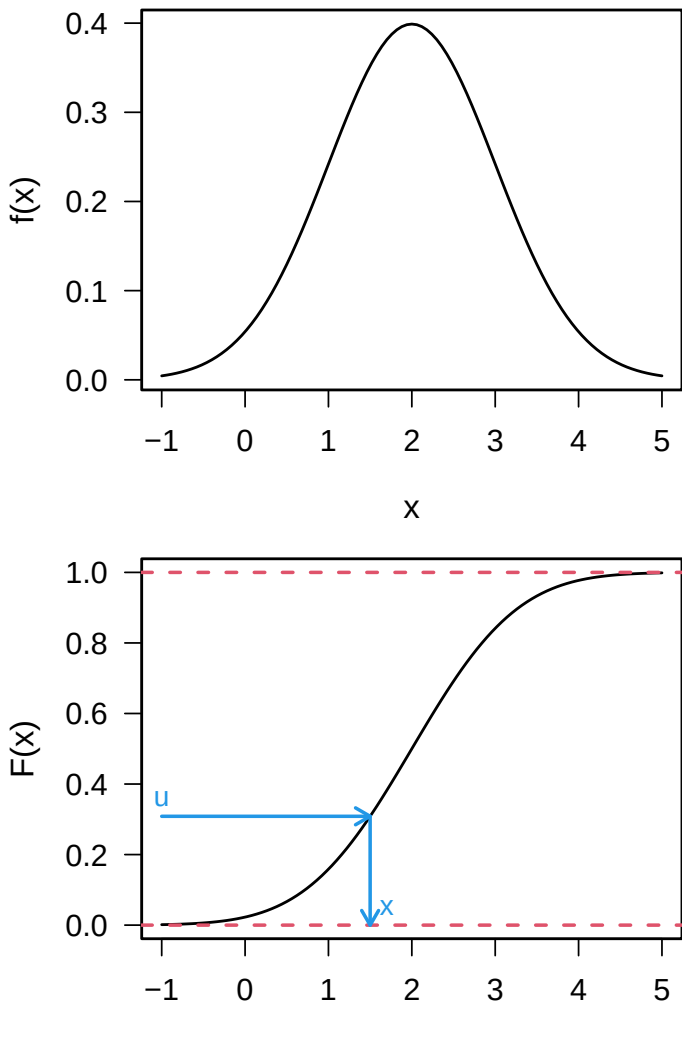
is a random variable from the target distribution.

Proof.

See blackboard □

Inverse cumulative distribution function (II)

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



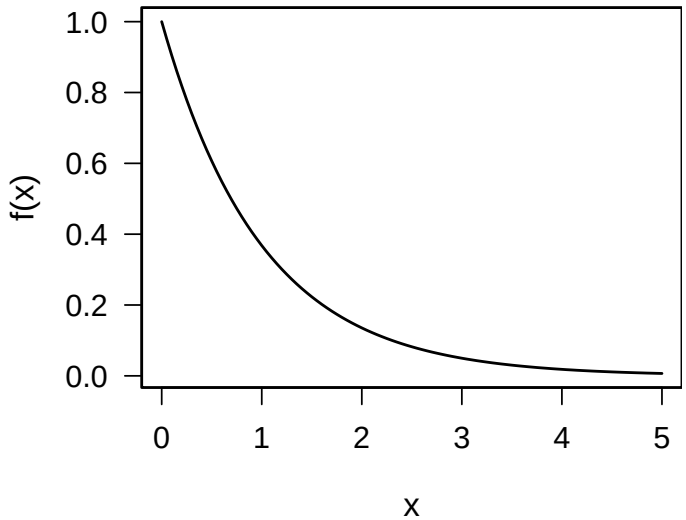
Simulation algorithm:

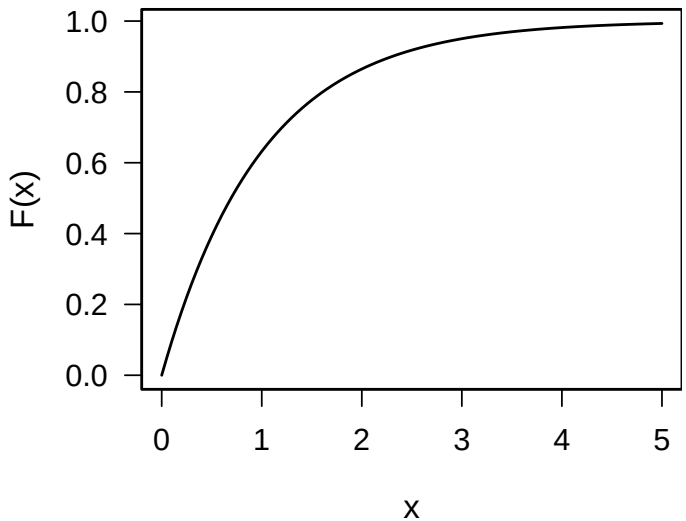
$u \sim U[0, 1]$

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

return x

Example - Exponential Distribution





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

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

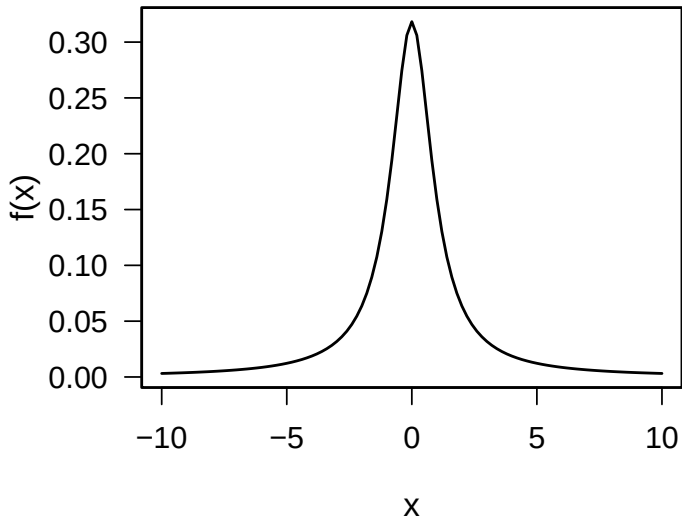
Simulation algorithm:

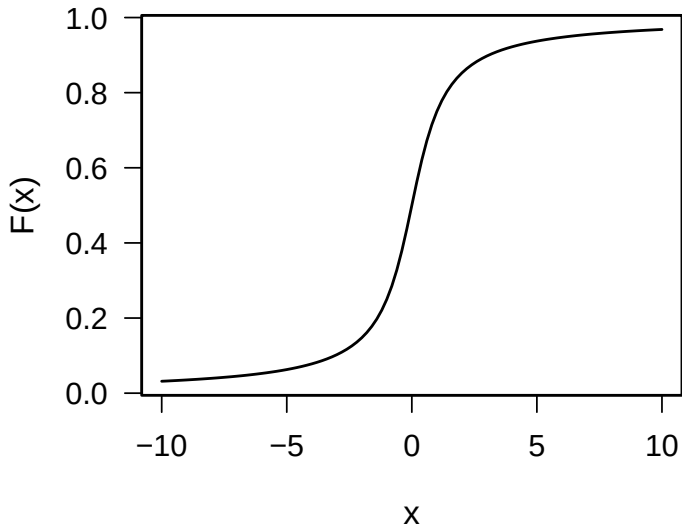
$u \sim U[0, 1]$

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

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:

$u \sim U[0,1]$

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

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

Temporary page!

LaTeX was unable to guess the total number of pages correctly. A
there was some unprocessed data that should have been added t
the final page this extra page has been added to receive it.
If you rerun the document (without altering it) this surplus page
will go away, because LaTeX now knows how many pages to expe
for this document.