

# Introduction

## Computer Arithmetics

732A90

Computational Statistics

Krzysztof Bartoszek  
([krzysztof.bartoszek@liu.se](mailto:krzysztof.bartoszek@liu.se))

08 XI 2022 (E326)

Department of Computer and Information Science  
Linköping University

# Computational Statistics — In brief

- Even simple data analysis (mean, variance) by hand is tedious
- Today: huge datasets, models capturing system complexity, interactions (between variables **and** observations)
- We will discuss:
  - Being careful with calculations—overflow
  - Generation of random variables including correlated ones
  - Numerically optimizing functions, esp. maximum likelihood
  - Computing confidence (credible) intervals for distributions when analytical ones are unobtainable

# Lesson structure

- Lectures
  - additionally a guest Zoom lecture 9 XII 2022 Ethics/Alumni
  - guest lecture for the Stats and ML programme
- Computer Labs
- Seminars
- Examination: Reports, seminars, final exam
- Final exam: computer based
- Answer in English.
- Electronic reports as **.PDF**.
- Disclose **ALL** collaborations and sources.
- Provide source code (if used).
- E-mail contact: [krzysztof.bartoszek@liu.se](mailto:krzysztof.bartoszek@liu.se)

# Course materials, software

- <https://www.ida.liu.se/~732A90/index.en.shtml> messages, **exam information** (materials incl. 2016 (732A38), course reading)
- <https://github.com/STIMALiU/CompStat>
- LISAM submission, materials
- James E. Gentle “Computational Statistics”, Springer, 2009
- Geof H. Givens, Jennifer A. Hoeting “Computational Statistics”, Wiley, 2013
- Various suggested www pages or articles
- **Googling**
  
- R

# Course contents

- Recap: R
- Recap: Basic Statistics
- Computer Arithmetics (JG pages 85–105)
- Optimization (JG pages 241–272, handouts)
- Random Number Generation (JG pages 305–312, 325–328, handouts)
- Monte Carlo Methods (JG pages 312–318, 328 417–429, handouts)
- Numerical Model Selection and Hypothesis Testing (JG pages 52–56, 424, 435–467, handouts)
- Expectation Maximization Algorithm and Stochastic Optimization (JG pages 275–284, 296–298, 480–483, handout)

Pages are recommended reading for each lecture, **NOT** exact lecture content. The lectures will build up on this material.

# Examination

Computer labs (need to be passed)

Presentation or opposition and attendance at seminars (see [732A90\\_ComputationalStatisticsHT2022\\_GroupSeminars.pdf](#)).

Computer exam points

A:  $[18, \infty)$ , B:  $[16, 18)$ , C:  $[14, 16)$ , D:  $[12, 14)$ , E:  $[10, 12)$ , F:  $[0, 10)$

Allowed aids for exam: printed books and own PDF document containing max 100 pages (see [732A90\\_ComputationalStatisticsHT2022\\_CourseInformation.pdf](#)).

# Teaching staff for course

## **Me: Krzysztof Bartoszek, background**

- ➊ MEng in Computer Science, Gdańsk Univ. of Technology 2007
- ➋ MPhil in Computational Biology, Univ. of Cambridge 2008
- ➌ PhD in Statistics, Univ. of Gothenburg 2013
- ➍ Postdoc, Dept. Mathematics Uppsala Univ. 2013–2017
- ➎ Lecturer, STIMA LiU 2017–

## **Bayu Brahmantio, Jaskirat Marar, Shashi Nagarajan**

- ➊ Labs
- ➋ Grading
- ➌ Support

# Evaluation of 2021HT course

- ① about 54 students took the course in 2021VT.
- ② 15 submitted an evaluation.
- ③ Course grade:  $2.40 \pm 1.24$ .
- ④ Changes for 2022HT.
  - ① Two additional mathematically oriented initial lectures
  - ② Updating of lecture and lab material.
  - ③ More clear deadlines for grading.
  - ④ Rules for group work.
  - ⑤ Part of seminars as lab support.



## SHOULD YOU CARE?

For a detailed mathematical treatment see Ch. 4.2 in  
D. E. Knuth (1998). The Art of Computer Programming, Volume 2, Addison–Wesley.

# Computer Arithmetics: Examples

Computations can be affected by magnitudes of numbers.

$x \leftarrow -0.5^{10000}; y \leftarrow -0.4^{10000}; x/(x+y) + y/(x+y)$

$x \leftarrow -0.5^{1000}; y \leftarrow -0.4^{1000}; x/(x+y) + y/(x+y)$

$x \leftarrow -0.1^{1000}; y \leftarrow -0.2^{1000}; x/(x+y) + y/(x+y)$

# Computer Arithmetics: Examples

Computations can be affected by magnitudes of numbers.

```
x<-0.5^10000;y<-0.4^10000;x/(x+y)+y/(x+y)
```

```
x<-0.5^1000;y<-0.4^1000;x/(x+y)+y/(x+y)
```

```
x<-0.1^1000;y<-0.2^1000;x/(x+y)+y/(x+y)
```

```
t<-rnorm(5,10^18,1);t[3]-t[4];t[1]-t[2]
```

# Computer Arithmetics: Examples

Computations can be affected by magnitudes of numbers.

```
x<-0.5^10000;y<-0.4^10000;x/(x+y)+y/(x+y)
```

```
x<-0.5^1000;y<-0.4^1000;x/(x+y)+y/(x+y)
```

```
x<-0.1^1000;y<-0.2^1000;x/(x+y)+y/(x+y)
```

```
t<-rnorm(5,10^18,1);t[3]-t[4];t[1]-t[2]
```

```
x<-10^800;sd<-10^400;y<-x/sd;y
```

# Computer Arithmetics: Examples

Computations can be affected by magnitudes of numbers.

```
x<-0.5^10000;y<-0.4^10000;x/(x+y)+y/(x+y)
```

```
x<-0.5^1000;y<-0.4^1000;x/(x+y)+y/(x+y)
```

```
x<-0.1^1000;y<-0.2^1000;x/(x+y)+y/(x+y)
```

```
t<-rnorm(5,10^18,1);t[3]-t[4];t[1]-t[2]
```

```
x<-10^800;sd<-10^400;y<-x/sd;y
```

And your are doing estimation under a nice, fancy model ...

# Data presentation

- Computers store information in binary form  
0 1 0 1 1 0 0 1
- 1Byte=8bits (typical counting unit)
- 1Word=32 or 64bits (depending on architecture)
- 1KB=1024bytes
- 1MB=1024KB
- and so on

# Data presentation

- Computers store information in binary form  
0 1 0 1 1 0 0 1
- 1Byte=8bits (typical counting unit)
- 1Word=32 or 64bits (depending on architecture)
- 1KB=1024bytes
- 1MB=1024KB
- and so on

**QUESTION:** Why binary form?

# Character encoding

- ASCII (American Standard Code for Information Interchange)
- Unicode
- `read.csv()`, `read.table()` have `fileEncoding` argument



# Fixed-point system (integers)

- We use the base-10 (decimal) system, e.g.  
 $1234 = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$
- Computers: base-2 (binary) system
- Each integer represented as:  
 $A = a_0 \cdot 2^0 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + \dots$

**EXERCISE:** 5, 16, 17, 31, 32, 33, 255, 256 in binary

# Fixed-point system (integers)

- We use the base-10 (decimal) system, e.g.

$$1234 = 1 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 4 \cdot 10^0$$

- Computers: base-2 (binary) system
- Each integer represented as:

$$A = a_0 \cdot 2^0 + a_1 \cdot 2^1 + a_2 \cdot 2^2 + \dots$$

**EXERCISE:** 5, 16, 17, 31, 32, 33, 255, 256 in binary

- Negative numbers

# Arithmetic operations

R operations on binary

- Addition, multiplication: base-2 instead of base-10
- Subtraction:  $A - B = A + (-B)$  (*twos-complement*)
- Division: tedious, rounded towards 0 `as.integer(17/3)`

# Arithmetic operations

R operations on binary

- Addition, multiplication: base-2 instead of base-10
- Subtraction:  $A - B = A + (-B)$  (*twos-complement*)
- Division: tedious, rounded towards 0 `as.integer(17/3)`
  
- **Overflow:** adding two large numbers the sign bit can be treated as a high order bit and on some architectures results in a negative number

# Floating-point system (rational, “real”)

- How can we represent fractions (rational numbers)?

# Floating-point system (rational, “real”)

- How can we represent fractions (rational numbers)?
- Sign
- Exponent (**signed**, read standards if interested)
- Mantissa or Significand
- on 64bits:

sign (1bit)	Exponent (11bits)	Mantissa (52 bits)
----------------	----------------------	-----------------------

$$\pm 0.d_1 d_2 \dots d_p \cdot b^e \quad b = 2, \quad p = 52$$

- **Range:**  $\approx [-10^{300}, 10^{300}] \approx [-b^{e_{max}}, b^{e_{max}}]$

# Floating-point system

- Rationals rounded towards the nearest computer float

**options**( digits=22) *#max possible*

0.1

[1] 0.100000000000000000055511

- **EXAMPLE:** Assume base  $b = 10$  and mantissa has 5 digits  $p = 5$ :

$$1.2345 = +0.12345 \cdot 10^1$$

$$4.0000567 = +0.40000 \cdot 10^1$$

- Problem remains whatever base ( $b$ ) is chosen

- **EXERCISE:** Try to convert some numbers

# Floating-point system

- Distribution of computer floats



- Dense from  $-1$  to  $1$
- Density decreases
- same number of points for each exponent:

$$\dots, \cdot 10^{-3}, \cdot 10^{-2}, \cdot 10^{-1}, \cdot 10^1, \cdot 10^2, \cdot 10^3, \dots$$

- What about integers?

$$5 = +0.50000 \cdot 10^1$$

**options**( digits=22)

9007199254740992

9007199254740993

9007199254740994



## Floating-point system, special “numbers”

- $\pm\text{Inf}$ : exponent is  $\text{exp}_{\max} + 1$ , mantissa is 0
- NaN: exponent is  $\text{exp}_{\max} + 1$ , mantissa is  $\neq 0$
- 0 **WHY?**

# Floating-point system, special “numbers”

- $\pm\text{Inf}$ : exponent is  $\text{exp}_{\text{max}} + 1$ , mantissa is 0
- NaN: exponent is  $\text{exp}_{\text{max}} + 1$ , mantissa is  $\neq 0$
- 0 **WHY?**

**Overflow:** number larger than can be represented

**Underflow:** loss of significant digits

$$10^{200} * 10^{200} = \text{Inf}$$

$$10^{400} / 10^{400} = \text{NaN}$$

$$10^{-200} / 10^{200} = 0$$

$$10^{-200} * 10^{200} =$$

$$0 * 10^{400} =$$

$$x < -10^{300}; \text{while}(1) \{ x < -x + 1 \}$$

# Arithmetic operations

- Floats are rounded so usual mathematical laws do not hold — floating point arithmetic

- **Examples**

$$1/3+1/3 = 0.6666667$$

**options**( digits=22)

$$1/3+1/3 = 0.66666666666666666666296592$$

$$10^{(-200)} / (10^{(-200)} + 10^{(-200)}) = 0.5$$

$$10^{(-200)} / (10^{(-200)} + 20^{(-200)}) = 1$$

- Software is designed to make operations as correct as possible

# Arithmetic operations

- Floats are rounded so usual mathematical laws do not hold — floating point arithmetic

- **Examples**

$$1/3+1/3 = 0.6666667$$

**options**( digits=22)

$$1/3+1/3 = 0.66666666666666666666296592$$

$$10^{(-200)} / (10^{(-200)} + 10^{(-200)}) = 0.5$$

$$10^{(-200)} / (10^{(-200)} + 20^{(-200)}) = 1$$

- Software is designed to make operations as correct as possible
- Do we need to work with such extreme numbers?

# Arithmetic operations

- $X + Y$ ,  $X \cdot Y$  can display overflow, underflow
- $A \neq B$  but  $X + A = B + X$
- $A + X = X$  but  $A + Y \neq Y$
- $A + X = X$  but  $X - X \neq A$

# Arithmetic operations

- $X + Y$ ,  $X \cdot Y$  can display overflow, underflow
- $A \neq B$  but  $X + A = B + X$
- $A + X = X$  but  $A + Y \neq Y$
- $A + X = X$  but  $X - X \neq A$
- **COMPARING FLOATS IS TRICKY!**

```
options(digits=22)
x<-sqrt(2)
x*x
[1] 2.00000000000000000444089
(x*x)==2
[1] FALSE
isTRUE(all.equal(x*x,2))
[1] TRUE
```

# Summation

Underflow problems can occur with any summation ( $x \leftarrow x+1$ )

```
options( digits=22)  
x<-1:1000000; sum(1/x); sum(1/rev(x))  
[1] 14.39272672286572252176  
[1] 14.39272672286572429812
```

- WHICH ONE IS CORRECT?

# Summation

Underflow problems can occur with any summation ( $x \leftarrow -x+1$ )

```
options( digits=22)  
x<-1:1000000; sum(1/x); sum(1/rev(x))  
[1] 14.39272672286572252176  
[1] 14.39272672286572429812
```

- WHICH ONE IS CORRECT?
- WHICH ONE IS MORE ACCURATE?



# Potential solutions

Solution A:

- ① Sort the numbers ascending **CAN BE EXPENSIVE**
- ② Sum in this order

# Potential solutions

Solution A:

- 1 Sort the numbers ascending **CAN BE EXPENSIVE**
- 2 Sum in this order

Solution B:

- 1 Sum numbers pairwise, from  $n$  obtain  $n/2$  numbers  
**HOW TO CHOOSE PAIRS?**
- 2 Continue until 1 number left

## Example

- Computing exponent using Taylor series

$$e^x = 1 + x + x^2/2 + x^3/6 + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

# The exponential

```
options( digits=22)
fTaylor<-function(x,N){1+sum(sapply(1:N,
  function(i ,x){x^i/prod(1:i )},x=x, simplify=
  TRUE))}
exp(20) #fine
[1] 485165195.4097902774811
fTaylor(20,100)
[1] 485165195.4097902774811
fTaylor(20,100)-exp(20)
[1] 0
exp(-20) #problem
[1] 2.061153622438557869942e-09
fTaylor(-20,100)
[1] -3.853877217352419393137e-10
fTaylor(-20,200)
[1] -3.853877217352419393137e-10
```

## Example

- Computing exponent using Taylor series

$$e^x = 1 + x + x^2/2 + x^3/6 + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

- WHY?

## Example

- Computing exponent using Taylor series

$$e^x = 1 + x + x^2/2 + x^3/6 + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

- **WHY?**
- Varying sign of terms
- **CANCELLATION** adding two numbers of almost equal magnitude but of opposite sign
- Effects of cancellations accumulate
- **SOLUTION:** Different algorithm ...

## Can you explain why?

```
## Example due to Thomas Ericsson in his  
Numerical Analysis course at Chalmers  
f1<-function(x){(x-1)^6}  
f2<-function(x){1-6*x+15*x^2-20*x^3+15*x^4-6*x  
^5+x^6}  
  
x<-seq(from=0.995,to=1.005,by=0.0001)  
y1<-f1(x);y2<-f2(x)  
  
plot(x,y1,pch=19,cex=0.5,ylim=c(-5*10^(-15),20  
*10^(-15)),main="Two ways to calculate (x  
-1)^6",xlab="x",ylab="y")  
points(x,y2,pch=18,cex=0.8)
```

# Matrix Calculations

- Many problems (in Statistics, Numerical methods, e.t.c) can be reduced to solving

$$\mathbf{A}\vec{x} = \vec{b}$$

$\mathbf{A}$  (design) matrix

$\vec{x}$  vector of unknowns

$\vec{b}$  vector of scalars (data)

- Algorithm should be **numerically stable**

(small changes in  $\mathbf{A}$  or  $\vec{b}$  imply in small changes in  $\vec{x}$ )



# Example: Linear regression models

Minimize

$$RSS(\beta_0, \beta_1, \dots, \beta_m) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_{i1} - \dots - \beta_m x_{im})^2$$

The system of equations

$$\frac{\partial RSS}{\partial \beta_0} = \frac{\partial RSS}{\partial \beta_1} = \dots = \frac{\partial RSS}{\partial \beta_m} = 0$$

can be written as

$$\mathbf{X}^T \mathbf{X} \vec{\beta} = \mathbf{X} \vec{y}$$

where

$$\mathbf{X} = \begin{bmatrix} 1 & x_{11} & \dots & x_{1m} \\ 1 & x_{21} & \dots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \dots & x_{nm} \end{bmatrix}$$

# Solving a linear system

- $\mathbf{A}\vec{x} = \vec{b}$  needs a stable numerical solution
- **Computer arithmetics**
  - Original system:  $\mathbf{A}\vec{x} = \vec{b}$
  - Perturbed system:  $\mathbf{A}\vec{x}' = \vec{b}', \vec{x}' = \vec{x} + \delta\vec{x}, \vec{b}' = \vec{b} + \delta\vec{b}$
- Stable: small perturbation in  $\vec{b}$ , small perturbations in  $\vec{x}$

# Solving a linear system

- $\mathbf{A}\vec{x} = \vec{b}$  needs a stable numerical solution
- **Computer arithmetics**
- - Original system:  $\mathbf{A}\vec{x} = \vec{b}$
  - Perturbed system:  $\mathbf{A}\vec{x}' = \vec{b}', \vec{x}' = \vec{x} + \delta\vec{x}, \vec{b}' = \vec{b} + \delta\vec{b}$
- Stable: small perturbation in  $\vec{b}$ , small perturbations in  $\vec{x}$
- 

$$\frac{\|\delta\vec{x}\|}{\|\vec{x}\|} \leq \|\mathbf{A}^{-1}\| \|\mathbf{A}\| \frac{\|\delta\vec{b}\|}{\|\vec{b}\|}$$

# Solving a linear system

Condition number of a matrix

$$\kappa(\mathbf{A}) = \|\mathbf{A}^{-1}\| \|\mathbf{A}\|$$

- Large  $\kappa(\mathbf{A})$  is a bad sign, but does not imply ill-conditioning

# Solving a linear system

Dealing with ill-conditioning

# Solving a linear system

Dealing with ill-conditioning

- Rescale the variables (columns)
- Use a different algorithm for solving  
e.g. QR, Cholesky, SVD

**Cholesky:**  $\mathbf{A}$  symmetric-positive-definite  
 $\mathbf{A}\vec{x} = \vec{b}$  is equivalent to  $\mathbf{L}\mathbf{L}^T\vec{x} = \vec{b}$  **WHY?**

# Solving a linear system

Dealing with ill-conditioning

- Rescale the variables (columns)
- Use a different algorithm for solving  
e.g. QR, Cholesky, SVD

**Cholesky:**  $\mathbf{A}$  symmetric-positive-definite  
 $\mathbf{A}\vec{x} = \vec{b}$  is equivalent to  $\mathbf{L}\mathbf{L}^T\vec{x} = \vec{b}$  **WHY?**

- 1 Solve  $\mathbf{L}\vec{y} = \vec{b}$
- 2 Solve  $\mathbf{L}^T\vec{x} = \vec{y}$

- Computations can behave “differently” at different numerical ranges.
- Floating point system.
- Computer arithmetics is not the same as “usual” arithmetic.
- Summing series, solving linear systems (inversion?)