



University
of Manitoba

MATH 8xyz – Lecture 01

General information about the course & (brief) Introduction to R and julia

Julien Arino

Department of Mathematics @ University of Manitoba

Maud Menten Institute @ PIMS

julien.arino@umanitoba.ca

Winter 20XX

The University of Manitoba campuses are located on original lands of Anishinaabeg, Ininew, Anisininew, Dakota and Dene peoples, and on the National Homeland of the Red River Métis.

We respect the Treaties that were made on these territories, we acknowledge the harms and mistakes of the past, and we dedicate ourselves to move forward in partnership with Indigenous communities in a spirit of Reconciliation and collaboration.

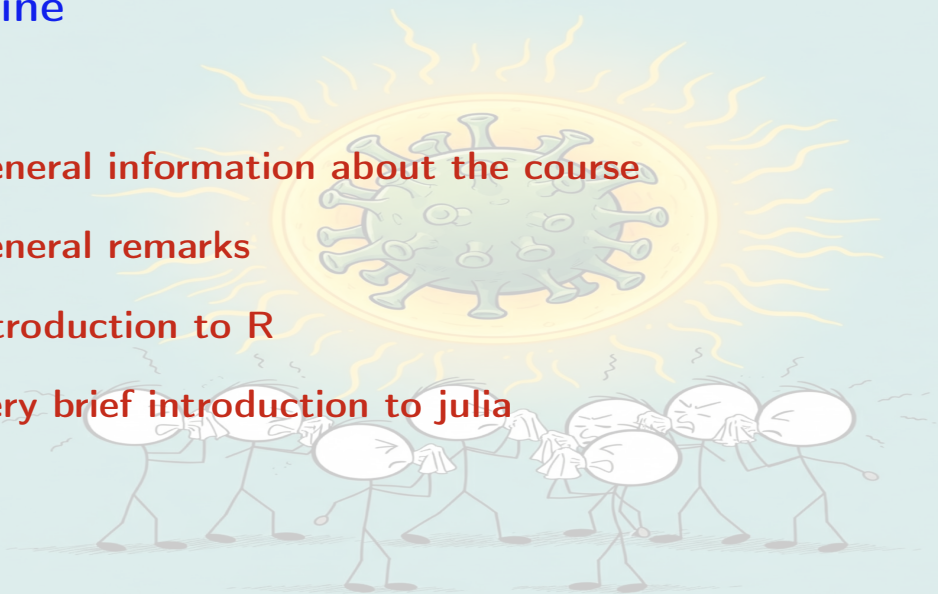
Outline

General information about the course

General remarks

Introduction to R

Very brief introduction to julia

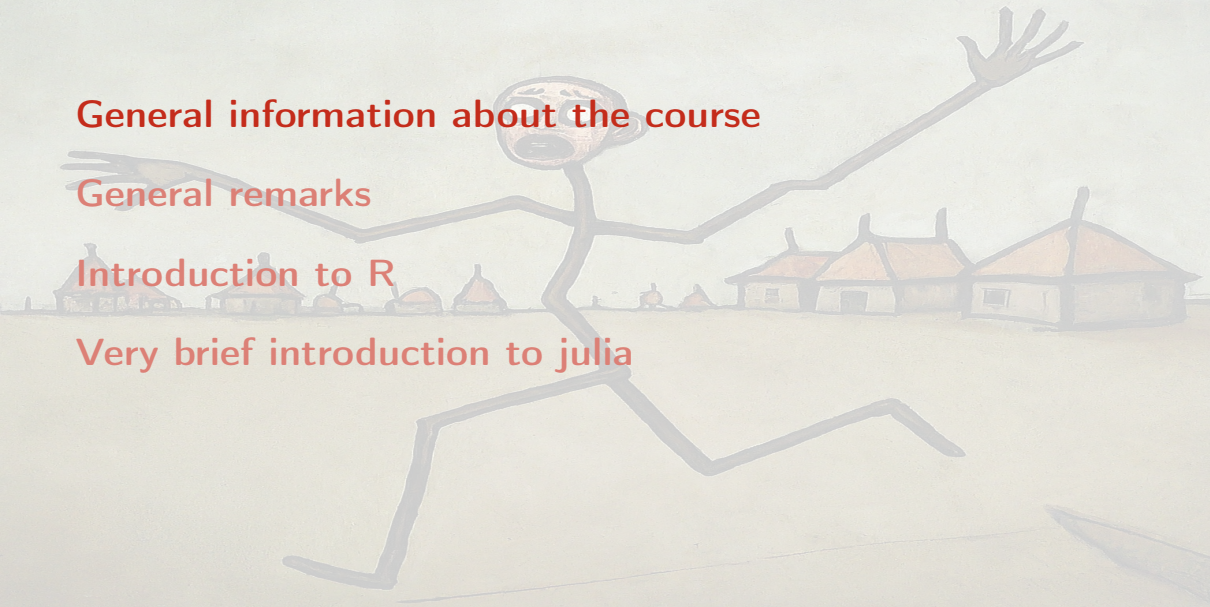


General information about the course

General remarks

Introduction to R

Very brief introduction to julia



Why this course?

► I may have sent you a link to this course because you just started an MSc or PhD with me. This course will serve as the base to a reading/topics course on mathematical epidemiology (*math epi*) that we will do together

(Hence the name: this will be an 8000 level course whose number will be decided when the course is calendared)

► I teach math epi in Summer Schools and short courses, for which I prepare slides (and sometimes videos). However, this is a more complete version, so that students in these events can dig into the topics more

► More generally, perhaps you want to learn about math epi

Who is this course for?

- ▶ Students in mathematics, physics or engineering
- ▶ I am assuming some working knowledge of ordinary differential equations (ODE) and a few related topics. For instance, I am not going into details and will assume that you know the link between local asymptotic stability of an equilibrium of an ODE and the location of the eigenvalues of the Jacobian of the ODE at that equilibrium
- ▶ This could also be interesting to students in less mathematically heavy areas... but will need some ground work to understand some concepts. (I plan a course for non specialists, but it is not finalised yet)

Getting in touch

Email me (email is on all lecture title pages)

Use a tag like [MathEpi8xyz] in your subject line, I get a lot of emails

If I don't reply, don't hesitate to try again... I am bad with emails

GitHub repository for the course

Most course material is available from this GitHub repository:

<https://julien-arino.github.io/math8xyz-reading-course-math-epi>

This repo includes the slides, code and data samples

This **does not** include pdf of the bibliographic references, although there are links to articles and books and, where needed, bibliographic information at the end of a slide set. As much as possible, I link to Open Access sources

One remark: I sometimes refer to Wikipedia. For the younger students here: this can be where you first look, not what you cite in proper work

Code

I use R and (less often) julia –Python would be a good choice as well but I prefer R and julia

Instructions on setting up R and julia for the course can be found in the GitHub repo

Some code is in the repo

For the epi side of things, a very useful open reference: R for applied epidemiology and public health

Slides

Slides are written in \LaTeX and Beamer and rendered using pdf \LaTeX (for the pdf slides) or \LaTeX and `knitr` and rendered R (typically in RStudio)

See the SLIDES folder on GitHub for instructions on how to compile

As much as possible, I have indicated provenance (by linking the file on the original website); when not possible, the file is saved with the name of the source indicated, or the source `tex` or `md` files include the link commented

Videos

Videos are being recorded for the course

They will be/are posted on YouTube

You can access the playlist [here](#). Links to individual videos are also available on the webpage

Course objectives

Introduction to Mathematical Epidemiology

- ▶ Problems
- ▶ Methods

We will have these particular problems in mind:

1. Modelling techniques
2. Mathematical analysis of models
3. Computational analysis of models
4. Use of data

It is important to do the 4 interactively

Course project (if taking this course for credit)

If you are at the University of Manitoba and are taking this course for credit:

- ▶ The mark for the course will be decided based on a project you will have to return
- ▶ The project will be quite involved

Content of the course – Lecture-oriented

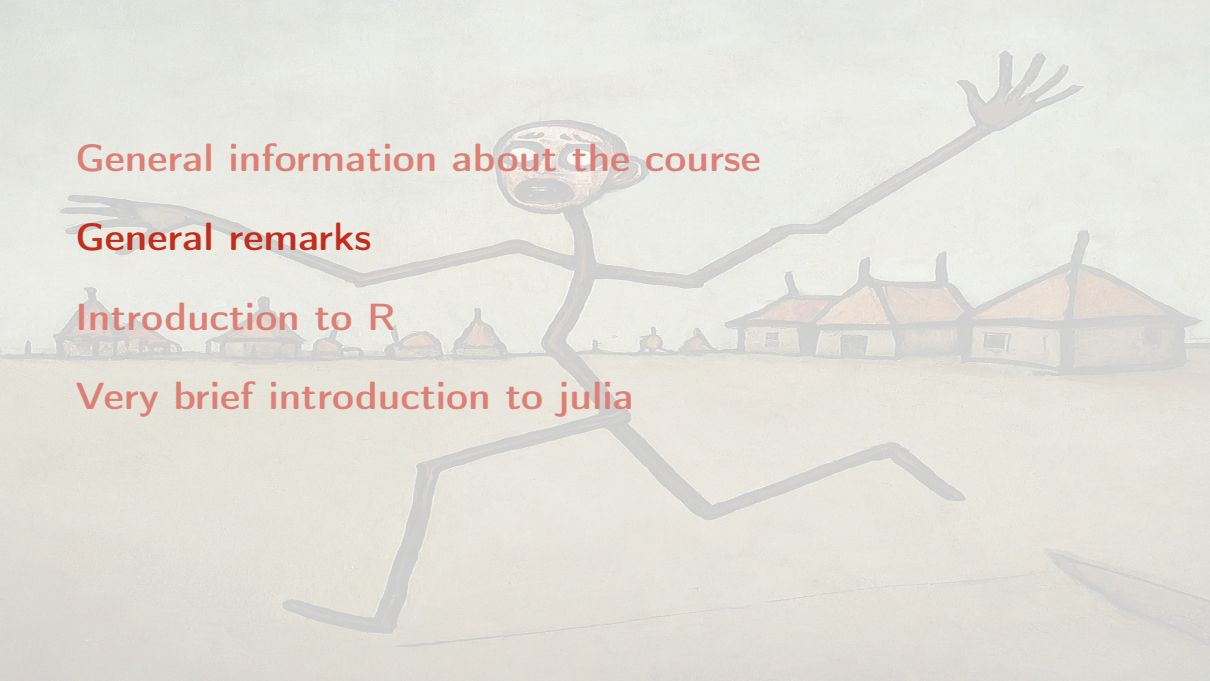
The slides for the course are organised by lecture

Each lecture is intended to be roughly 1 hour and 15 minutes (as a course taught Tuesday and Thursday at UM)

There are 26 sets of slides, each corresponding to a lecture

Some content will continue across several slide sets; this will be explicit from the slide set title

Ultimately, there will also be 26 videos



General information about the course

General remarks

Introduction to R

Very brief introduction to julia

About modelling

- ▶ Do not neglect this *crucial* step
- ▶ Think outside the box
- ▶ Take the time
- ▶ Try to remain “as simple as possible without being too simple”
- ▶ Do not hesitate to modify your model, even when you have already done work on it: if something doesn’t “smell right”, your model might need fixing

About mathematical analysis

- ▶ Used to be the sole purpose of most papers
- ▶ Important to carry out to understand the basic properties of models
- ▶ Judge your audience: global asymptotic stability is cool, but is it really required if you want to present work to a public health person?
- ▶ Do as much as possible (for instance, knowing the value of \mathcal{R}_0 can be useful to set parameters), do not hesitate to move to an appendix

About numerics

- ▶ Numerics should be used to **complement** the mathematical analysis
- ▶ If you have shown the global stability of some equilibrium point, **do not** show a simulation where solutions converge to this equilibrium
- ▶ In fact, it is rarely useful to show a solution (cases where it is okay: before going to zero the number of infectious does something really cool, you have a period doubling, etc.)
- ▶ Instead, use numerics to investigate scenarios or test the effect of varying parameters
- ▶ A good figure tells a story, it is worth spending time thinking about how to make good figures

About data

- ▶ Acquiring data has become much easier than even 20 years ago
- ▶ As a modeller, it is not necessary that everything be data-driven, but it is necessary to be "context-aware" (this means getting a sense of the quantities involved in the process you want to model)
- ▶ Fred Brauer (1932-2021): "If your model and the data disagree, question the quality of your data"

Reading recommendations

- ▶ Waltman. Deterministic threshold models in the theory of epidemics (1974)
- ▶ Capasso. Mathematical structures of epidemic systems (1993)
- ▶ Hethcote. The mathematics of infectious diseases. SIAM Review (2000)
- ▶ Daley & Gani. Epidemic modelling (2001)
- ▶ Brauer, PvdD & Wu. Mathematical Epidemiology (2008)
- ▶ Brauer & C³. Mathematical Models in Population Biology and Epidemiology (2012)
- ▶ Brauer, C³ & Feng. Mathematical Models in Epidemiology (2019)

Word of warning!

The references on the previous slide are **my** favourite references

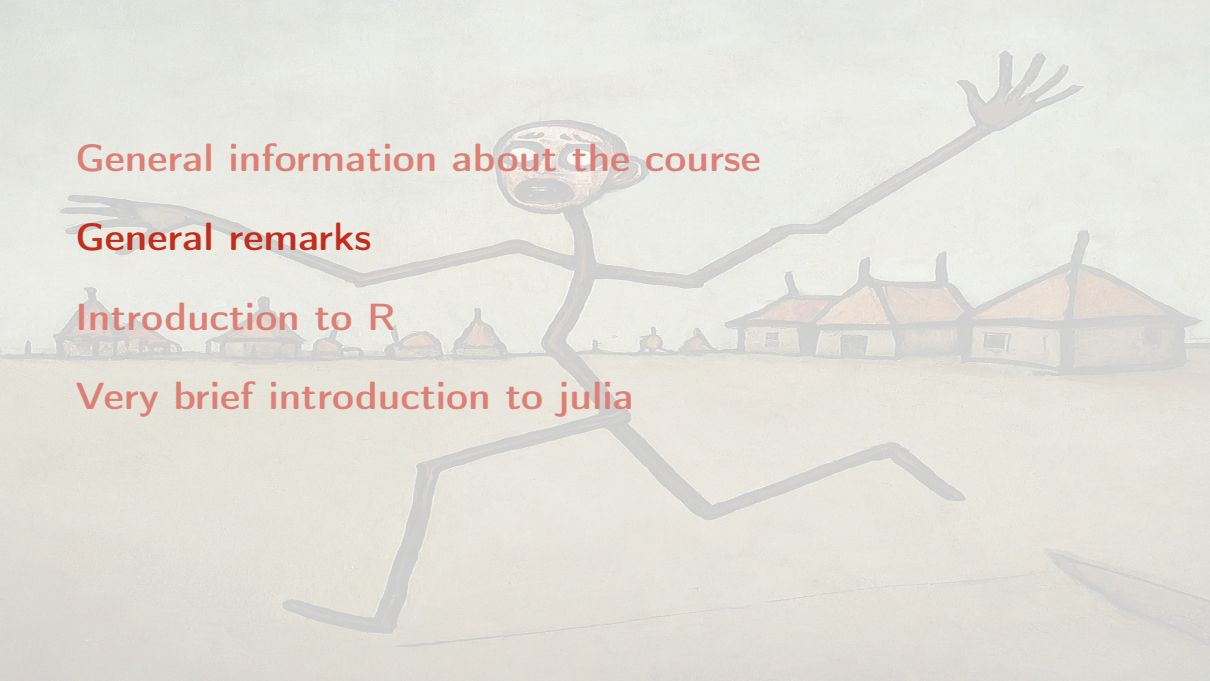
This extends to the entire content of this course: the topics I treat and the way I treat them originate in **my** personal itinerary through math epi and as a consequence

- ▶ Some stuff might be understood (and therefore explained) differently by someone else
- ▶ I focus on some topics that I think are cool and omit others that I like less
- ▶ This doesn't mean these topics are less worthy of consideration

⇒ Don't trust me blindly and "shop around": other colleagues have courses that are also really worth your time

Some common abbreviations

- EP: equilibrium point
- DFE: disease-free equilibrium
- EEP: endemic equilibrium point
- LAS: locally asymptotically stable / local asymptotic stability
- GAS: globally asymptotically stable / global asymptotic stability



General information about the course

General remarks

Introduction to R

Very brief introduction to julia

Most of the programming in this course is done using the R language

Code that you see in these slides is executable (and executed to create the slides)

Each slide set with code (most of them) also generates a corresponding R file in the CODE directory of the repo. This file strips all the \LaTeX formatting and just keeps the R code

R was originally for stats but is now more

- ▶ Open source version of S
- ▶ Appeared in 1993
- ▶ Now version 4.2
- ▶ One major advantage in my view: uses a lot of C and Fortran code. E.g., deSolve:
The functions provide an interface to the FORTRAN functions lsoda, lsodar, lsode, lsodes of the ODEPACK collection, to the FORTRAN functions dvide, zvode and daspk and a C-implementation of solvers of the Runge-Kutta family with fixed or variable time steps
- ▶ Very active community on the web, easy to find solutions (same true of Python, I just prefer R)

Development environments

- ▶ Terminal version, not very friendly
- ▶ Nicer terminal: `radian`
- ▶ Execute R scripts by using `'Rscript name__of__script.R'`. Useful to run code in `'cron'`, for instance
- ▶ Use IDEs:
 - ▶ RStudio has become the reference
 - ▶ RKWard is useful if you are for instance using an ARM processor (Raspberry Pi, some Chromebooks..)
- ▶ Integrate into jupyter notebooks

Going further

- ▶ RStudio server: run RStudio on a Linux server and connect via a web interface
- ▶ Shiny: easily create an interactive web site running R code
- ▶ Shiny server: run Shiny apps on a Linux server
- ▶ Rmarkdown: markdown that incorporates R commands. Useful for generating reports in html or pdf, can make slides as well..
- ▶ RSweave: LaTeX incorporating R commands. Useful for generating reports. Not used as much as Rmarkdown these days

R is a scripted language

- ▶ Interactive
- ▶ Allows you to work in real time
 - ▶ Be careful: what is in memory might involve steps not written down in a script
 - ▶ If you want to reproduce your steps, it is good to write all the steps down in a script and to test from time to time running using 'Rscript': this will ensure that all that is required to run is indeed loaded to memory when it needs to, i.e., that it is not already there..

Introduction to R

Crash course on programming in R

Dealing with data in R

Solving ODE with R



R is similar to matlab..

.. with some differences, of course! Otherwise, where would the fun be? ;)

Assignment

Two ways:

```
X <- 10
```

```
X = 10
```

First version is preferred by R purists.. I don't really care (but be ready for an argument if you use = in a discussion with an R purist)

Lists

A very useful data structure, quite flexible and versatile. Empty list: `L <- list()`. Convenient for things like parameters. For instance

```
L <- list()
L$a <- 10
L$b <- 3
L[["another_name"]] <- "Plouf plouf"
```

Could also set some or all entries right away

```
L <- list(a = 10, b = 3, another_name = "Plouf plouf")
```

Accessing list entries

```
L[1]
```

```
## $a
```

```
## [1] 10
```

```
L[[2]]
```

```
## [1] 3
```

```
L$a
```

```
## [1] 10
```

```
L[["b"]]
```

```
## [1] 3
```


Vectors

```
x = 1:10
y <- c(x, 12) # Append 12 to x
y

## [1] 1 2 3 4 5 6 7 8 9 10 12

z = c("red", "blue")
z = c(z, 1) # Append 1 to z
z

## [1] "red" "blue" "1"
```

In `z`, since the first two entries are characters, the added entry is also a character. Contrary to lists, vectors have all entries of the same type

Matrices

```
A <- mat.or.vec(nr = 2, nc = 3)
B <- matrix(c(1,2,3,4), nr = 2, nc = 2)
```

B

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    2    4
```

```
C <- matrix(c(1,2,3,4), nr = 2, nc = 2,
             byrow = TRUE)
```

C

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
```

Naming the arguments (e.g., `nr = 2`) allows to use arguments in any order

Matrix operations – Be careful!

Probably the biggest annoyance in R compared to other languages

- ▶ The notation $A*B$ is the *Hadamard product* $A \circ B$ (what is denoted $A.*B$ in most other languages), not the standard matrix multiplication
- ▶ Standard matrix multiplication is written $A \%*\% B$

Vector operations

Vector addition is also frustrating. Say you write `x=1:10`, i.e., make the vector

```
x  
## [1] 1 2 3 4 5 6 7 8 9 10
```

Then `x+1` should give an error but instead gives

```
x+1  
## [1] 2 3 4 5 6 7 8 9 10 11
```

i.e., adds a vector of all ones to the vector

Beware of this in particular when addressing sets of indices in lists, vectors or matrices

For the matlab-ers here

- ▶ R does not have the keyword `end` to access the last entry in a matrix/vector/list..
- ▶ Use `length` (lists or vectors), `nchar` (character chains), `dim` (matrices.. careful, of course returns 2 values), `nrow`, `ncol`...

Data frames

- ▶ Specific to R, although they are now present in many other languages
- ▶ Like matrices under steroids
- ▶ Like lists, can contain entries of different types, e.g., a column with numbers and a column with characters

Data frames

```
A = data.frame(column_1 = runif(9),  
               colour = ifelse(runif(9) < 0.5,  
                              "red", "green"))
```

A

##	column_1	colour
## 1	0.5587599	red
## 2	0.3641301	red
## 3	0.7191633	green
## 4	0.2042513	green
## 5	0.6245046	red
## 6	0.3104442	red
## 7	0.4117041	green
## 8	0.5871464	green
## 9	0.4139732	red

The summary function

```
summary(A)

##      column_1      colour
##  Min.      :0.2043  Length:9
##  1st Qu.:0.3641   Class :character
##  Median :0.4140   Mode  :character
##  Mean    :0.4660
##  3rd Qu.:0.5871
##  Max.    :0.7192
```

Also works on other data types

Naming positions/rows/columns/etc

Very useful: it is possible to name entries in a list but also positions in vectors, matrices, data frames

```
v = c(alpha = 2, beta = 3, gamma = 4, delta = 5)
v

## alpha  beta gamma delta
##      2      3      4      5
```

is a vector and

```
v["beta"]

## beta
##      3
```

Sometimes you need to get rid of names

```
3 * v["beta"]
```

```
## beta
```

```
##      9
```

```
3 * as.numeric(v["beta"])
```

```
## [1] 9
```

Assign a name *a posteriori*

```
v = c(1,2,3)
names(v) = c("alpha", "beta", "gamma")
v

## alpha  beta gamma
##      1      2      3
```

```
A = matrix(c(1,2,3,4), nrow = 2, byrow = TRUE)
rownames(A) = c("alpha","beta")
colnames(A) = c("thingama","doodle")
```

```
A
```

```
##           thingama doodle
## alpha           1       2
## beta            3       4
```

```
A[1,2]
```

```
## [1] 2
```

```
A["alpha","doodle"]
```

```
## [1] 2
```

Tibbles

- ▶ New data format part of the `tidyverse` (set of libraries facilitating data wrangling)
- ▶ Quite similar to data frame, with variations
- ▶ Will be the default output if you use `tidyverse` functions

Flow control

```
if (condition is true) {  
  list of stuff to do  
}
```

Even if list of stuff to do is a single instruction, best to use curly braces

```
if (condition is true) {  
  list of stuff to do  
} else if (another condition) {  
  ...  
} else {  
  ...  
}
```

For loops

for applies to lists or vectors

```
for (i in 1:10) {  
  something using integer i  
}  
for (j in c(1,3,4)) {  
  something using integer j  
}  
for (n in c("truc", "muche", "chose")) {  
  something using string n  
}  
for (m in list("truc", "muche", "chose", 1, 2)) {  
  something using string n or integer n, depending  
}
```

lapply

Very useful function (a few others in the same spirit: `sapply`, `vapply`, `mapply`)

Applies a function to each entry in a list (/vector/matrix for `sapply` and `vapply`)

There are parallel versions (`future_lapply` & `parLapply`) \implies worth learning

lapply – Setup an example list

```
L = list()
for (i in 1:3) {
  L[[i]] = runif(i) # i=1 has 1 entry, i=2 has 2 entries, etc.
}
L

## [[1]]
## [1] 0.9145447
##
## [[2]]
## [1] 0.5518902 0.2406821
##
## [[3]]
## [1] 0.6922259 0.5982469 0.6337687
```

lapply

```
lapply(X = L, FUN = mean)
```

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

```
## [1] 0.9145447
```

```
##
```

```
## [[2]]
```

```
## [1] 0.3962861
```

```
##
```

```
## [[3]]
```

```
## [1] 0.6414138
```

To make a vector from lapply

To make a vector

```
unlist(lapply(X = L, FUN = mean))  
## [1] 0.9145447 0.3962861 0.6414138
```

or

```
sapply(X = L, FUN = mean)  
## [1] 0.9145447 0.3962861 0.6414138
```

“Advanced” lapply

Can “pick up” nontrivial list entries

```
L = list()
for (i in 1:10) {
  L[[i]] = list()
  L[[i]]$a = runif(i)
  L[[i]]$b = runif(2*i)
}
sapply(X = L, FUN = function(x) length(x$b))

## [1] 2 4 6 8 10 12 14 16 18 20
```

The argument to the function you define is a list entry (L[[1]], L[[2]], etc., here)

Avoid parameter variation loops with `expand.grid`

```
variations = list(  
  p1 = seq(1, 10, length.out = 10),  
  p2 = seq(0, 1, length.out = 10),  
  p3 = seq(-1, 1, length.out = 10)  
)  
# Create the list  
tmp = expand.grid(variations)  
PARAMS = list()  
for (i in 1:dim(tmp)[1]) {  
  PARAMS[[i]] = list()  
  for (k in 1:length(variations)) {  
    PARAMS[[i]][[names(variations)[k]]] = tmp[i, k]  
  }  
}
```

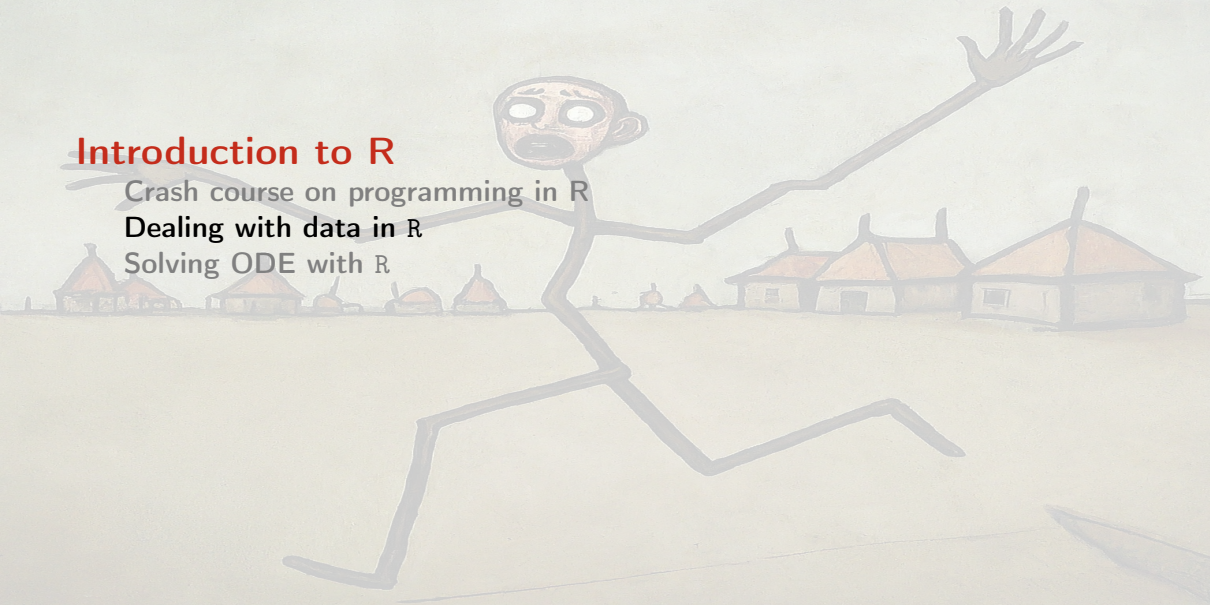
You can split this list, use it on different machines, etc. And can use `future_lapply`

Introduction to R

Crash course on programming in R

Dealing with data in R

Solving ODE with R



It is important to be “data aware”

See, e.g., JA. Mathematical epidemiology in a data-rich world. *Infectious Disease Modelling* **5**:161-188 (2020) and the GitHub repo for that paper

Using R, it is really easy to grab data from the web, e.g., from Open Data sources

- ▶ More and more locations have an open data policy
- ▶ As a modeller, you do not need to use data everywhere, but you should be *acutely* aware of the context in which your model operates
- ▶ If you want your work to have an impact, for instance in public health, you cannot be completely disconnected from reality

Data is everywhere

Closed data

- ▶ Often generated by companies, governments or research labs
- ▶ When available, come with multiple restrictions

Open data

- ▶ Often generated by the same entities but “liberated” after a certain period
- ▶ More and more frequent with governments/public entities
- ▶ Wide variety of licenses, so beware
- ▶ Wide variety of qualities, so beware

Open Data initiatives

Recent movement (5-10 years): governments (local or higher) create portals where data are centralised and published

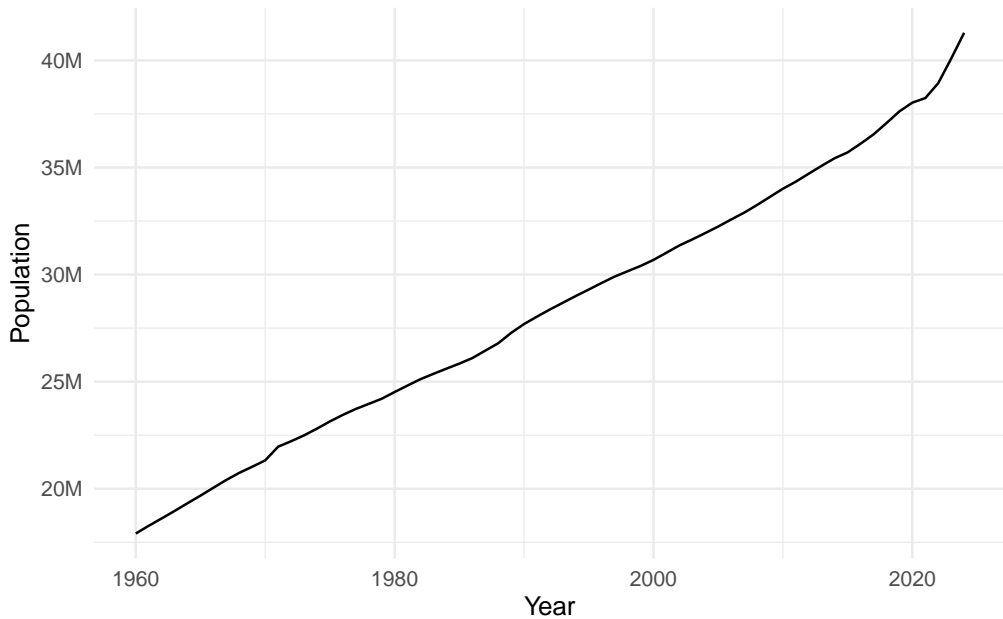
- ▶ Winnipeg
- ▶ Alberta, B.C., Ontario
- ▶ Canada
- ▶ Europe
- ▶ UN
- ▶ World Bank
- ▶ WHO

Example: population of Canada

(using libraries wbstats, ggplot2 and scales)

```
pop_data_CTRY <- wb_data(country = "CAN", indicator = "SP.POP.TOTL",  
                          mrv = 100, return_wide = FALSE)  
  
ggplot(pop_data_CTRY, aes(x = date, y = value)) +  
  geom_line() +  
  labs(title = "Population of Canada", x = "Year", y = "Population") +  
  scale_y_continuous(  
    labels = label_number(scale = 1e-6, suffix = "M")) +  
  theme_minimal()
```

Population of Canada

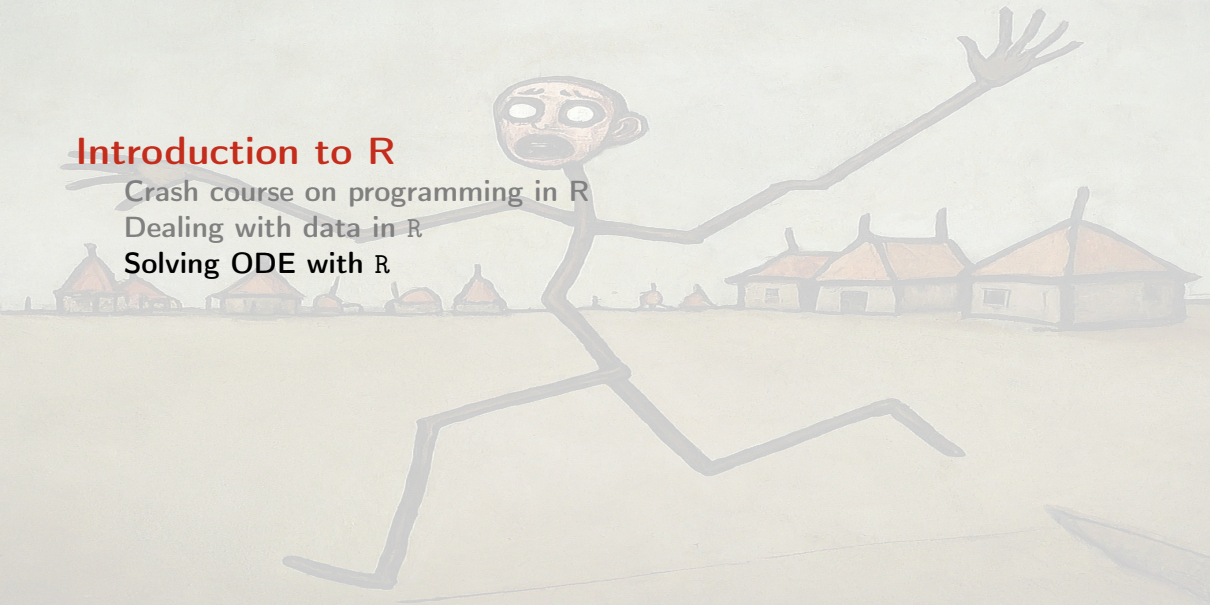


Introduction to R

Crash course on programming in R

Dealing with data in R

Solving ODE with R



The deSolve library

As I have already pointed out, deSolve:

The functions provide an interface to the FORTRAN functions 'lsoda', 'lsodar', 'lsode', 'lsodes' of the 'ODEPACK' collection, to the FORTRAN functions 'dvide', 'zvode' and 'daspk' and a C-implementation of solvers of the 'Runge-Kutta' family with fixed or variable time steps

- ▶ You are benefiting from years of experience: ODEPACK is a set of Fortran (originally 77!) solvers developed at Lawrence Livermore National Laboratory (LLNL) starting in the late 70s
- ▶ Other good solvers are also included, those written in C
- ▶ Refer to the package help for details

Using deSolve for simple ODEs

As with more numerical solvers, you need to write a function returning the value of the right hand side of your equation (the vector field) at a given point in phase space, then call this function from the solver

```
library(deSolve)
rhs_logistic <- function(t, x, p) {
  with(as.list(x), {
    dN <- p$r * N *(1-N/p$K)
    return(list(dN))
  })
}
params = list(r = 0.1, K = 100)
IC = c(N = 50)
times = seq(0, 100, 1)
sol <- ode(IC, times, rhs_logistic, params)
```

This also works: add `p` to arguments of `as.list` and thus use without `p$` prefix

```
library(deSolve)
rhs_logistic <- function(t, x, p) {
  with(as.list(c(x, p)), {
    dN <- r * N *(1-N/K)
    return(list(dN))
  })
}
params = list(r = 0.1, K = 100)
IC = c(N = 50)
times = seq(0, 100, 1)
sol <- ode(IC, times, rhs_logistic, params)
```

In this case, beware of not having a variable and a parameter with the same name..

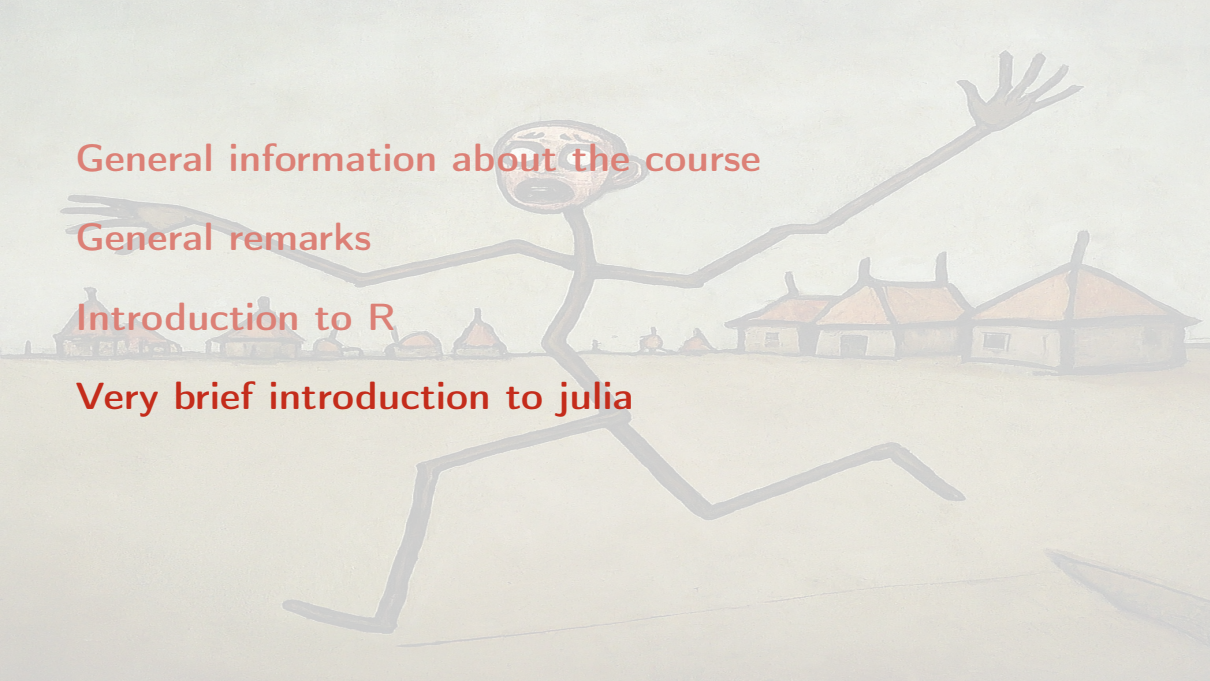
Default method: lsoda

lsoda switches automatically between stiff and nonstiff methods

You can also specify other methods: "lsode", "lsodes", "lsodar", "vode", "daspk", "euler", "rk4", "ode23", "ode45", "radau", "bdf", "bdf_d", "adams", "impAdams" or "impAdams_d", "iteration" (the latter for discrete-time systems)

```
ode(y, times, func, parms, method = "ode45")
```

You can even implement your own integration method



General information about the course

General remarks

Introduction to R

Very brief introduction to julia

CAN I HAVE THIS WRAPPED UP TO GO?

To finish, we use the command `purl` to generate an R file (L01-course-organisation-and-intro-R-julia.R) in the CODE directory with all the code chunks in this Rnw file

```
# From https://stackoverflow.com/questions/36868287/purl-within-knit-duplica
rmd_chunks_to_r_temp <- function(file){
  callr::r(function(file, temp){
    out_file = sprintf("../CODE/%s", gsub(".Rnw", ".R", file))
    knitr::purl(file, output = out_file, documentation = 1)
  }, args = list(file))
}
rmd_chunks_to_r_temp("L01-course-organisation-and-intro-R-julia.Rnw")

## [1] "../CODE/L01-course-organisation-and-intro-R-julia.R"
```

About that R file

Source the file `L01-course-organisation-and-intro-R-julia.R` (in the `CODE` directory) in R to reproduce all the results in these slides

Some small changes are required; for instance, when sourcing (instead of knitting or interactively), some figures are created but not printed, so in the R file, you need to print them “manually” (set the output to some variable and print them)

```
pp = ggplot(...)  
print(pp)
```

Bibliography I