Presentation of the course and setting up R

MATH 2740 – Mathematics of Data Science – Lecture 01

Julien Arino
julien.arino@umanitoba.ca

Department of Mathematics @ University of Manitoba

Fall 2026

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

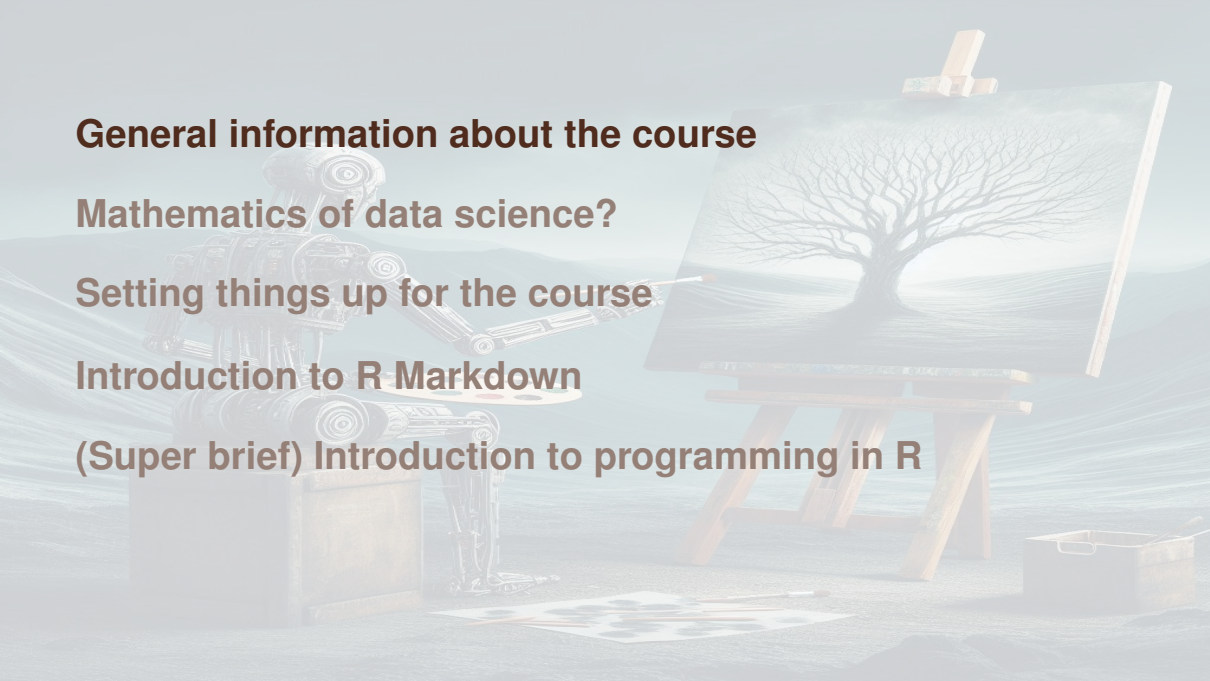
General information about the course

Mathematics of data science?

Setting things up for the course

Introduction to R Markdown

(Super brief) Introduction to programming in R



General information about the course

Mathematics of data science?

Setting things up for the course

Introduction to R Markdown

(Super brief) Introduction to programming in R

Foreword

- "Numerical dates" are in the form YYYY-MM-DD (e.g, these slides were compiled on 2026-09-15)
- Times are 24h (HHMM)
- Units are SI

In case you want to know, the slides in the course are `Rnw` compiled using `R`. All (including source code) are available on GitHub here

Getting in touch

- julien.arino@umanitoba.ca
- Please use your `myumanitoba` email address. Use a tag such as `[MATH 2740]` in your subject line, if you want to be read..
- Word of warning: I am bad with email! I do answer questions after class or during office hours

Office hours

- Because of the ongoing renovation of Machray Hall, I am sharing an office with 8 colleagues, connected to another office shared by 8 colleagues
- $\Longrightarrow$ it is **impossible** to see you in my office
- I have booked 316 St Paul's from 1400 to 1800 on Tuesday for office hours
- Seeing you outside of office hours is impossible without ample warning, since I actually need to book a room

Course website - UMLearn

- All information about the course is posted on UMLearn
- It is **your responsibility** to check the UMLearn site regularly: *Announcements* is how I normally communicate with you about the course
- (Remember to hit the link at the top of the page that says *MATH-2740-A01 - Mathematics of Data Science*, sometimes UMLearn takes you directly to Content, which is not where Announcements are)

Important – I **DO NOT** answer emails that ask about things easy to find on UMLearn

Lectures

- TR 1130–1245 in 100 Fletcher Argue

- Videos for the course as I taught it in 2021 are available as a YouTube playlist. There is no guarantee that that the content will be the same this year, but there will be commonalities for sure

Tutorials

Section	Day and time	Location
B01	W 0830–0920	P230 Duff Roblin
B03	W 0930–1020	303 Tier
B04	W 1130–1220	P230 Duff Roblin

- In tutorials, you will review some of the mathematical content and work on computations
- Three (3) Lab Tests (45 minutes) will be held during tutorials (dates later in the slides)

Work in labs

- The lecture notes contain problems that will be worked on in labs (when not in a Lab Test)
- During the tutorial, the TA will go over some of the problems
- A dedicated solutions manual will be posted on UMLearn
- Solutions will only be posted after the corresponding tutorial period has ended

R worksheets

- Working on R coding is important in this course, although because of LLMs, it is not realistic anymore to use assignments to test this capacity
- Worksheets will be posted on UMLearn at the start of each week
- Solutions will be posted on UMLearn the next week
- They are meant as practice and will not be graded
- Lab Tests, the Midterm and Final Examinations will all contain *one* question asking you to explain what a piece of R code does

Evaluation

- 3 Lab Tests during tutorials (**10%** each for a total **30%**)
- 1 midterm on Thursday 5 November 2026 1800–2000 (**30%**)
- 1 final examination during the December 2026 Examination period (**40%**)
- In all written work, explain what you are doing. No explanation $\implies$ lost marks (even if correct)

Lab Tests

- Held during tutorials on 23 Sept, 21 Oct, 25 Nov
- 45 minutes long
- You **must** write Lab Tests in the tutorial you are registered in
- Tests your capacity to remember definitions or results, do simple proofs, carry out computations and explain R code

Absences from scheduled evaluations

- If you miss a lab test or the midterm examination, you can either use a self-declaration form or provide proper documentation (MD's note)
- Self-declaration **must be made within 48 hours** of the event you are using it for
- They are intended for occasional and unforeseen circumstances, I **will not accept more than one during the term**
- You **must** use the form on this page to self-declare your absence

What if you missed an evaluation?

If you have proper documentation (self-declaration or MD's note), then

- Lab Tests: the weight of the missed test is shifted to the final examination
- Midterm Examination: you will write a Deferred Midterm Examination (date set by Department of Mathematics)

Miss more than two (2) evaluations (Lab Tests or Midterm) $\Longrightarrow$ **F in the course**

Unhappy about a mark

If you realise that some points were forgotten (this happens) or feel that you have been marked unfairly, feel free to reach **me** about it, preferably in person after class or during office hours

Do not bother the TAs with this, they do not have the authority to change your marks

- I will only consider requests made within seven (7) calendar days from the date you receive your mark
- When processing requests, I **remark the entire examination** $\Longrightarrow$ your overall mark could increase, stay the same **or even decrease**

Colour coding in slides & lecture notes

Memorising the definitions is part of the course. To help, definitions are colour coded

Definition 1 (Definitions)

These definitions are important, you need to know them

Definition 2 (Less important definitions)

These definitions are a little less important, you will not be asked to state them (although it is a good idea to know them anyway)

In lecture notes, colour coding is only for important stuff

Results are colour coded

Memorising some of the results is part of the course. To help, results are colour coded

Theorem 3 (Theorems)

Theorems in blue boxes are worth knowing but you will not be asked to reproduce them

Theorem 4 (Important theorems)

Theorems in red boxes are important, you should know them and be able to reproduce them

You must know how to do some proofs

There are a few proofs (not many!) that I want you to know how to do

Such proofs appear on slides like the present one, with a red background

BECOME A VOLUNTEER NOTETAKER

Why?

- Help reduce barriers for your peers
- Contribute to creating an inclusive University of Manitoba
- Receive recognition on your Experience Record

How?

- Login and upload your notes directly at https://sasclockwork.cc.umanitoba.ca/ClockWork/
- Submit notes in your preferred style (typed or handwritten)
- It only takes an extra 15 minutes weekly
- Full instructions on to the Student Accessibility Services website

https://umanitoba.ca/student-supports/accessibility#volunteer-note-sharing

Student Accessibility Services | UM

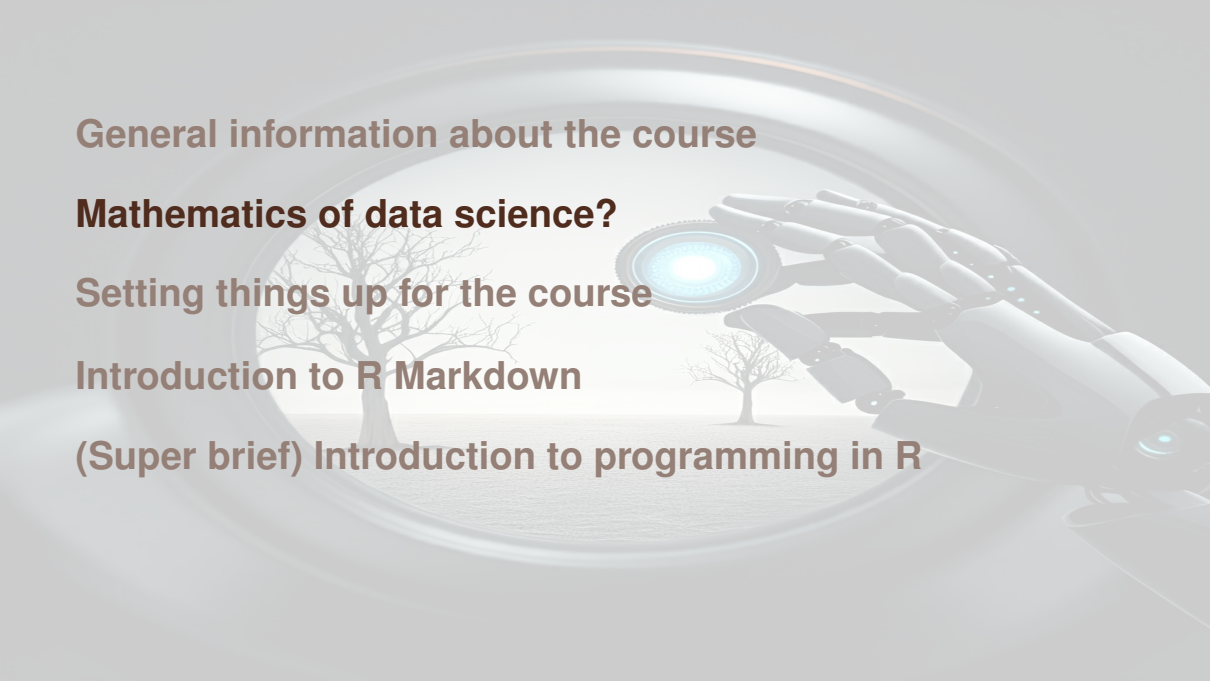
General information about the course

Mathematics of data science?

Setting things up for the course

Introduction to R Markdown

(Super brief) Introduction to programming in R



In days of yore (circa 2010)

> *Big data is like teenage sex: everyone talks about it, nobody really knows how to do it, everyone thinks everyone else is doing it, so everyone claims they are doing it...*

Attributed to Dan Ariely (Duke University)

The vocabulary has evolved, big data $\rightarrow$ complex data $\rightarrow$ data science, but Data Science remains a loosely defined concept, although things are becoming better

Data Science (according to Wikipedia)

Data science is an **interdisciplinary field** that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from **structured** and **unstructured data**, and apply knowledge and actionable insights from data across a broad range of application domains.

[..] It uses techniques and theories drawn from many fields within the context of **mathematics**, **statistics**, **computer science**, **information science**, and **domain knowledge**.

The data deluge

► Data science is nothing new (some statisticians argue it is just another name for statistics), but it has become prominent in recent years as a consequence of the unprecedented mass of information generated and collected by our modern societies

► One speaks of *information explosion* or *data deluge*. See some considerations, e.g., here

A wide variety of jobs

We have absolutely insane amounts of data and we try to make sense of it

$\Longrightarrow$ data science

However, except for the name, the situation has not improved significantly since the days of yore of Ariely's quote: data science is a hodge-podge that contains everything but the kitchen sink

To caricature

- two main types of data: structured and unstructured
- two main branches: statistics and computer science
- two main types of jobs: users and developpers

Math of Data Science?

- DS has two main branches: statistics and computer science
- DS has two main types of jobs: users and developpers

So why a course on Math of Data Science?

If you plan to be a user and are not curious about *the how* and *the why* and can tolerate errors due to misuse of methods, then you probably don't care about this course

In other cases, many of the concepts used have their roots in math and to understand where the methods are coming from and, even more importantly, to develop new methods, math is often required

Warning! We barely brush the surface

- Some techniques from linear algebra
- Some graph theory ideas
- A little bit of multivariable calculus

There is a lot more to see!!!

Prerequisites / What you will learn

► (MATH 1210 or MATH 1220 or MATH 1300) and (MATH 1232 or MATH 1700 or MATH 1710)

$\Rightarrow$ You **must know and be comfortable** with 1st year linear algebra (3 CH) and 1st year calculus (6 CH)

► We need more: some stuff you would learn in 2090 (Linear Algebra 2), some stuff from 2130, 2150 or 2720 (Multivariable Calculus) and some stuff from 2070 (Graph Theory)

Focus here is not on mathematical “precision”

- We won’t do complicated proofs. I will show some when they are useful in understanding *why* something works

- We will cover just enough of the mentioned math topics that you can understand *how* to do things

- In classic math courses, we work on “small” examples so we can work them by hand and are able to do them in tests

Here, we will do small examples by hand, indeed. But you will do regularly sized examples in computer worksheets

General information about the course

Mathematics of data science?

Setting things up for the course

Introduction to R Markdown

(Super brief) Introduction to programming in R

We will be programming quite a bit

As already indicated, Data Science is a very hands-on discipline. We will be programming quite a bit in this course

Indeed, we can work out some of the examples "by hand", but to make things interesting, we typically need to consider larger examples where hand calculations are not pleasant or not even feasible

R versus Python

Slightly different take on life :)

In short: `Python` is more CS, `R` is more Stats/Math

Both are good languages for data science

In this course, we use `R`

R was originally for stats but is now more

- Open source version of S
- Appeared in 1993
- Now version 4.6
- 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 'dvode', '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

Getting your computer ready for the course

All computer coding is in R

$\Longrightarrow$ you need to find a way to run R. Next slide: some methods, from the easiest to the most challenging.

Note that all options described below are Open Source (completely free)

In short...

- 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

Install R and RStudio

This is probably the best option if you intend to go a little further than what we will do in the course. `R` is available on most platforms, while `RStudio` is available on most platforms except for Linux ARM devices (but can be compiled there)

Visit https://www.r-project.org/

Choose your version: Windows or Mac. Under Linux, you can install directly from your package manager (e.g., `sudo apt install R-base` for Debian-based distros)

To install RStudio, see here

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..
- Sweave/knitr: LaTeX incorporating R commands. Useful for generating reports. Not used as much as Rmarkdown these days (but used for these slides)

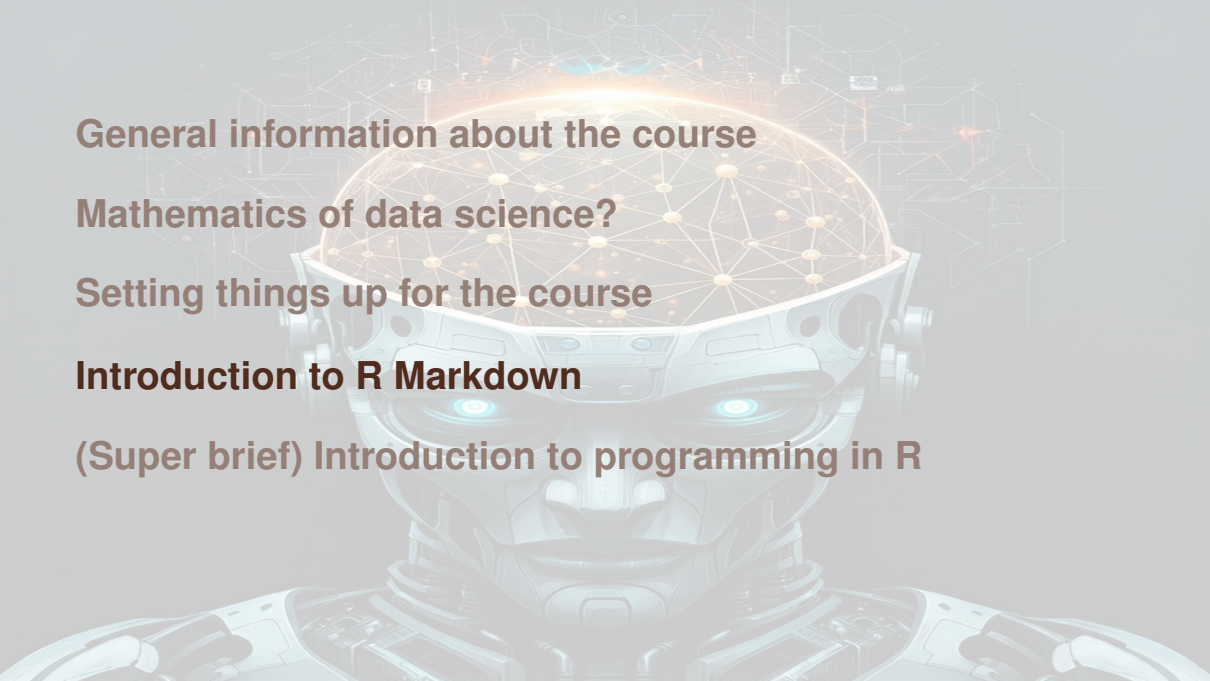
General information about the course

Mathematics of data science?

Setting things up for the course

Introduction to R Markdown

(Super brief) Introduction to programming in R



R Markdown?

- File format for making dynamic documents with R
- Combines code, its results and narrative text in a single document
- Uses simple `Markdown` syntax for text and `R` code chunks for analysis
- From one '.Rmd' file, you can create HTML, PDF, Word documents, presentations, ...

The YAML header

Every R Markdown document starts with a YAML header, enclosed by `---`. This controls the overall properties of the document

Example YAML for a PDF document

```
---
title: "My Report"
author: "My Name"
date: "2025-08-17"
output:
  pdf_document:
    toc: true
    number_sections: true
---
```

Basic Text Formatting

Markdown provides a simple syntax for formatting text

- `# Header` creates a section title, `## Sub-header` creates a subsection title, etc.
- `*italic text*` produces *italic text*
- `**bold text**` produces **bold text**
- `` `code font` `` produces `code font`

Lists

Unordered lists start with * or -

```
* Item 1
* Item 2
```

Ordered lists use numbers

```
1. First Item
2. Second Item
```

(Once the numbered list is “initiated”, the number doesn’t matter, you can write 1., 1., 1., etc., provided you have a new line each time)

Chunks: the core of R Markdown

R code is placed in “chunks”, which start with ```` ```{r chunk-name} ```` and end with ```` ``` ````

````
```{r cars-summary}
# Your R code goes here
summary(cars)
```
````

Chunk names (`cars-summary` here) are not mandatory (you could write ‘‘‘`{r}` but are very useful, in particular when debugging

Chunk output

When the document is "knit," the R code is executed and its output is embedded in the final document

````
```{r cars-summary}
summary(cars)
```
````

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Controlling chunks with options

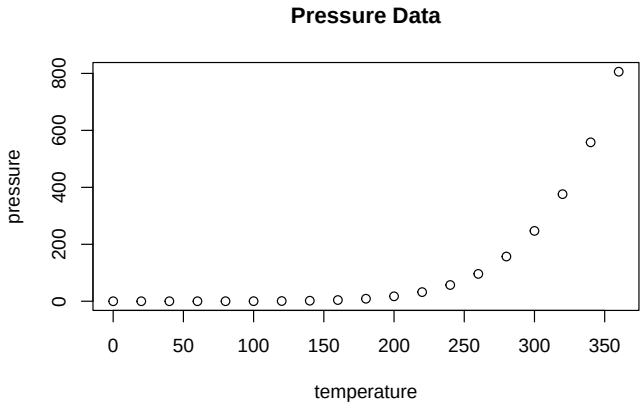
Chunk behavior is controlled by options inside the curly braces

- `echo=FALSE` hides the `R` code, but shows the output
- `eval=FALSE` shows the code, but does not execute it
- `include=FALSE` executes the code, but hides both code and output
- `warning=FALSE`, `message=FALSE` hides warnings or messages
- `fig.width=5`, `fig.height=4` sets figure dimensions

Figure chunks

Plots are automatically embedded

```
```{r pressure-plot, echo=FALSE, fig.cap="Pressure Data"}
plot(pressure)
```
```



Inline R Code

You can embed R code directly into text with backticks: `` `r ...` ``

```
The `cars` dataset has `r nrow(cars)` rows.
```

The cars dataset has 50 rows.

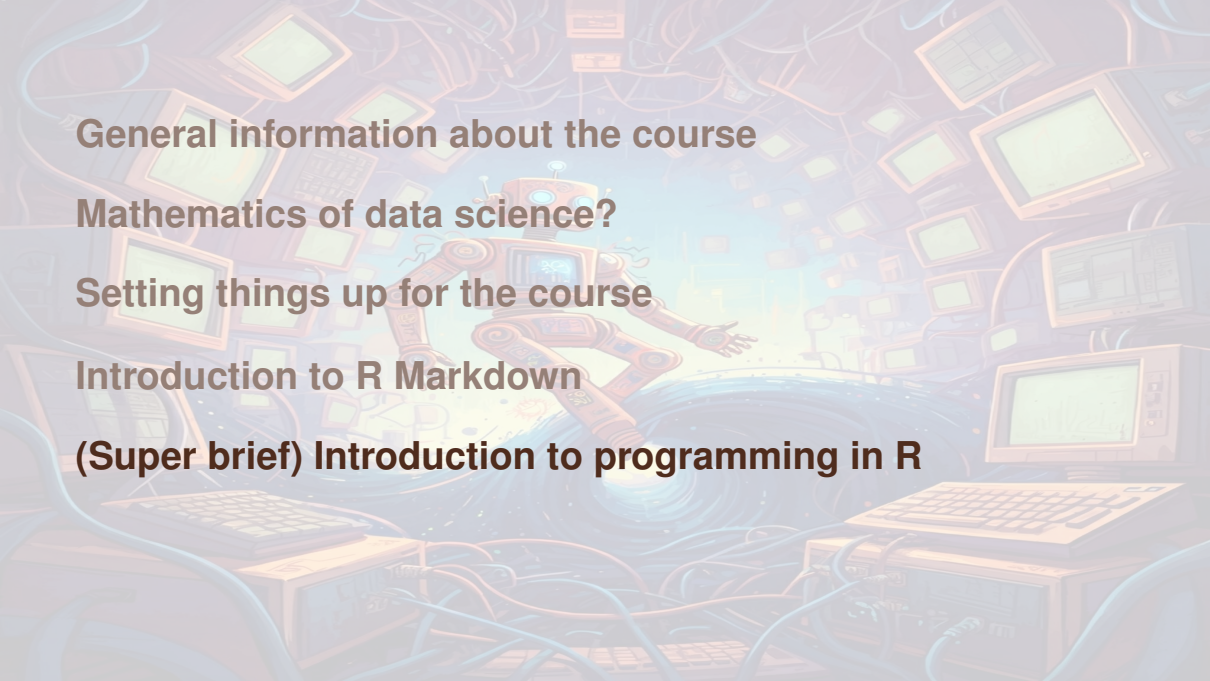
General information about the course

Mathematics of data science?

Setting things up for the course

Introduction to R Markdown

(Super brief) Introduction to programming in R



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..

Assignment

Two ways:

```
X <- 10
```

or

```
X = 10
```

First version is preferred by R purists.. I don't really care

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"
```

```r
L[1]
```

```
## $a
## [1] 10
```

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

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

```r
L$a
```

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

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

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

```r
L$another_name
```

```
## [1] "Plouf plouf"
```

Vectors

```
x = 1:10
y <- c(x, 12)
y
z = c("red", "blue")
z
z = c(z, 1)
z
```

Note that 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

Matrix (or vector) of zeros

```
A <- mat.or.vec(nr = 2, nc = 3)
A
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    0
## [2,]    0    0    0
```

Matrix are column-first

```
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
```

Remark that here and elsewhere, naming the arguments (e.g., `nr = 2`) allows to use arguments in any order

Matrix operations

Probably the biggest annoyance in R compared to other languages

- The notation `A*B` is the *Hadamard product* $A \circ B$ (what would be denoted `A.*B` in matlab), not the standard matrix multiplication
- Matrix multiplication is written `A %*% B`

Vector operations

Vector addition is also frustrating. Say you create the vector

```
x = 1:10
x

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

Then `x+1` gives

```
x+1

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

i.e., adds 1 to all entries in the vector. Beware of this in particular when addressing sets of indices in lists, vectors or matrices

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. Because there is a parallel version (`parLapply`) that we will see later, worth learning

```
l = list()
for (i in 1:10) {
        l[[i]] = runif(i)
}
lapply(X = l, FUN = mean)
```

or, to make a vector

```
unlist(lapply(X = l, FUN = mean))
```

or

```
sapply(X = l, FUN = mean)
```

“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))
```

gives

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

Just recall: the argument to the function you define is a list entry (`l[[1]]`, `l[[2]]`, etc., here)

Avoid parameter variation loops with expand.grid

```
# Suppose we want to vary 3 parameters
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]
    }
}
```

Can I have this wrapped up to go?

For each slide set including `R` code, we generate an `R` file in the `CODE` directory with all the code chunks in this `Rnw` file and no LaTeX

Source the file (if you are reading `SLIDES/filename.pdf`, then you want `CODE/filename.R`) to reproduce all results in the slide set without generating a pdf

Some small changes might be required (for instance, in the file for this lecture, quite a lot is commented out)