Introduction to OSCAR and Julia

Lecture 1

A first hands-on session

Ilaria Colazzo

University of Leeds (UK)

About the reading group

This lecture is part of the **OSCAR and JuliaLang Reading Group**.

The idea is simple:

- we learn by reading and trying things
- we discuss examples together
- we keep space for questions we do not yet know how to answer

How the reading group works

Everyone is encouraged to contribute.

You might present:

- a package
- a mathematical method
- or a topic close to your own research

This does **not** need to be a polished talk.

The goal is to explore together.

Interactive and question-driven

It is completely fine to come with:

- an open question
- a computation you do not yet know how to do
- a package or function you want to understand better
- a research example you want to test

Please interrupt, ask questions, and suggest variants as we go.

What is OSCAR?

OSCAR is an open-source computer algebra system built in Julia.

It gives us a way to work with mathematical objects such as:

- groups
- rings and fields
- matrices
- polynomials
- objects from number theory and algebraic geometry

Why use Julia + OSCAR?

A good first question is:

Why do we need a system like OSCAR?

- calculators work mainly with numbers
- OSCAR works with mathematical objects
- it lets us compute exactly, experiment, and ask bigger questions

What are we going to do?

- **Getting started**
  - starting Julia
  - loading OSCAR
  - basic commands
- **Basic computations**
  - integers and rationals
  - polynomial rings
  - finite fields
- **How to get help**
  - `?`
  - tab completion
  - examples from the docs
- **First algebraic objects**
  - matrices
  - groups
  - polynomials
- **Exercises**

Starting Julia and OSCAR

Once Julia is open, we start with

```
using Oscar
```

If OSCAR is not installed yet:

```
using Pkg
Pkg.add("Oscar")
using Oscar
```

- who has already used Julia before?

First session

Let us begin with a few commands we can understand immediately.

```
2 + 5
gcd(2025, 15)
factor(2025)
divisors(60)
is_prime(101)
```

- which of these are plain Julia?
- which outputs already look mathematically useful?

Julia as a fancy calculator

Here are a few more quick examples.

```
2^10
binomial(10, 3)
sum(i^2 for i in 1:10)
factorial(8)
```

- how does this compare with the systems you already use?

Variables, loops, and functions

We are not limited to one-line calculations.

```
for i in 1:5
    println(i^2)
end
```

```
f(n) = sum(i for i in 1:n)
f(10)
```

Main idea:

- we can do mathematics interactively, and we can also write small programs

How do we get help?

When we meet a new command or object, we do not need to guess.

We can:

- ask Julia for help with `?`
- read examples in the documentation
- use tab completion to discover names
- try a small example and see what happens

How do we get help?

Examples

```
?gcd
?factor
?polynomial_ring
?symmetric_group
```

What do you notice in the documentation?

Look for three things:

- a short description
- an example
- the kind of object the command works with

What can I do with this object?

A very common question is:

"I have an object: how do I find out what I can do with it?"

For example, if we have

```
using Oscar
R, x = polynomial_ring(QQ, "x")
f = x^3 + 2*x + 1
```

Aside: Careful with names

First we defined a function:

```
f(n) = sum(i for i in 1:n)
```

Then we tried to do this:

```
R, x = polynomial_ring(QQ, "x")
f = x^3 + 2*x + 1
```

Why is this a problem?

- the name `f` was already attached to a function
- now we want it to refer to a polynomial
- Julia keeps track of the kind of object we are working with

Let's get back to our example

```
using Oscar
R, x = polynomial_ring(QQ, "x")
g = x^3 + 2*x + 1
```

we can explore with:

```
typeof(g)
methods(degree)
methodswith(typeof(g))
```

- what kind of object is `g`?
- which commands seem relevant?
- which ones would you try first?

Polynomial rings

Let us build a polynomial ring and see what becomes available.

```
R, x = polynomial_ring(QQ, "x")
p = x^4 - 1
q = x^2 - 1
gcd(p, q)
factor(p)
```

- what is the `gcd` here?
- how does the factorisation reflect the algebra we know?

Finite fields

Now let us change the coefficient field.

```
F = GF(7)
R, x = polynomial_ring(F, "x")
expand((x + 3)^7)
```

This gives an error.

- What can we learn from it?
- not every function works on every kind of object
- the type of the object matters
- the error message tells us which combination is unsupported

Finite fields

Do we need `expand`?

```
F = GF(7)
R, x = polynomial_ring(F, "x")
p = (x + 3)^7;
p
```

Matrices

We can also work with exact linear algebra.

```
A = matrix(QQ, [1 2; 3 4])
det(A)
rank(A)
inv(A)
```

- which of these outputs would you expect before running the code?
- why is it nice to work over `QQ` here?

A first group

Let us create a group and inspect it.

```
G = symmetric_group(4)
order(G)
gens(G)
```

Natural follow-up questions:

- what does `symmetric_group(4)` mean?
- what are these generators?
- how could we compute the order of a particular element?

Exercise 1

A summation identity

Write a function that checks the identity

$$1 + 2 + \cdots + n = \frac{n(n+1)}{2}$$

for the first few positive integers.

- how would you test several values of `n`?
- do you want your function to print something, or return `true` / `false`?

Exercise 2

Sieve of Eratosthenes

Write a function that returns all prime numbers up to `n`.

Hints:

- start with the list `2:n`
- remove multiples step by step
- compare your result with built-in functionality

Question:

- what is the clearest way to organise the loop?

Exercise 3

A polynomial identity over $\mathbb{F}_7$

Work over the field with 7 elements and investigate

$$(x + a)^7$$

for several values of `a` .

Questions:

- what pattern do you observe?
- does the answer depend on `a` in the way you expected?
- can you explain it mathematically?

Exercise 4

The Klein four-group

Construct the Klein four-group and explore it.

Tasks:

- compute its order
- list its elements if possible
- determine the order of each element
- check whether the group is cyclic

Good follow-up:

- which feature of this group is visible very quickly from computation?

Optional extra exercise

Lagrange's theorem in practice

Pick a finite group `G` and a subgroup `H`.

Compute:

- `order(G)`
- `order(H)`

Then check experimentally that `order(H)` divides `order(G)`.

Take-away points

By the end of today, I would like you to feel that:

- Julia is a flexible language for mathematical experimentation
- OSCAR gives us interesting algebraic objects very quickly
- small examples already lead to meaningful mathematics
- asking questions and modifying examples is part of the method

Resources for further learning

- Official Julia Documentation
- Julia Academy

Thank you

Questions?