Deep dive into the Xeus-based Cling kernel for Jupyter

QuantStack SAS - https://quantstack.net - @QuantStack

Sylvain Corlay

Founder and CEO of **QuantStack**

Open Source Developer

- ***Jupyter* Steering Committee Member**
- Core developer of **conda-forge**.
- Co-creator of Voilà, Xeus, Xtensor

Open Source volunteer work

- Director at ***NumFOCUS***
- Organizer of the ***PyData Paris*** Meetup, vice chair of ***JupyterCon***.

🎖️ Recipient of the **2017 ACM Software System Award** for Project Jupyter

🐙 🐦 @SylvainCorlay

QuantStack
Scientific Computing

QuantStack is

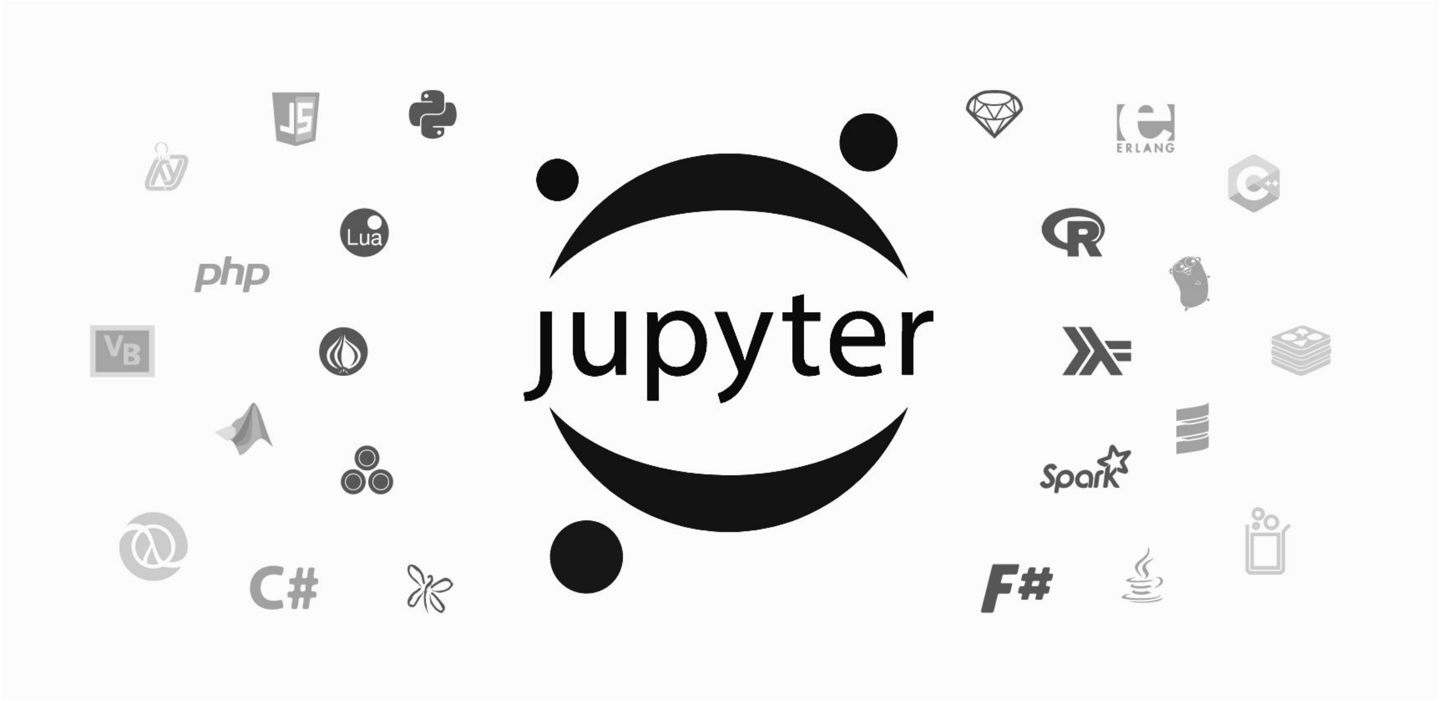
- An **open-source development studio** specialized in scientific computing
- A **team of maintainers of major opens-source projects** of the stack

  (Jupyter, Conda-Forge, Xtensor, Voilà, Mamba, Quetz, ROS…)

We provide

- professional support and development services for this ecosystem
- custom development and consulting services for the key software of the open-source scientific computing ecosystem.

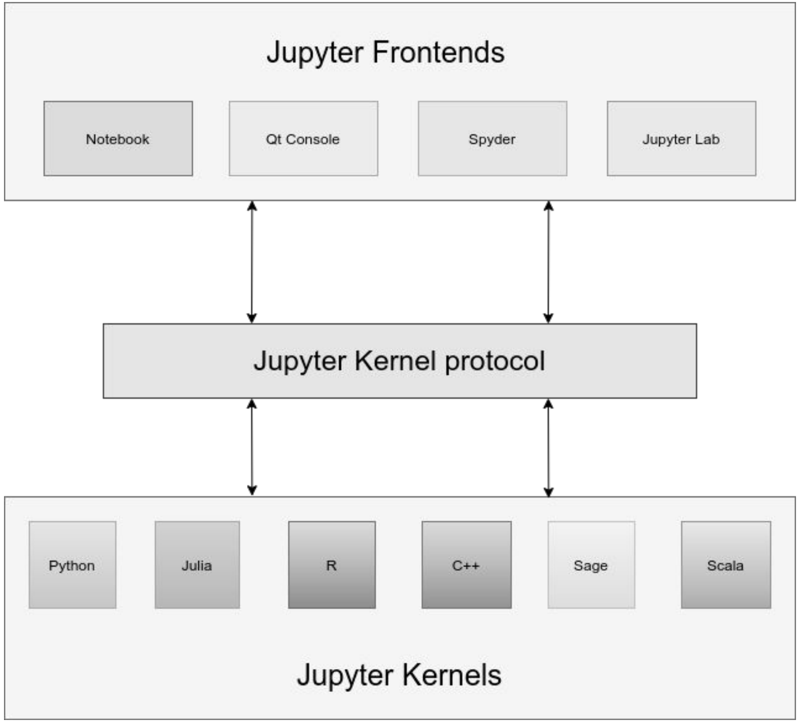
Jupyter's language agnosticism



Jupyter's language agnosticism

The **Kernel** is the part of the Jupyter infrastructure responsible for executing the user's code.

From the perspective of the other components of the Jupyter stack, a kernel is merely a process implementing a **well-specified communication protocol**.



Authoring Jupyter kernels

The existing kernels:

JavaScript, C++, Python, Julia, R, Haskell, Go, C#, Robotframework, OCaml, Perl, IDL, Scala, Fortran, Octave, Scilab, SQLite, Ruby...

There are Jupyter kernels for dozens of languages.

But these kernels have very different levels of quality and support for the features of the protocol.

How to make new language kernel?

1. Rewrite in **from scratch** in e.g. the target language... Not that easy.
    - Deal with a complex concurrent programming models
    - Make use of the ZMQ interprocess communication library
    - Cryptographically sign messages
    - Properly implement JSON messages schemas
2. Use a **framework**
    - Ipykernel
    - Xeus

Authoring Jupyter kernels

The wrapper kernel approach

IPykernel includes a reference implementation of the Kernel protocol.

To make a kernel with ipykernel, inherit from **`ipykernel.kernelbase.Kernel`** and implement the language-specific parts in the derived class.

This is the approach used for the kernel shipped with Cling.

Issues with the wrapper approach:

- Dependency on the Python runtime. *(consequences for the packaging of the Cling project).*
- The wrapped interpreter may not have a Python API, and we need to make one.
- We may need to expose the API of the kernel to the target language for advanced use cases (widgets, rich display...).
- A native implementation may be more efficient.

Authoring Jupyter kernels

What is Xeus?

Xeus is a modern C++ implementation of the Jupyter protocol. It is *not* a kernel, but a *tool* to make new kernels.

To make a kernel with Xeus, inherit from **`xeus::xinterpreter`** and implement the language-specific parts in the derived class.

This is the approach used in xeus-cling.

Our motivation for starting Xeus

- We were asked by a client to make a lightweight kernel for a DSL. IPykernel seemed overkill and too heavy.
- We think that the kernel protocol is stable enough for a strongly typed reference implementation to exist.
- Most interpreters are written in C or offer a C API. This makes it easy to embed them in a C++ application.

Xeus: an ecosystem of Jupyter kernels

Xeus-python: A xeus-based Jupyter kernel for the Python language

- GitHub Try it Here
- Used in SlicerJupyter for embedding in the Slicer Qt application.
- Supports the new JupyterLab interactive debugger.

Xeus-cling: A xeus & cling-based Jupyter for the C++ language

- GitHub Try it Here
- Started as a demonstrator for the Xeus framework. Used to teach C++ at Université Paris Sud.

Xeus-SQL: (And Xeus-SQLite)**:** Xeus-based kernels for SQL

- GitHub Try it Here

Xeus-Robot: Xeus-based kernel for RobotFramework

- GitHub Try it Here
- RobotFramework is an open-source language and framework for Robotic Process Automation.

LFortran: LFortran is an LLVM-based Fortran compiler and interpreter. It includes a Xeus-based kernel

- GitHub Try it Here

```
$ jupyter console --kernel=fortran
Run with XEUS 0.24.1
Jupyter console 6.1.0

LFortran
Jupyter kernel for Fortran
Fortran
In [1]: integer :: x

In [2]: x = 5

In [3]: x*3.5
Out[3]: 17.500000

In [4]:
```

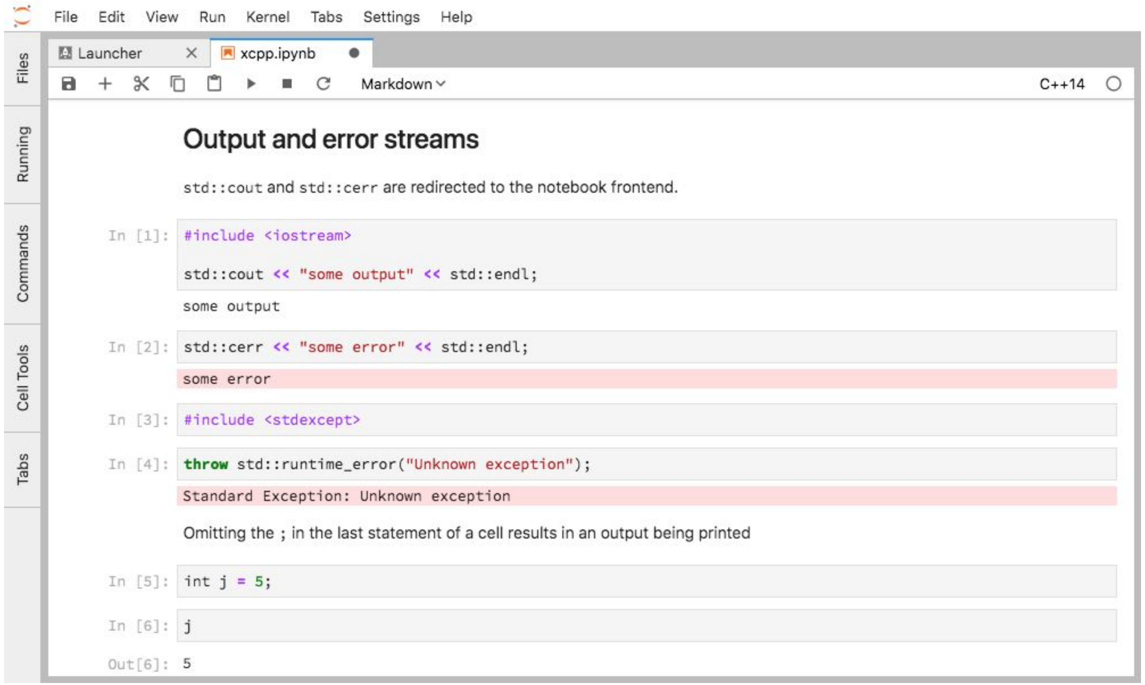
And many more (xeus-octave, xeus-fift, JuniperKernel)...

Xeus-cling: A C++ Jupyter kernel

… based on cling and Xeus

Never give a live demo

Xeus-cling: redirecting streams



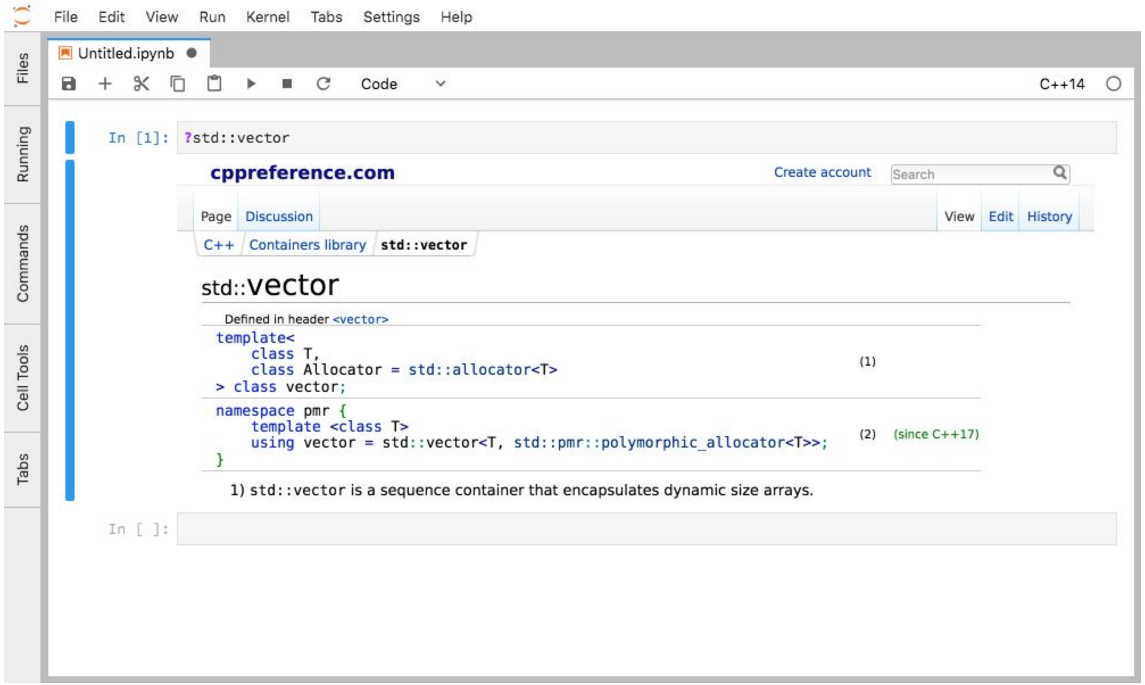
The main means of printing are redirected to the front-end.

- **std::cout** and **std::cerr**, asl well as **printf** are redirected to the front-end.

However.

- **std::clog** prints to the kernel standard output, which can be used for logging.

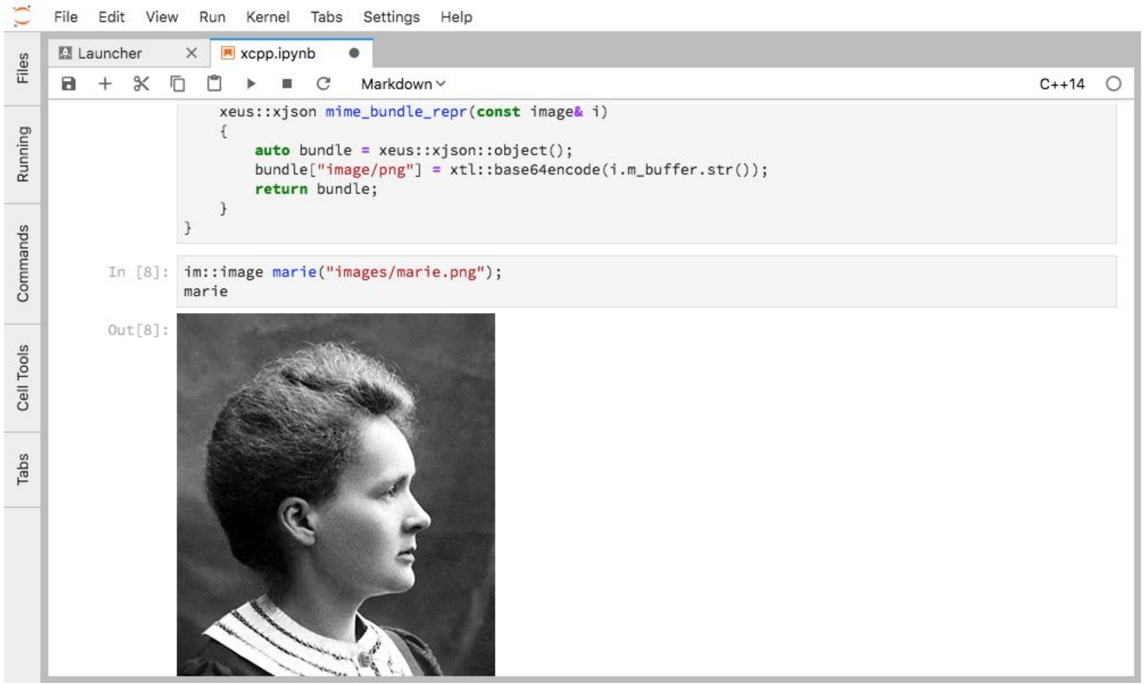
Xeus-cling: inline help



The "?" magic can be used to get inline help on types and functions.

- For the standard library makes use of cppreference.
- This is extensible for user-defined libraries. (Demo example with xtensor)

Xeus-cling: rich outputs



Xeus-cling leverages the Jupyter rich mime type rendering system.

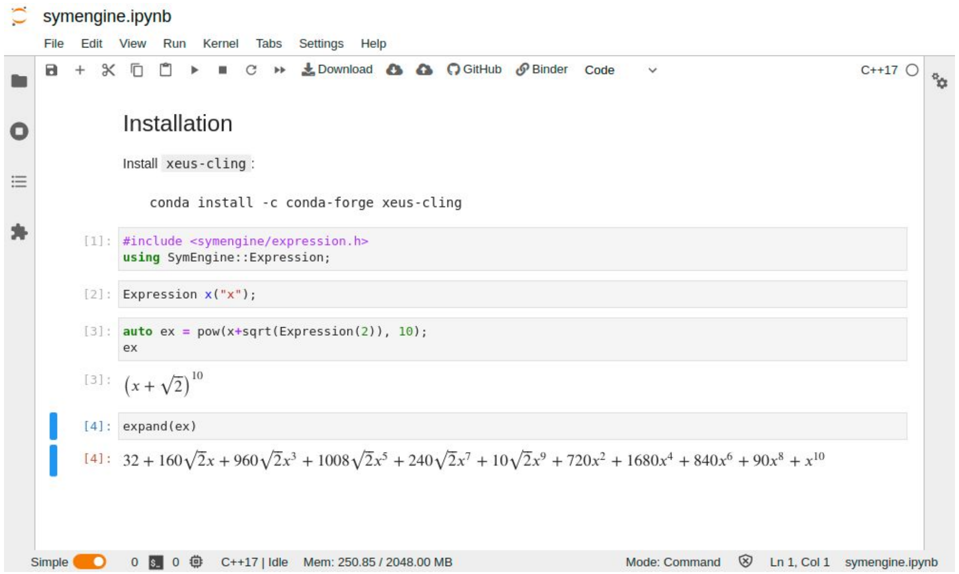
- This can be defined for any type by specializing the **mime_bundle_repr** function for the said type.

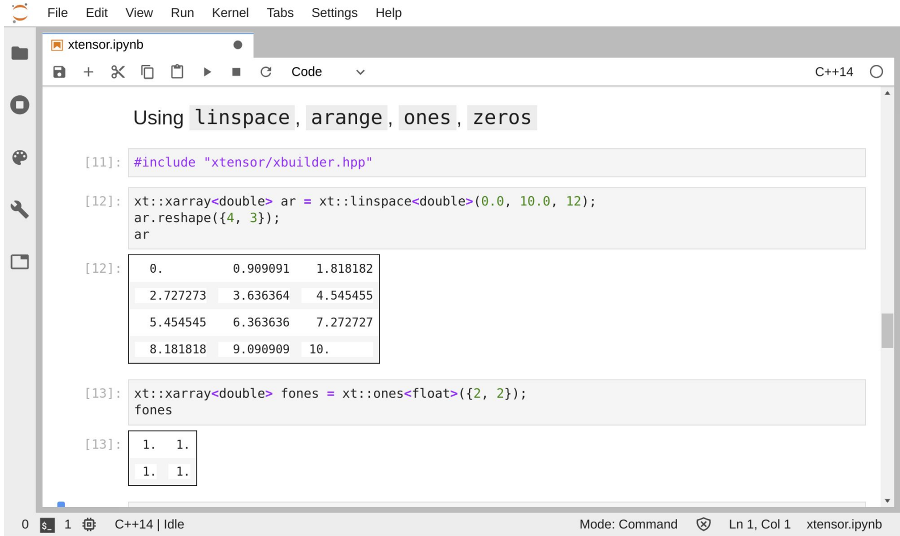
- This overload is picked up by xeus-cling through **argument dependent lookup**.

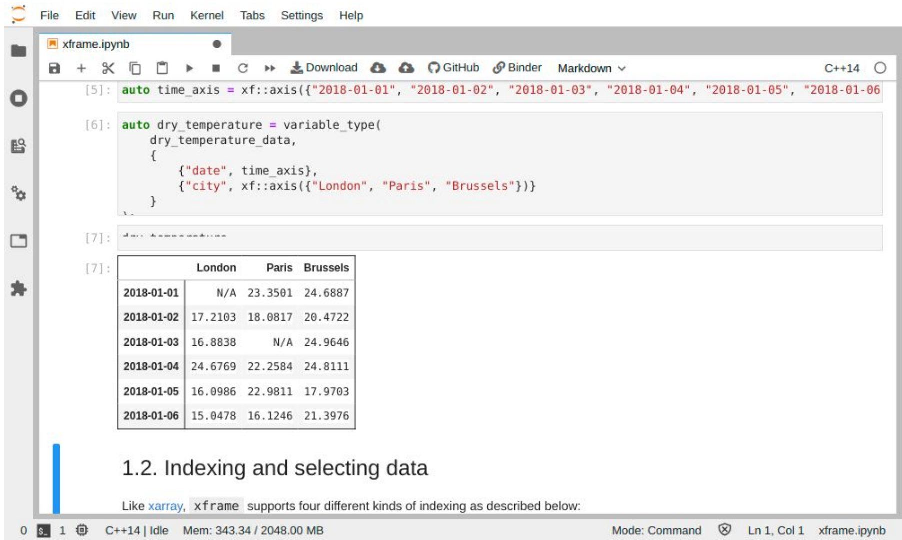
Rich output

Examples with

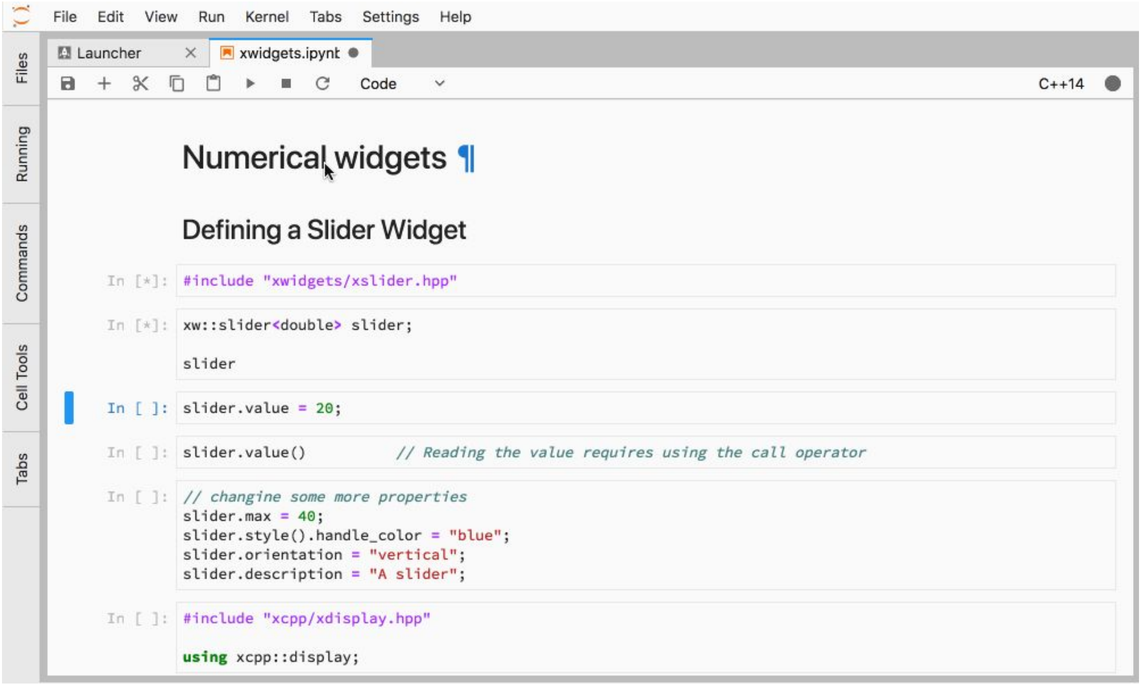
- **Xtensor** and **Xframe** (HTML tables for visualizing tensors)
- **Symengine** (MathJax)







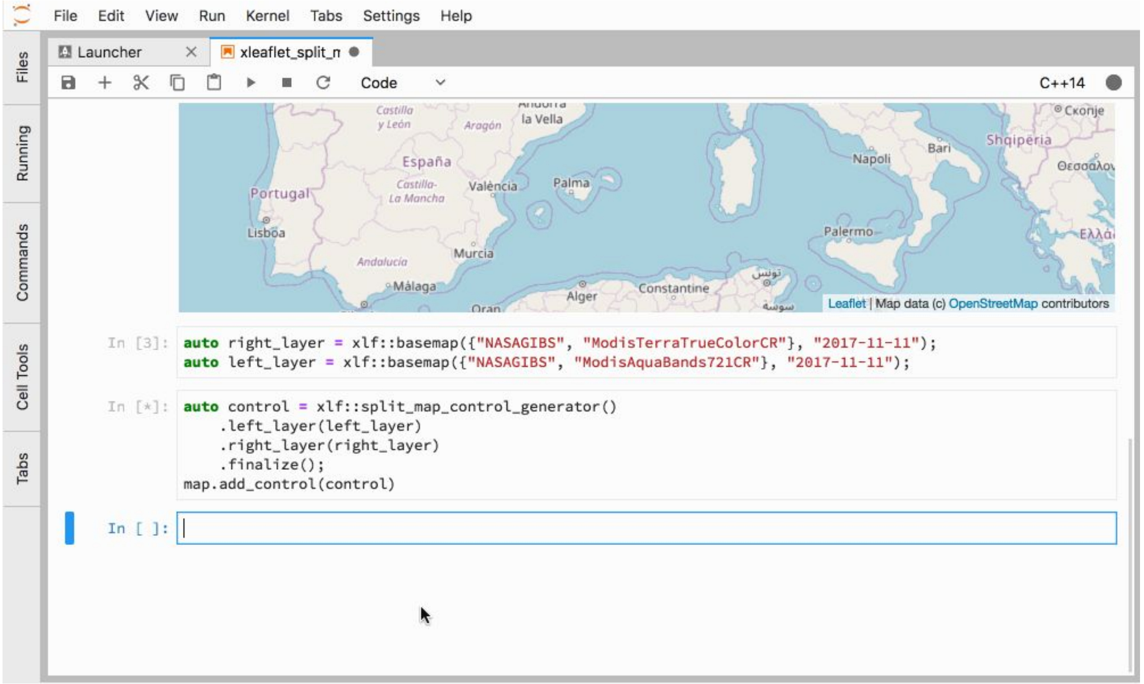
Xeus-cling: interactive widgets



Interactive widgets

- A C++ backend for the Jupyter interactive widgets is available in the xwidgets package.

Xeus-cling: more data visualization



Jupyter widgets are a framework

- Xleaflet
- Xwebrtc
- Xplot

And many more coming…

An opportunity for interactive C++: leverage the huge ecosystem of JavaScript data visualization tools.

Xeus Cling: how to get started

We provide a xeus-cling package on conda-forge.
It can be installed with mamba or conda

```
mamba install xeus-cling
```

You can also try it out online on binder.

Xeus Cling: about the future?

- Provide a VS2019 build on conda-forge to fully support windows
    - Windows support is tested on CI but we don't provide a build for it
    - We will wait for the LLVM9-based version of cling.
- Work with library authors on including cling pragmas in library headers
    - https://github.com/xtensor-stack/xtensor-blas/blob/master/include/xtensor-blas/xblas_config_cling.hpp.in
- Dashboarding with Voilà and Xeus-cling
    - build notebooks into full executables that don't require the cling runtime, and respond to the protocol as static backend for Voilà apps
    - Subject of an internship?
- Work with upstream on improving rich mime type rendering?
- What is needed for a an upstream adoption in ROOT?
    - Provide an extensible magics system providing all the dots commands?
- Implementing the Jupyter Debug Protocol in xeus-cling to enable visual debugging in JupyterLab.