Sage and Latex interaction

Sebastiano Tronto - sebastiano.tronto@uni.lu

2021-05-07

It can happen that you need to include the results of your Sage computations and/or Sage code inside a LaTeX document. Luckily Sage provides some functions to translate its objects into LaTeX, and the listings package for LaTeX can be used to include any code (Sage, Python or any other language) in a LaTeX document.

In this document we will describe some of these interactions between LaTeX and Sage.

1 The `show()` command

Reference: [1] (`show()` is just an alternative name for `pretty_print()`).

With this command Sage will generate a picture displaying the object. The result depends on the object itself: most of them will be typeset in Latex, but for example graphics primitives (such as plots) will be displayed as pictures.

You can see it as an alternative to `print()`.

[4]:
```
s = (e^x).series(x==0, 4)
M = matrix([[1,2,3],[4,5,6],[8,9,10]])
print(s)
show(s)
print(M)
show(M)
print(pi)
show(pi)
```

```
1 + 1*x + 1/2*x^2 + 1/6*x^3 + Order(x^4)
```

$1 + 1x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \mathcal{O}\left(x^4\right)$

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

$$\left(\begin{array}{rrr} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 8 & 9 & 10 \end{array}\right)$$

```
pi
```

π

In a Jupyter notebook, the results above are displayed using MathJax.

If you are running this code in an interactive console (terminal) instead of a Jupyter notebook, you will get the Latex source code for those objects. You can force this behavior by using the `latex()` command.

2 The `latex()` command

Reference: [2]

This command is potentially very useful if you need to include the results of Sage computations in a Latex file, especially with complex objects like matrices or very large polynomials.

Technically, this is a function that returns a string, so you need to `print()` it to see the result.

[5]:
```
print(latex(s))
print("\n")
print(latex(M))
```

```
1 + 1 x + \frac{1}{2} x^{2} + \frac{1}{6} x^{3} + \mathcal{O}\left(x^{4}\right)



\left(\begin{array}{rrr}
1 & 2 & 3 \\
4 & 5 & 6 \\
8 & 9 & 10
\end{array}\right)
```

Interestingly, Sage can use matplotlib's PGF backend to generate Latex code for a plot. (PGF is the graphics language underlying TikZ, like TeX is the language underlying Latex).

[15]:
```
#latex(plot(x^2)) # The output is more than 20 pages long
```

It is probably easier to just generate the picture and include that in your Latex document with `\includegraphics`.

2.1 A Latex name for your variables

Reference: [3]

Sometimes you might want to use variables and functions that have, for example, a Greek letter as a name. You can tell Sage that you want them displayed this way when you declare them:

[16]:
```
var('epsilon', latex_name="\\varepsilon")
function('phi1', latex_name="\\phi_1")

print(phi1(epsilon))
show(phi1(epsilon) + e^epsilon)
latex(phi1(epsilon) + e^epsilon)
```

```
phi1(epsilon)
```

$e^{\varepsilon} + \phi_1(\varepsilon)$

```
[16]: e^{{\varepsilon}} + \phi_1\left({\varepsilon}\right)
```

Warning: You need to use two backspaces `\\`. The reason is that in Python (like in many other programming languages) the backslash symbol inside a string is used to print special characters, such as a newline `\n`.

3 From Jupyter to Latex

Reference: [4]

From the Jupyter menu `File > Download as` you can choose to download your work in many formats, among which there are also Latex and pdf. Personally I prefer downloading the .tex file, so then I can change the title, add an author name and make any other change I like before compiling it into a pdf file.

If you choose to download the pdf file, you might need to install some extra packages. For example I had to install `pandoc`, `texlive-XeTeX` and `texlive-Xdvi`, but this depends on your operating system and Latex distribution.

4 SageTex

Reference: [5]

With SageTex it is possible to run Sage commands directly inside Latex, using the `\sage{}` command. In this way you don't need to run your Sage code first and then copy the results in Latex. It can be useful especially for short Sage commands.

You might need to take some extra steps to make this work on your system, see the link above.

5 The Latex `listings` package

References: [6] and [7]

If you want to include some code (Sage, Python or anything else) in a Latex document you can use the listings package.

```
\usepackage{listings}

...

\begin{lstlisting}[language=Python]
for i in range(0,100):
    if i%5 == 0:
        print("Multiple of 5!")
\end{lstlisting}
```

You need to specify the language you are using with the `language=` option. This option can also be set at the beginning of the document using the `\lstset{language=Python}` command.

As an alternative, you can include a file directly without copying the code into the tex file, like you would do for a picture:

```
\lstinputlisting[language=Python]{file.py}
```

It is technically possible to include Latex listings in a markdown cell of the Jupyter notebook using this package, but it does not make much sense. So we will move to a Latex editor for the examples.

```
[ ]:
```