Julia with Emacs Org mode

Picaud V.

[2018-03-07 Wed 20:01]

Contents

1 Introduction

This post details how to use Emacs Org mode to create Julia notebooks and to perform HTML or PDF exports. I tried to get the simplest working solution.

Julia notebook functionality works out of the box thanks to ob-julia.el and this is what I am using instead of Jupiter notebooks. However, the solution to export HTML and PDF is not straightforward.

I wanted to:

- have nice Julia code snippets with full UTF8 supports,
- being able to export in both HTML and PDF, including bibliography.

I get a solution which is certainly not perfect, ideas to improve it are welcomed.

There are two points to take care of:

- LaTeX does not fully support UTF8, nor its listings package.
  - for UTF8 support I had to switch to luatex, biber and minted package
  - I also had to use proper fonts, DejaVu, to support Greek letters and mathematical symbols.
- to make the bibliography exportable in both HTML and PDF without **.org** file modification, I had to use a little trick.

The proposed solution uses:

- ob-julia.el : to support notebook functionality,
- ox-bibtex.el : used for html-export of the bibliography, requires **bibtex2html** (Debian package),
- luatex : under Debian, included in the **texlive-latex-base** package,
- biber : under Debian, the **biber** package,
- pygments : under Debian, the **python-pygments** package. Attention with **python3-pygments**, it does not on my computer. I have not investigated this.

Maybe I have forgotten something, just tell me (I am only using Linux).

There is a GitHub repository to reproduce results of this post. The **example.pdf** generated file is also present.

2 Emacs configuration

2.1 Getting ob-julia.el and ox-bibtex.el

You can found **ob-julia.el** and **ox-bibtex.el** in Org-mode Contributed Packages. Easy download can be performed using:

```
curl -o emacs_files/ob-julia.el
↪  https://code.orgmode.org/bzg/org-mode/raw/master/contrib/lisp/ob-julia.el
curl -o emacs_files/ox-bibtex.el
↪  https://code.orgmode.org/bzg/org-mode/raw/master/contrib/lisp/ox-bibtex.el
```

2.2 Minimal init.el file

This is a minimal configuration to reproduce the results. The code with its comments is self-explaining:

```
;; Use your own packages for classical stuff
(package-initialize)
;; requires Emacs speaks statistics, Org
(require 'ess-site)
(require 'org)

;; removes ugly horizontal lines in html-exported code
;; (not mandatory)
(setq org-html-keep-old-src t)

;; As ob-julia.el and ox-bibtex are less common,
;; we use a local repository.
;;
;; Usage: emacs -q --load emacs_files/init.el
;;
;; In a more usual setting one should use:
;; (require 'ob-julia.el)
;; (require 'ox-bibtex)
(load-file "emacs_files/ob-julia.el") ; works with ess-site, our notebook engine
(load-file "emacs_files/ox-bibtex.el"); used for bibliography HTML-export

;; allows julia src block (requires ob-julia.el)
(setq org-confirm-babel-evaluate nil)

(org-babel-do-load-languages
 'org-babel-load-languages
 '((julia . t)))

;; defines image width in the OrgMode buffer (this is not for html
;; exports, for this you must use #+HTML_ATTR: :width 900px for
;; instance)
;;
;; This is not mandatory, but useful when one uses the gr() Plots.jl
;; backend as it exports wide .png files. CAVEAT: use imagemagick for
;; image resizing.
;;
```

```
(setq org-image-actual-width (/ (display-pixel-width) 4))

;; uses the minted package instead of the listings one
(setq org-latex-listings 'minted)

;; defines how to generate the pdf file using lualatex + biber
(setq org-latex-pdf-process
      '("lualatex -shell-escape -interaction nonstopmode -output-directory %o %f"
        "biber %b"
        "lualatex -shell-escape -interaction nonstopmode -output-directory %o %f"
        "lualatex -shell-escape -interaction nonstopmode -output-directory %o
        ↪  %f"))
```

3 .org file configuration

For demonstration purpose we define an **.org** file example. This file is kept very simple to do not distract from the required configuration part.

3.1 LaTeX directives

We have the LaTeX configuration part:

```
# uses minted package instead of listings
#+LATEX_HEADER: \usepackage{minted}

# uses fonts to support Greek letters etc...
#+LATEX_HEADER: \usepackage{fontspec}
#+LATEX_HEADER: \setmonofont{DejaVu Sans Mono}[Scale=MatchLowercase]

# defines the \begin{comment} \end{comment} environment, used to avoid
# conflict between bibtex and biblatex
#+LATEX_HEADER: \usepackage{verbatim}

# uses the biblatex package (and not the old bibtex)
#+LATEX_HEADER: \usepackage[backend=biber, bibencoding=utf8 ]{biblatex}
# our bibliography file
#+LATEX_HEADER: \addbibresource{my-bib.bib}
```

We then define our the Julia code highlight style. This style is used by **minted** for PDF export.

```
#+BEGIN_EXPORT latex
\definecolor{bg}{rgb}{0.95,0.95,0.95}
\setminted[julia]{
```

```
  bgcolor=bg,
  breaklines=true,
  mathescape,
  fontsize=\footnotesize}
#+END_EXPORT
```

3.2 Your notebook

Now this is the beginning of our notebook. One can use Org as usual…

```
#+TITLE: My title
#+AUTHOR: author

* Very simple demo

#+BEGIN_SRC julia  :eval no-export :session *demo_session* :exports none
using Plots
#+END_SRC

** UTF8 support + escape math equation
Note that UTF8 is supported (the \alpha variable) :

#+BEGIN_SRC julia :eval no-export :session *demo_session* :exports both :results
↪  silent :wrap "SRC julia :eval never"
# Generate a matrix $a_{i,j}=\mathcal{U}([0,1[)$
α=rand(4,5)
#+END_SRC

** Long lines are wrapped

#+BEGIN_SRC julia :eval no-export :session *demo_session* :exports both :results
↪  output :wrap "SRC julia :eval never"
function ⊗(a::AbstractArray{T},b::AbstractArray{S}) where {T<:Number,S<:Number}
↪  kron(a,b) end;

β=rand(2,5);
γ = α ⊗ β
#+END_SRC

** Plot example

You can easily generate plots, one example from
↪  [[http://docs.juliaplots.org/latest/examples/pyplot/][Plots Julia package]],
 is used to generate Figure [[PolarPlot]].

#+BEGIN_SRC julia  :eval no-export :session *demo_session* :exports code :results
↪  silent
```

```
θ = linspace(0,1.5π,100)
r = abs(0.1 * randn(100) + sin.(30))
plot(θ,r,proj=:polar,m=2)
#+END_SRC

#+BEGIN_SRC julia  :eval no-export :session *demo_session* :results graphics :file
↪  example.png :exports results
savefig("example.png")
#+END_SRC

#+CAPTION: A polar plot.
#+ATTR_HTML: :width 900px
#+NAME: PolarPlot
#+RESULTS:
[[file:example.png]]

** Org with bibliography

\begin{align}
\label{eq:one_eq}
{\frac {d}{dt}}\iint _{\Sigma (t)}\mathbf {F} (\mathbf {r} ,t)\cdot d\mathbf {A} =
↪  & \iint _{\Sigma (t)}\left(\mathbf {F} _{t}(\mathbf {r},t)+\left[\nabla \cdot
↪  \mathbf {F} (\mathbf {r} ,t)\right]\mathbf {v}
\right)\cdot d\mathbf {A} - \\
& \oint _{\partial \Sigma (t)}\left[\mathbf{v} \times \mathbf {F} (\mathbf {r}
↪  ,t)\right]\cdot d\mathbf {s} \nonumber
\end{align}

Eq. \ref{eq:one_eq} is demonstrated in cite:Flanders1973.
```

3.3 Bibliography

Now we reach a little trick to support both HTML and PDF bibliography exports:

```
#+BEGIN_EXPORT latex
\printbibliography
#+END_EXPORT

#+BEGIN_EXPORT latex
\begin{comment}
#+END_EXPORT
#+BIBLIOGRAPHY: my-bib plain
#+BEGIN_EXPORT latex
\end{comment}
#+END_EXPORT
```

Explanation:
To export HTML bibliography, **ox-bibtex** does the job with only one directive:

```
#+BIBLIOGRAPHY: my-bib plain
```

However, for PDF export we do not want to use **ox-bibtex**, as it does not support UTF8. The solution is to wrap this directive into a comment section in the generated **.tex** code:

```
#+BEGIN_EXPORT latex
\begin{comment}
#+END_EXPORT
#+BIBLIOGRAPHY: my-bib plain
#+BEGIN_EXPORT latex
\end{comment}
#+END_EXPORT
```

Now we must tell LaTeX to use **biblatex**, this is done thanks to this directive:

```
#+BEGIN_EXPORT latex
\printbibliography
#+END_EXPORT
```

Putting everything together you get the proposed solution. This is certainly not the cleanest approach, but I have not found simpler.

4 The my-bib.bib file

For our example we need a small bibliography **my-bib.bib** file:

```
@article{Flanders1973,
  doi = {10.2307/2319163},
  url = {https://doi.org/10.2307/2319163},
  year  = {1973},
  month = {jun},
  publisher = {{JSTOR}},
  volume = {80},
  number = {6},
  pages = {615},
  author = {Harley Flanders},
  title = {Differentiation Under the Integral Sign},
  journal = {The American Mathematical Monthly}
}
```

5 Usage

You can visit the GitHub repo to reproduce the results.

5.1 Starting Emacs with the local configuration

From project root directory type

```
emacs -q --load emacs_files/init.el
```

to start a new Emacs with our local configuration.

5.2 Recomputing the notebook

As I potentially have several notebooks to publish I have used the **:eval no-export** argument. By consequence the notebooks are not evaluated each time you publish but only once. If you want to recompute everything every time, simply remove this option. You can also use the :cache option.

By consequence, before exporting you must begin by a first evaluation of the notebook. Visit the **example.org** buffer and do **M-x org-babel-execute-buffer** (or use the **C-c C-v b** shortcut). Attention, be sure that **Plots.jl** is installed.

5.2.1 ERROR: MethodError: no method matching start(::...)

In the $*\text{demo}_{\text{session}*}$ Julia session buffer you will certainly see this error:

```
ERROR: MethodError: no method matching start(::...)
```

This is not our fault, but a known problem julia-print-commands-not-working-in-emacs-org-mode. It does not affect the computed result (but only the output processing). To get the right output (without the error message) one workaround is to restart computation of the source block (**C-c C-c**).

5.3 Exporting

Still from the **example.org** buffer, you can do:

- HTML export with: **C-c C-e h o**
- PDF export with: **C-c C-e l o**

This should generate and open fresh **hmtl** and **pdf** files.

Note: concerning **html** files, this is a basic export, you can use your own HTML theme.