Computo Journal Format

To be used as template for contribution to Computo

The Computo Team a friend

10/20/22

This is the abstract - Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur posuere vestibulum facilisis. Aenean pretium orci augue, quis lobortis libero accumsan eu. Nam mollis lorem sit amet pellentesque ullamcorper. Curabitur lobortis libero eget malesuada vestibulum. Nam nec nibh massa. Pellentesque porttitor cursus tellus. Mauris urna erat, rhoncus sed faucibus sit amet, venenatis eu ipsum.

Contents

Introduction 2
- About this document 2
- Advice for writting your manuscript 2

Formatting 2
- Basic markdown formatting 2
- Mathematics 3
  - Mathematical formulae 3
  - Theorems and other amsthem-like environments 3
- Code 4
  - R 4
  - Python 4
- Figures 5
- Tables 6
- Algorithms 10
- Handling references 11
  - Bibliographic references 11
  - Other cross-references 11
- Bibliography 11

Introduction

About this document

This document provides a template based on the quarto system for contributions to **Computo** (Computo Team 2021). We show how `Python` (Perez, Granger, and Hunter 2011) or `R` (R Core Team 2020) code can be included.

Advice for writting your manuscript

First make sure that you are able to build your manuscript as a regular notebook on your system. Then you can start configure the binder environment.

Formatting

This section covers basic formatting guidelines. Quarto is a versatile formatting system for authoring HTML based on markdown, integrating LaTeX and various code block interpreted either via Jupyter or Knitr (and thus deal with Python, R and many other langages). It relies on the Pandoc Markdown markup language.

> **Note**
>
> We will only give some formatting elements. Authors can refer to the Quarto web page for a complete view of formatting possibilities.

To render/compile a document, run `quarto render`. A document will be generated that includes both content as well as the output of any embedded code chunks within the document:

```
quarto render content.qmd # will render to html
```

Basic markdown formatting

Bold text or *italic*

- This is a list
- With more elements
- It isn't numbered.

But we can also do a numbered list

1. This is my first item
2. This is my second item
3. This is my third item

Mathematics

Mathematical formulae

$\LaTeX$ code is natively supported[1], which makes it possible to use mathematical formulae:

will render

$$f(x_1, \dots, x_n; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2\right)$$

It is also posible to cross-reference an equation, see Equation 1:

$$\begin{aligned} D_{x_N} &= \frac{1}{2}\left[\begin{array}{cc} x_L^\top & x_N^\top \end{array}\right] \left[\begin{array}{cc} L_L & B \\ B^\top & L_N \end{array}\right] \left[\begin{array}{c} x_L \\ x_N \end{array}\right] \\ &= \frac{1}{2}(x_L^\top L_L x_L + 2x_N^\top B^\top x_L + x_N^\top L_N x_N), \end{aligned} \tag{1}$$

Theorems and other amsthem-like environments

Quarto includes a nice support for theorems, with predefined prefix labels for theorems, lemmas, proposition, etc. see this page. Here is a simple example:

Theorem 0.1 (Strong law of large numbers)**.** *The sample average converges almost surely to the expected value:*

$$\overline{X}_n \xrightarrow{a.s.} \mu \qquad \textit{when } n \to \infty.$$

See Theorem 0.1.

[1]We use katex for this purpose.

Code

Quarto uses either Jupyter or knitr to render code chunks. This can be triggered in the yaml header, e.g., for Jupyter (should be installed on your computer) use

```
---
title: "My Document"
author "Jane Doe"
jupyter: python3
---
```

For knitr (R + knitr must be installed on your computer)

```
---
title: "My Document"
author "Jane Doe"
---
```

You can use Jupyter for Python code and more. And R + KnitR for if you want to mix R with Python (via the package reticulate Ushey, Allaire, and Tang (2020)).

R

`R` code (R Core Team 2020) chunks may be embedded as follows:

```r
x <- rnorm(10)
```

Python

```python
import plotly.express as px
df = px.data.tips()
fig = px.histogram(df, x="total_bill", y="tip", color="sex",
                   marginal="box", # or violin, rug
                   hover_data=df.columns)
fig
```

Figure 1: A simple python plotly example

Figures

Plots can be generated as follows and referenced. See plot Figure 2:

```r
library("ggplot2")
p <- ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  geom_smooth() + theme_bw()
p
```

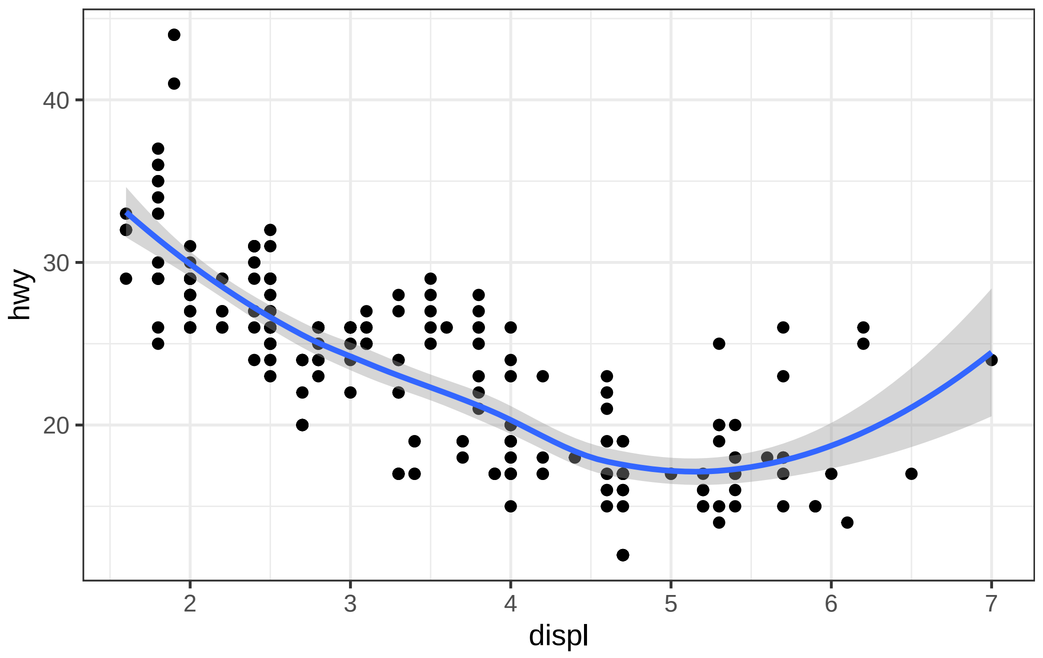

Figure 2: A simple ggplot example

Interactive plots may also be produced in the HTML output of the document:

It is also possible to create figures from static images:

Figure 3: SFdS logo (c.a. 2021)

Tables

Tables (with label: `@tbl-mylabel` renders Table 1) can be generated with markdown as follows

Table 1: my table caption

Tables	Are	Cool
col 1 is	left-aligned	$1600
col 2 is	centered	$12
col 3 is	right-aligned	$1

Table can also be generated by some code, for instance with knitr here:

```
knitr::kable(summary(cars), caption = "Table caption.")
```

Table 2: Table caption.

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

```
ip <- as.data.frame(installed.packages()[,c(1,3:4)])
rownames(ip) <- NULL
ip <- ip[is.na(ip$Priority),1:2,drop=FALSE]
print(ip, row.names=FALSE)
```

```
    Package Version
    askpass     1.1
 assertthat   0.2.1
  backports   1.4.1
  base64enc   0.1-3
        bit   4.0.4
      bit64   4.0.5
       blob   1.2.3
```

```
         brio   1.1.3
        broom   1.0.1
        bslib   0.4.0
       cachem   1.0.6
        callr   3.7.2
   cellranger   1.1.0
          cli   3.4.1
        clipr   0.8.0
   colorspace   2.0-3
        cpp11   0.4.3
       crayon   1.5.2
    crosstalk   1.2.0
         curl   4.3.3
   data.table  1.14.4
          DBI   1.1.3
       dbplyr   2.2.1
         desc   1.4.2
      diffobj   0.3.5
       digest  0.6.30
        dplyr  1.0.10
       dtplyr   1.2.2
     ellipsis   0.3.2
     evaluate    0.17
        fansi   1.0.3
       farver   2.1.1
      fastmap   1.1.0
      forcats   0.5.2
           fs   1.5.2
       gargle   1.2.1
     generics   0.1.3
      ggplot2   3.3.6
         glue   1.6.2
  googledrive   2.0.0
googlesheets4   1.0.1
       gtable   0.3.1
        haven   2.5.1
         here   1.0.1
       hexbin  1.28.2
        highr     0.9
          hms   1.1.2
    htmltools   0.5.3
  htmlwidgets   1.5.4
         httr   1.4.4
         ids   1.0.1
     isoband   0.2.6
   jquerylib   0.1.4
    jsonlite   1.8.2
       knitr    1.40
    labeling   0.4.2
       later   1.2.0
    lazyeval   0.2.2
   lifecycle   1.0.3
   lubridate   1.8.0
    magrittr   2.0.3
     memoise   2.0.1
        mime    0.12
      modelr   0.1.9
     munsell   0.5.0
     openssl   2.0.4
      pillar   1.8.1
   pkgconfig   2.0.3
     pkgload   1.3.0
      plotly  4.10.0
         png   0.1-7
      praise   1.0.0
 prettyunits   1.1.1
    processx   3.7.0
    progress   1.2.2
    promises 1.2.0.1
          ps   1.7.1
       purrr   0.3.5
          R6   2.5.1
    rappdirs   0.3.3
RColorBrewer   1.1-3
        Rcpp   1.0.9
    RcppTOML   0.1.7
       readr   2.1.3
      readxl   1.4.1
     rematch   1.0.1
    rematch2   2.1.2
      reprex   2.0.2
  reticulate    1.26
       rlang   1.0.6
   rmarkdown    2.17
   rprojroot   2.0.3
  rstudioapi    0.14
```

rvest	1.0.3
sass	0.4.2
scales	1.2.1
selectr	0.4-2
stringi	1.7.8
stringr	1.4.1
sys	3.4.1
testthat	3.1.5
tibble	3.1.8
tidyr	1.2.1
tidyselect	1.2.0
tidyverse	1.3.2
tinytex	0.42
tzdb	0.3.0
utf8	1.2.2
uuid	1.1-0
vctrs	0.4.2
viridisLite	0.4.1
vroom	1.6.0
waldo	0.4.0
withr	2.5.0
xfun	0.34
xml2	1.3.3
yaml	2.3.6

Note

Alternatively you may use the list tables format, easier to write and maintain:

```
:::list-table
   * - row 1, column 1
     - row 1, column 2
     - row 1, column 3

   * - row 2, column 1
     -
     - row 2, column 3

   * - row 3, column 1
     - row 3, column 2
:::
```

row 1, column 1	row 1, column 2	row 1, column 3
row 2, column 1		row 2, column 3
row 3, column 1	row 3, column 2	

Algorithms

A solution to typeset pseudocode just like you would do with LaTeX, yet with HTML output is to rely on the JavaScript pseudocode.js. Your pseudocode is written inside a Code Block with the `pseudocode` class. Do not forget the class tag, that will trigger the rendering process of your pseudo-code. The result is as follows[2]:

~~~
```{.pseudocode}
% This quicksort algorithm is extracted from Chapter 7, Introduction
% to Algorithms (3rd edition)
\begin{algorithm}
\caption{Quicksort}
\begin{algorithmic}
\Procedure{Quicksort}{$A, p, r$}
    \If{$p < r$}
        \State $q = $ \Call{Partition}{$A, p, r$}
        \State \Call{Quicksort}{$A, p, q - 1$}
        \State \Call{Quicksort}{$A, q + 1, r$}
    \EndIf
\EndProcedure
\Procedure{Partition}{$A, p, r$}
    \State $x = A[r]$
    \State $i = p - 1$
    \For{$j = p, \dots, r - 1$}
        \If{$A[j] < x$}
            \State $i = i + 1$
            \State exchange
            $A[i]$ with $A[j]$
        \EndIf
        \State exchange $A[i]$ with $A[r]$
    \EndFor
\EndProcedure
\end{algorithmic}
~~~

[2]For proper pdf rendering, use Camel cased names for all `algorithmic` keywords, not upper case ones like the examples in `pseudocode.js`'s documentation

```
\end{algorithm}
```
```

Algorithm 1 Quicksort

```
procedure QUICKSORT($A, p, r$)
    if $p < r$ then
        $q = $ PARTITION($A, p, r$)
        QUICKSORT($A, p, q - 1$)
        QUICKSORT($A, q + 1, r$)
procedure PARTITION($A, p, r$)
    $x = A[r]$
    $i = p - 1$
    for $j = p, \ldots, r - 1$ do
        if $A[j] < x$ then
            $i = i + 1$
            exchange $A[i]$ with $A[j]$
        exchange $A[i]$ with $A[r]$
```

Handling references

Bibliographic references

References are displayed as footnotes using BibTeX, e.g. `[@computo]` will be displayed as (Computo Team 2021), where `computo` is the bibtex key for this specific entry. The bibliographic information is automatically retrieved from the `.bib` file specified in the header of this document (here:`references.bib`).

Other cross-references

As already (partially) seen, Quarto includes a mecanism similar to the bibliographic references for sections, equations, theorems, figures, lists, etc. Have a look at this page.

Bibliography

Computo Team. 2021. “Computo: Reproducible Computational/Algorithmic Contributions in Statistics and Machine Learning.” *Computo.*

Perez, Fernando, Brian E Granger, and John D Hunter. 2011. “Python: An Ecosystem for Scientific Computing.” *Computing in Science & Engineering* 13 (2): 13–21.
```

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Ushey, Kevin, JJ Allaire, and Yuan Tang. 2020. *Reticulate: Interface to Python*. https://github.com/rstudio/reticulate.