Notes template

Tadej Svetina[a]
MIT

June 2019

1 Introduction

This is a sample document using the custom `notes` class (with the `macros` package). The primary aim of the `notes` class is to move all the boilerplate usually found in the preamble of the `main.tex` file to a reusable class file, making the actual document code shorter and easier to read.

Complementing that, the secondary aim of the `notes` class and the `macros` package is to create short commands for some frequently used things, cutting down on typing/tinkering time and improving readability.

Finally, as we have a reusable class and package, we might as well define some nice looking defaults. For this purpose, the class/package define (or show)

- a custom color palette, based on Paul Tol's vibrant scheme
- nice fancy theorem environments (optional),

I showcase all of the above mentioned features in this template, and as a bonus, also how to get easy code highlighting in LaTeX, how to typset nice tables with `booktabs` and how to adjust fonts and colors for python graphs to match the document.

Finally, to use the class/package combo, just put both files in the root project repository, and add the following line to the top of the `main.tex` file

```
\documentclass{notes}
```

Nothing else is needed!

2 Class options

The `notes` class defines the following class-specific options:

- `letterpaper`: Sets the paper size to letter paper, with 1 inch margins. This is the default.
- `a4paper`: Sets the paper size to A4 paper, with 1 inch margins.
- `a5paper`: Sets the paper size to A5 paper, with 0.75 inch margins.
- `b5paper`: Sets the paper size to B5 paper, with 0.75 inch margins.

[a]`tadej@mit.edu`

- `nofancy`: This removes the fancy formatting from the theorem environments, giving them the standard `amsthm` formatting, see section 4.5.

Any other options passed to `notes` class will simply be passed to the `article` class, on which `notes` builds.

3 Colors

The color palette defines the following colors (which are obtained by `m` + capitalized name of the color):

- **blue, cyan, teal, orange, red, magneta,** grey

There are two commands that use these colors: `\alert` and `\define`.

4 Theorems

The only special custom thing are the theorems, which are (by default) made "fancy" with the `tcolorbox` environment. This can be turned off by passing `nofancy` option to the document class. The following theorem-environments are supported:

- Theorem (`thm`)
- Lemma (`lem`)
- Example (`exm`)
- Definition (`definition`)

Additionally, `proof` also has a fancy version with a colorful frame, but it is not numbered.

4.1 Theorem

The following example shows the basic syntax of the `thm` command:

```
\begin{thm}{Cauchy-Schwarz inequality}{cs}
    For all vectors $\vec{u}$ and $\vec{v}$ in ...
\end{thm}
```

Output of this command is:

Theorem 4.1: Cauchy-Bunyakovsky-Schwarz inequality

For all vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ in an inner product space it is true that

$$|\langle \boldsymbol{u}, \boldsymbol{v}\rangle|^2 \leq \langle \boldsymbol{u}, \boldsymbol{u}\rangle \cdot \langle \boldsymbol{v}, \boldsymbol{v}\rangle \quad (1)$$

4.2 Lemma

The following example shows the basic syntax of the `lem` command:

```
\begin{lem}{Titu's Lemma}{titu}
    For some $u_i, v_i \in \Rnn$, it is true that ...
\end{lem}
```

Output of this command is:

Lemma 4.1: Titu's Lemma

For some $u_i, v_i \in \mathbb{R}_{\geq 0}$, it is true that

$$\frac{\left(\sum_{i=1}^n u_i\right)^2}{\sum_{i=1}^n v_i} \leq \sum_{i=1}^n \frac{u_i^2}{v_i} \tag{2}$$

4.3 Proof

The following example shows the basic syntax of the `proof` command, as well as how the references to the above Lemma and Theorem work.

```
\begin{proof}
    To prove Lemma \ref{lem:titu}, we just need to replace $u_i' =
     ↪  \frac{u_i}{\sqrt{v_i}}$ and $v_i' = \sqrt{v_i}$, and then apply
     ↪  Theorem \ref{thm:cs}.
\end{proof}
```

The output is:

Proof. To prove Lemma 4.1, we just need to replace $u_i' = \frac{u_i}{\sqrt{v_i}}$ and $v_i' = \sqrt{v_i}$, and then apply Theorem 4.1. ☐

4.4 Example and definition

The syntax is basically the same as for Lemma and Proof, so I omit the code. Note the use of `define` command in the definition.

Definition 4.1

We say that a set A is **closed**, if its complement A^C is open.

Example 4.1

In the discrete topology, the closed sets are X and $\emptyset$.

4.5 `nofancy` theorems

Here we show how the previous examples in this section would have looked like if we passed the `nofancy` option to the document class. The code syntax would have remained the same, of course.

Theorem 4.1 (Cauchy-Schwarz inequality)**.** *For all vectors* $\boldsymbol{u}$ *and* $\boldsymbol{v}$ *in an inner product space it is true that*

$$|\langle \boldsymbol{u}, \boldsymbol{v}\rangle|^2 = \langle \boldsymbol{u}, \boldsymbol{u}\rangle \cdot \langle \boldsymbol{v}, \boldsymbol{v}\rangle \tag{3}$$

Lemma 4.1 (Titu's Lemma)**.** *For some* $u_i, v_i \in \mathbb{R}_{\geq 0}$*, it is true that*

$$\frac{\left(\sum_{i=1}^n u_i\right)^2}{\sum_{i=1}^n v_i} \leq \sum_{i=1}^n \frac{u_i^2}{v_i} \tag{4}$$

Proof. To prove Lemma 4.1, we just need to replace $u_i' = \frac{u_i}{\sqrt{v_i}}$ and $v_i' = \sqrt{v_i}$, and then apply Theorem 4.1. □

Definition 4.1. We say that a set A is **closed**, if its complement A^C is open.

Example 4.1. In the discrete topology, the closed sets are X and $\emptyset$.

5 Tables

The example bellow shows how to make a simple table using the `booktabs` package:

Item		
Animal	Description	Price ($)
Gnat	per gram	13.65
	each	0.01
Gnu	stuffed	92.50
Emu	stuffed	33.33
Armadillo	frozen	8.99

Table 1: Price list

The code used to produce this example is

```latex
\begin{table}[H]
    \centering

    \begin{tabular}{@{}llr@{}}
        \toprule
        \multicolumn{2}{c}{Item} \\
        \cmidrule(r){1-2}
        Animal    & Description & Price (\$) \\
        \midrule
        Gnat      & per gram    & 13.65      \\
                  &    each     & 0.01       \\
        Gnu       & stuffed     & 92.50      \\
        Emu       & stuffed     & 33.33      \\
        Armadillo & frozen      & 8.99       \\
        \bottomrule
    \end{tabular}

    \caption{Price list}
    \label{tab:my_label}
\end{table}
```

6 Code

This isn't anything special to the `notes` class, but I just show how to write beautifully formatted Python code for reference. Here is how some python code looks like:

```python
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

We can make use of code separation and import the code to be displayed from a separate `.py` file, indeed, the previous example was produced by (`linenos` is used to add line numbers)

```
\inputminted[linenos]{python}{code.py}
```

and the line of code above was produced with[1]

[1] In the example we wrote `m inted` instead of `minted`, as otherwise `\end{minted}` is not properly escaped.

```
\begin{m inted}{latex}
    \inputminted{python}{code.py}
\end{m inted}
```

7 Python graphs

When trying to match the style of python (matplotlib) graphs, two main adjustments have to be considered: fonts and colors.

7.1 Text width

But even before that, another "adjustment" should be made: the dimensions of the figure should be set to be smaller than those of the document. The relevant dimension here is width, or to be precise `\textwidth`. But the pure LaTeX ways of getting this give us the size in points (and we need it in inches), so to get it we use the `printlen` package:

```
\uselengthunit{in}\printlength{\textwidth}
```

As an example, the text width of this document is 6.26894 in.

7.2 Fonts

We need to make sure that the fonts in the document match the fonts in the graph. Matplotlib has a limited selection of fonts, so its safest to let LaTeX do all the font processing. We need to load the fonts and change the size to match that of our document. We would do this by putting this at the beginning of our code

```
from matplotlib import rc

rc('text.latex',preamble=r'\usepackage[charter, cal=cmcal]{mathdesign}')
rc('text',usetex=True)
rc('font', family='serif', size=11)
```

Note that this will slow down processing quite a bit, so **only change the fonts when the figures are done!**

7.3 Colors

We might also want to use our standard color palette (see section 3) – although this is more a matter of taste, there is nothing wrong with matplotlib color palettes. The way we would implement them is with

```python
from matplotlib import rc
from cycler import cycler

colors = ['#0077BB', '#EE7733', '#009988', '#CC3311', '#33BBEE',
 ↪ '#EE3377']
rc('axes', prop_cycle = cycler(c=colors))
```

7.4 An example

Let's put the two things together, to create a simple example. Here is the code:

```python
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
from cycler import cycler

# Set up fonts
rc('text.latex',preamble=r'\usepackage[charter, cal=cmcal]{mathdesign}')
rc('text',usetex=True)
rc('font', family='serif', size=11)

# Set up colors
colors = ['#0077BB', '#EE7733', '#009988', '#CC3311', '#33BBEE',
 ↪ '#EE3377']
rc('axes', prop_cycle = cycler(c=colors))

# Plot
x = np.linspace(-np.pi, np.pi, int(1e3))
fig, ax = plt.subplots(figsize=(6,4))
ax.plot(x, np.sin(x), label=rf'$\sin(x)$')

for i in range(1,6):
    ax.plot(x, np.sin(x + i*np.pi/6),
            label=r'$\sin\left(x+ \frac{' + str(i) + r'}{6}\pi\right)$')

# Label graph
ax.set_title('Some trig functions', fontsize=11)
ax.set_xlabel(r'$x$')
ax.set_ylabel(r'$f(x)$')

# Tweak legend and layout
plt.legend(loc='upper left', bbox_to_anchor=(1.05, 1), borderaxespad=0.)
plt.tight_layout()

# Save figure
fig.savefig('../../Dropbox (MIT)/Apps/Overleaf/Templates/Notes/trig.pdf',
            pad_inches=0, bbox_inches='tight')
```

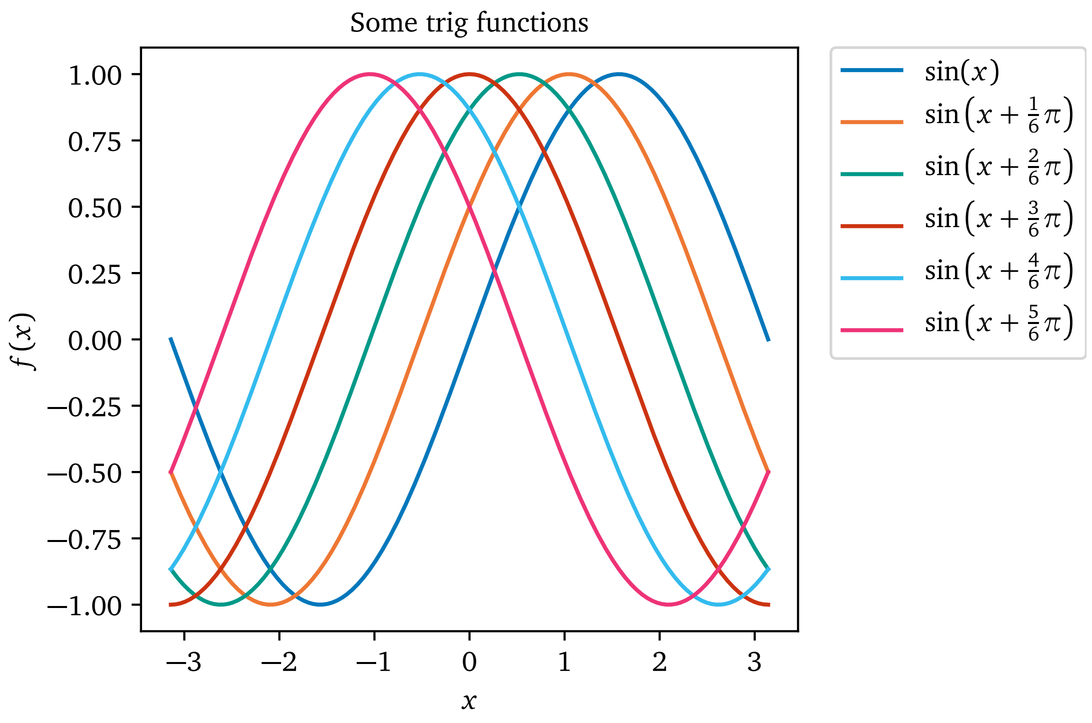

Figure 1: Some trigonometric functions

The result is shown in Figure 1. To completely visually merge the figure in the document we would probably need to tweak a few more things (for example the legend box), but as far as the fonts and colors are concerned, we are done. Note that we set the width of the figure to 6 inches – that's ok, as we have seen that the width of our document is 6.26894 in.

There are also a few extra useful tweaks in the code:

- The title size is set, in line 25, to the document size (11) – this is because by default, the title size is a bit bigger (other graph elements do not have this problem). But this is most likely not relevant anyways, as you would not need a title on a figure that will already have a caption.
- The path in `savefig` (line 34) is set to the Overleaf Dropbox repository. This is a great way to get continuous updating with Overleaf every time you change the figure in the code, without having to manually copy/paste or git commit/push.

8 Todo

There are a couple of things that I would like to add to this class (package), including

- Macros for some commonly used list variations from the `enumitem` package
- Macros for multiline equations (`IEEEtrantools` based)
- Explore `captionof` package in conjunction with `tcolorbox` to include figures in theorem environments
- Tweak some spacing (after `minted`, between list items)