Skeleton Tutorial Template

TJ McKinley

Contents

Chapter 1

Opening page

Basically a standard Bookdown template with a few tweaks. New chapters need to be in separate '.Rmd' files, where each file starts with a chapter heading as seen here. In order to use the task and solution blocks in LaTeX, you must input the order of the files into the `_bookdown.yml` file, and the first file must be called `index.Rmd` e.g.

```
rmd_files:
    html: ['index.Rmd', 'ch1.Rmd']
    latex: ['index.Rmd', 'ch1.Rmd', 'ch_appendix.Rmd']
output_dir: "docs"
```

The `latex:` path above ***must*** have `'ch_appendix.Rmd'` as its last entry. This ensures that the appendix is properly formatted for the solutions to the problems.

You must have the following lines at the **start** of your `index.Rmd` file (please do not change the chunk options for this chunk, and place it as the **first** chunk in the document):

~~~
```{r, child = "_setup.Rmd", include = FALSE, purl = FALSE, cache = FALSE}
```
~~~

> **Note**: the `task` / `solution` and `info` blocks detailed below cannot be cached. Code chunks within these blocks can be cached as usual. The caching is automatically disabled in order to make sure the PDF outputs are correctly cross-referenced.

There are a couple of useful special blocks. A `task` block, and a `solution` block. These can be used as e.g.

~~~
```{task}
Here is a task written in **markdown**.
```
~~~

which renders as:

Task 1

Here is a task written in **markdown**.

You can include chunks within the `task` chunk, but you need to use double backticks *within* the chunk, and leave carriage returns around the internal chunk e.g.

~~~~
```{task}

``{r}
x <- 2 + 2
x
``

```
~~~~

which renders as:

Task 2

```r
x <- 2 + 2
x
```

```
## [1] 4
```

Be careful to have suitable carriage returns around e.g. `enumerate` or `itemize` environments inside the chunk also. For example:

~~~~
```{task}
Here is a list:
1. item 1
2. item 2
```
~~~~

will not render nicely. But

~~~~
```{task}
Here is a list:

1. item 1
2. item 2

```
~~~~

will:

Task 3

Here is a list:

1. item 1
2. item 2

The `solution` chunk works in the same way, and the numbers will follow the previous `task` chunk (so you can set tasks without solutions) e.g.

~~~~
```{task}
Add 2 and 2 together
```
~~~~

~~~
```{solution}

``{r}
2 + 2
``

```
~~~

gives:

Task 4

Add 2 and 2 together

Show: Solution on P15

1.1 Additional extensions

1.1.1 Different task and solution titles

Task and solution boxes can also be given different names using the `title` option e.g.

~~~
```{task, title = "Question"}
What is the meaning of life, the universe and everything?
```

```{solution, title = "Answer"}
Why 42 of course!
```
~~~

gives:

Question 5

What is the meaning of life, the universe and everything?

Show: Answer on P15

1.1.2 Turning tasks and solutions on and off

Sometimes you might want to hide task and/or solution boxes. This can be done with the `renderTask` and `renderSol` chunk options, which can be set globally or locally. For example:

~~~
```{task, title = "Question"}
Can I set a task and not show the answer?
```

```{solution, title = "Answer", renderSol = FALSE}
Indeed, though you won't see this answer unless `renderSol = TRUE`...
```
~~~

typesets as:

Question 6

Can I set a task and not show the answer?

1.1.3 Generic information environments

You can also set generic boxed environments containing arbitrary information.

~~~
```{info, title = "Some interesting titbit"}
This box contains invaluable information!
```
~~~

typesets as:

Show: Some interesting titbit on P17

Note that it is useful to set the `title` option here, else it defaults to `info`. You can also use this environment to simply display an alert box with information, by setting the `collapsible` argument to `FALSE` in the chunk options e.g.

~~~
```{info, title = "Some interesting aside", collapsible = FALSE}
Yet more valuable information - this time displayed directly!
```
~~~

typesets as:

Some interesting aside

Yet more valuable information - this time displayed directly!

In the PDF output, setting `collapsible = TRUE` will place the information boxes in a separate Appendix, with links in the main document. You can again hide the `info` boxes by setting `renderInfo = FALSE` in the chunk options.

You can also put boxes around text without any titles by setting `title = NA` and `collapsible = FALSE` e.g.

~~~
```{info, title = NA, collapsible = FALSE}
Just some stuff but no titles!
```
~~~

Just some stuff but no titles!

1.1.4 Tabbed boxed environments

Originally developed to put base R and `tidyverse` solutions side-by-side, using a `multCode = TRUE` option to the solution box. Here the two tabs are separated by four consecutive hashes: `####`, and the `titles` option gives the tab titles (these can be set globally if preferred) e.g.

~~~
```{task}
Filter the `iris` data by `Species == "setosa"` and find the mean `Petal.Length`.
```

```{solution, multCode = TRUE, titles = c("Base R", "tidyverse")}

``{r}
## base R solution
mean(iris$Petal.Length[
    iris$Species == "setosa"])
``

####

``{r}
## tidyverse solution
iris %>%
    filter(Species == "setosa") %>%
    select(Petal.Length) %>%
    summarise(mean = mean(Petal.Length))
``

```
~~~

will typeset to:

Task 7

Filter the `iris` data by `Species == "setosa"` and find the mean `Petal.Length`.

Show: Solution on P15

Note that there is also a `multCode` chunk that does not link to task and solution boxes e.g.

~~~
```{multCode}

Two options:

* Option 1

####

Two options:

* Option 2

```
~~~

will typeset to:

Option 1

Two options:

- Option 1

Option 2

Two options:

- Option 2

The `titles` option can be set as before.

1.1.5 Resize code chunks

Code chunks can be resized using a `size`, `htmlsize` or `latexsize` chunk option. These take either HTML or LaTeXsizes and convert accordingly. For example,

~~~
```{r, size = "scriptsize"}
rnorm(10, 0, 1)
```
~~~

typesets as:

```
rnorm(10, 0, 1)
```

```
##  [1] -0.11032580 -0.02240588 -1.31424225  3.30336349  0.69458163 -0.64004619
##  [7]  0.36418021  0.08013076  0.38607532 -0.79358265
```

Setting `htmlsize` or `latexsize` overwrites the `size` argument and allows for sizes to be different for HTML or LaTeXoutput respectively. This is most useful for `multCode` chunks that might need shrinking in a LaTeXenvironment in order to fit nicely on a page side-by-side.

1.1.6 References

References can be added in the usual way e.g. `@somepaper:2000` using BibTeX files. Typesets as author (2000). These get added to each chapter

1.1.7 Hypertargets

You can link to arbitrary parts of the document by adding `hypertarget` blocks at the point in the text where you want the target, making sure to set a `label` argument set to a unique ID tag e.g. I added

~~~
```{hypertarget, label = "idtag"}
```
~~~

just before the interesting aside earlier, and then you can use e.g. `[click to go to target](#idtag)` in the text as follows: click to go to target.

References

author, Some. 2000. “Some Title.” *Some Journal* 1: 1–20.

Appendices

Answers

Solution 4

```
2 + 2
```

```
## [1] 4
```

Return to task on P7

Answer 5

Why 42 of course!

Return to task on P7

Solution 7

Base R

```
## base R solution
mean(iris$Petal.Length[
    iris$Species == "setosa"])
```

```
## [1] 1.462
```

tidyverse

```
## tidyverse solution
iris %>%
    filter(Species == "setosa") %>%
    select(Petal.Length) %>%
    summarise(mean = mean(Petal.Length))
```

```
##    mean
## 1 1.462
```

Return to task on P9

Additional information

Some interesting titbit

This box contains invaluable information!

Return to P8