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](https://bookdown.org/yihui/bookdown/usage.html). 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:

````

```{r, child = "\_setup.Rmd", include = F, purl = F, cache = F}`r ''`

```

````

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

````

```{task}`r ''`

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

```

````

which renders as:

```{task}

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 ''`

``{r}

x <- 2 + 2

x

``

```

````

which renders as:

```{task}

``{r}

x <- 2 + 2

x

``

```

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

````

```{task}`r ''`

Here is a list:

1. item 1

2. item 2

```

````

will not render nicely. But

````

```{task}`r ''`

Here is a list:

1. item 1

2. item 2

```

````

will:

```{task}

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}`r ''`

Add 2 and 2 together

```

```{solution}`r ''`

``{r}

2 + 2

``

```

````

gives:

```{task}

Add 2 and 2 together

```

```{solution}

``{r}

2 + 2

``

```

## Additional extensions

### Different task and solution titles

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

````

```{task, title = "Question"}`r ''`

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

```

```{solution, title = "Answer"}`r ''`

Why 42 of course!

```

````

gives:

```{task, title = "Question"}

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

```

```{solution, title = "Answer"}

Why 42 of course!

```

### 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"}`r ''`

Can I set a task and not show the answer?

```

```{solution, title = "Answer", renderSol = FALSE}`r ''`

Indeed, though you won't see this answer unless `renderSol = TRUE`...

```

````

typesets as:

```{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`...

```

### Generic information environments

You can also set generic boxed environments containing arbitrary information.

````

```{info, title = "Some interesting titbit"}`r ''`

This box contains invaluable information!

```

````

typesets as:

```{info, title = "Some interesting titbit"}

This box contains invaluable information!

```

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}`r ''`

Yet more valuable information - this time displayed directly!

```

````

typesets as:

```{info, title = "Some interesting aside", collapsible = FALSE}

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.

### Tabbed boxed environments

Originally developed to put base R and `tidyverse` solutions side-by-side, using a `multCode = T` 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.

```{r, include = F}

library(tidyverse)

```

````

```{task}`r ''`

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

```

```{solution, multCode = T, titles = c("Base R", "tidyverse")}`r ''`

``{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}

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

```

```{solution, multCode = T, 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))

``

```

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

````

```{multCode}`r ''`

Two options:

\* Option 1

####

Two options:

\* Option 2

```

````

will typeset to:

```{multCode}

Two options:

\* Option 1

####

Two options:

\* Option 2

```

The `titles` option can be set as before.

-->