UNSW SYDNEY

CENTRE FOR BIG DATA RESEARCH IN HEALTH

HDAT9800 Visualisation and Communication of Health Data

Chapter 2 - Literate programming - part A

Drs James Farrow & Tim Churches

Literate programming

Literate programming is a paradigm introduced by Donald Knuth in which a program is given as an explanation of the program logic in a natural language (such as English), interleaved with sections of computer program code, from which a compilable source code can be generated — on the assumption, at that time, that the program would be used to create executable software that runs independently of its original source code.

Here we will be using a variant of that idea where a document with the output of the executed code embedded in it is produced as the final artefact rather than executable software to be used elsewhere or by others.

Literate programming

Let us change our traditional attitude to the construction of programs: Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to humans what we want the computer to do.

— Donald E. Knuth, Literate Programming, 1984

Literate programming

Thus, literate programming in modern data science means the use of a framework that lets us write documents which are ***both***:

- programmes — they can perform calculations and analyses
- reports, scientific papers, blog posts, web pages, presentation slide decks, tutorials etc — anything readable by humans

Mixing the two together means that when we finish or update one, when we finish or update the other.

The report *is* the analysis, and the analysis is the report. This dramatically improves reproducibility and verification.

Literate programming

What is literate programming

How will it help me?

How will it help others?

Literate programming using R markdown in RStudio

Literate programming as a communication tool

Help us clarify our intent to others, and to our future selves

- having the explanation and the code together helps both

Helps transparency

- others can ‘check our working’

Helps reproducibility

- our document is not just static text but can be executed

Literate programming in the R software ecosystem

R and RStudio arguably provide better support for literate programming paradigm than any other data analysis environment

The *knitr* package started life as means to bring literate programming to static document production

Since then it has been used to enable creation of presentation slide decks, blog posts, whole web sites, dashboards, and fully-fledged interactive applications via *shiny*.

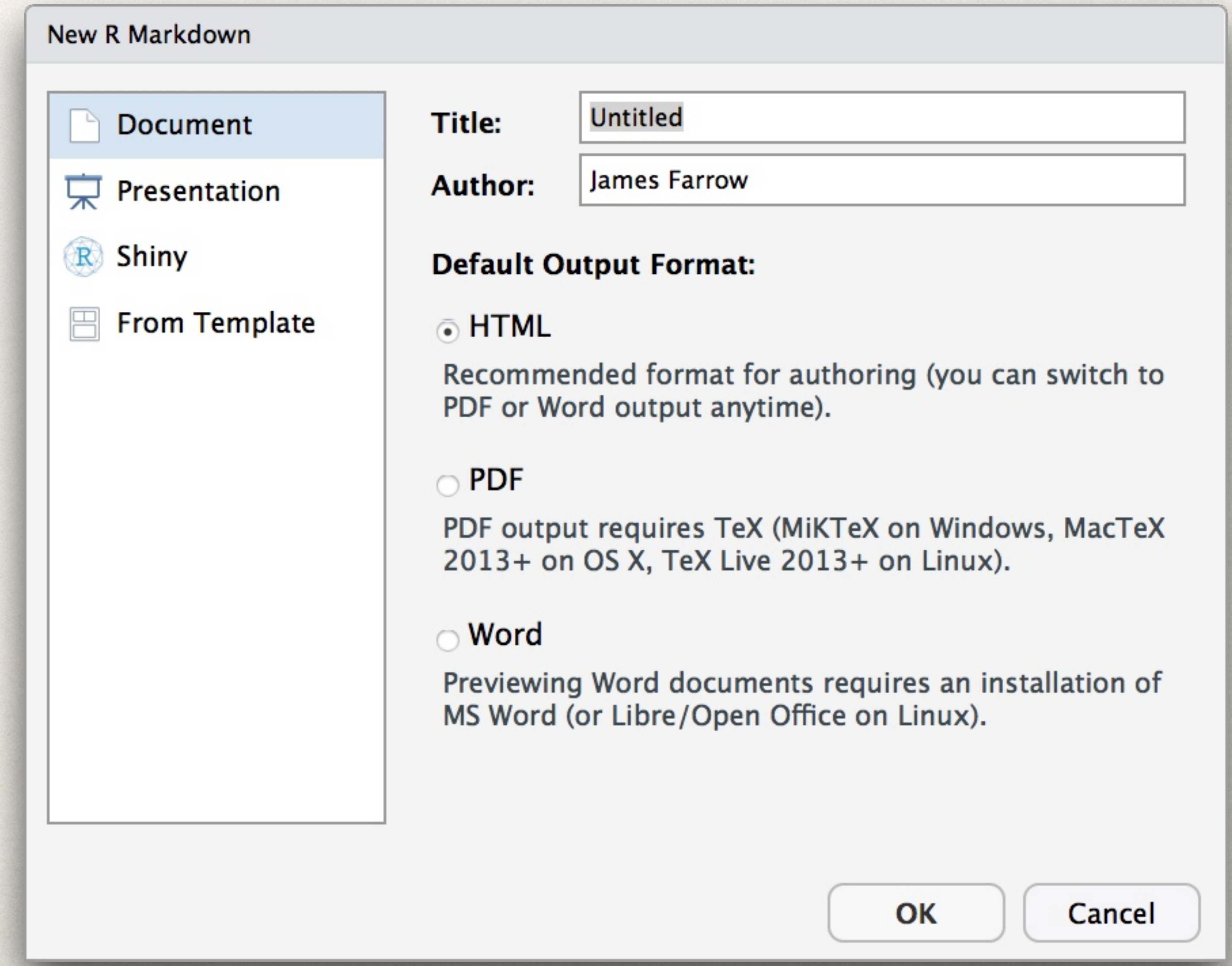
Literate programming in the R software ecosystem

RStudio has integrated facilities for editing *Rmarkdown* (.Rmd) files

The *File >> New File* menu has an option for *R markdown…*

Different output formats

- HTML, PDF, Word, $\LaTeX$
- Slides
- Interactive documents


New R Markdown

Title: Untitled

Author: James Farrow

Default Output Format:

[✓ HTML]

Recommended format for authoring (you can switch to PDF or Word output anytime).

PDF

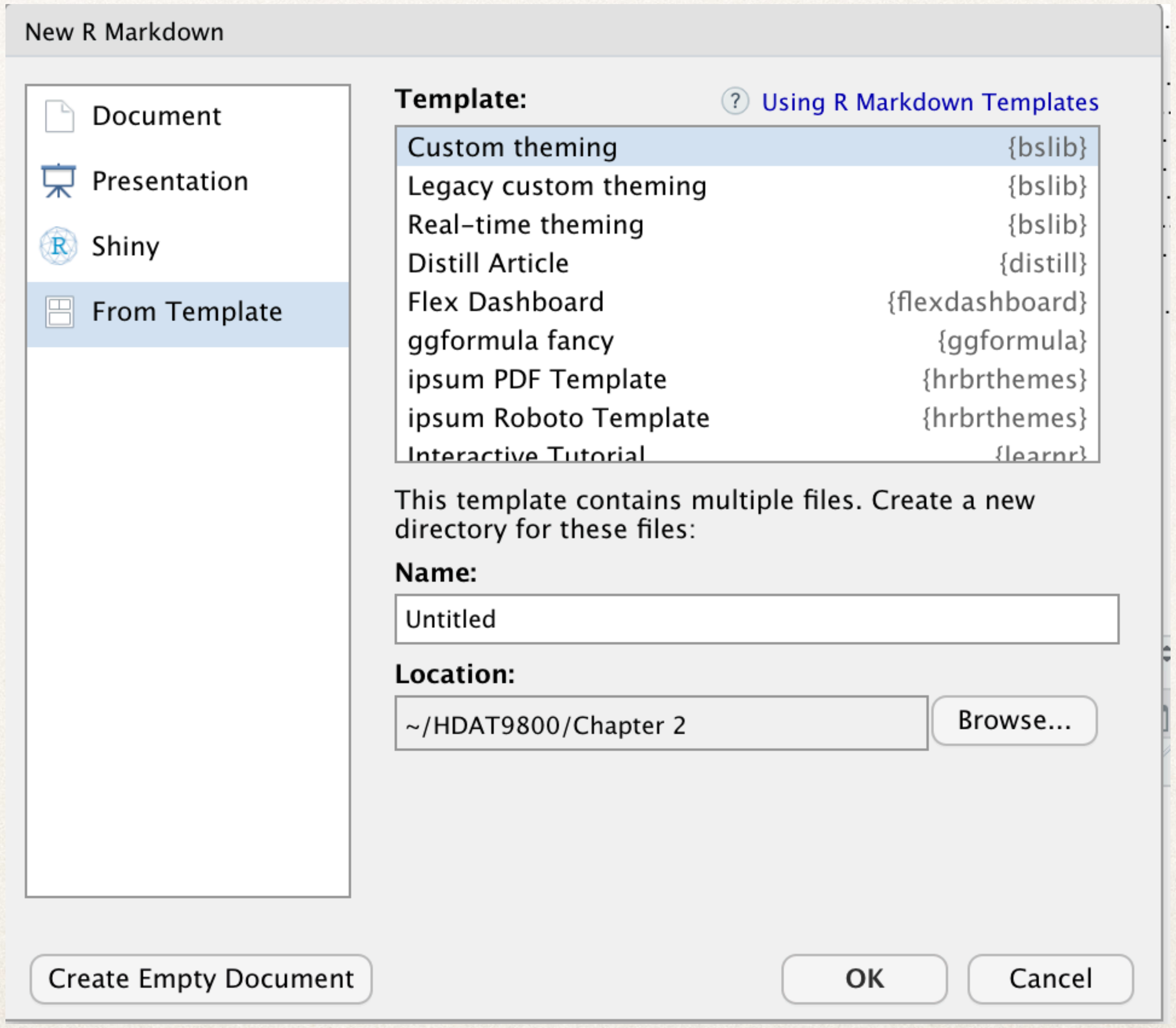
PDF output requires TeX (MiKTeX on Windows, MacTeX 2013+ on OS X, TeX Live 2013+ on Linux).

Word

Previewing Word documents requires an installation of MS Word (or Libre/Open Office on Linux).

R Markdown templates

Installable packages of additional R Markdown document templates


New R Markdown

Template: Using R Markdown Templates

Custom theming	{bslib}
Legacy custom theming	{bslib}
Real-time theming	{bslib}
Distill Article	{distill}
Flex Dashboard	{flexdashboard}
ggformula fancy	{ggformula}
ipsum PDF Template	{hrbrthemes}
ipsum Roboto Template	{hrbrthemes}
Interactive Tutorial	{learnr}

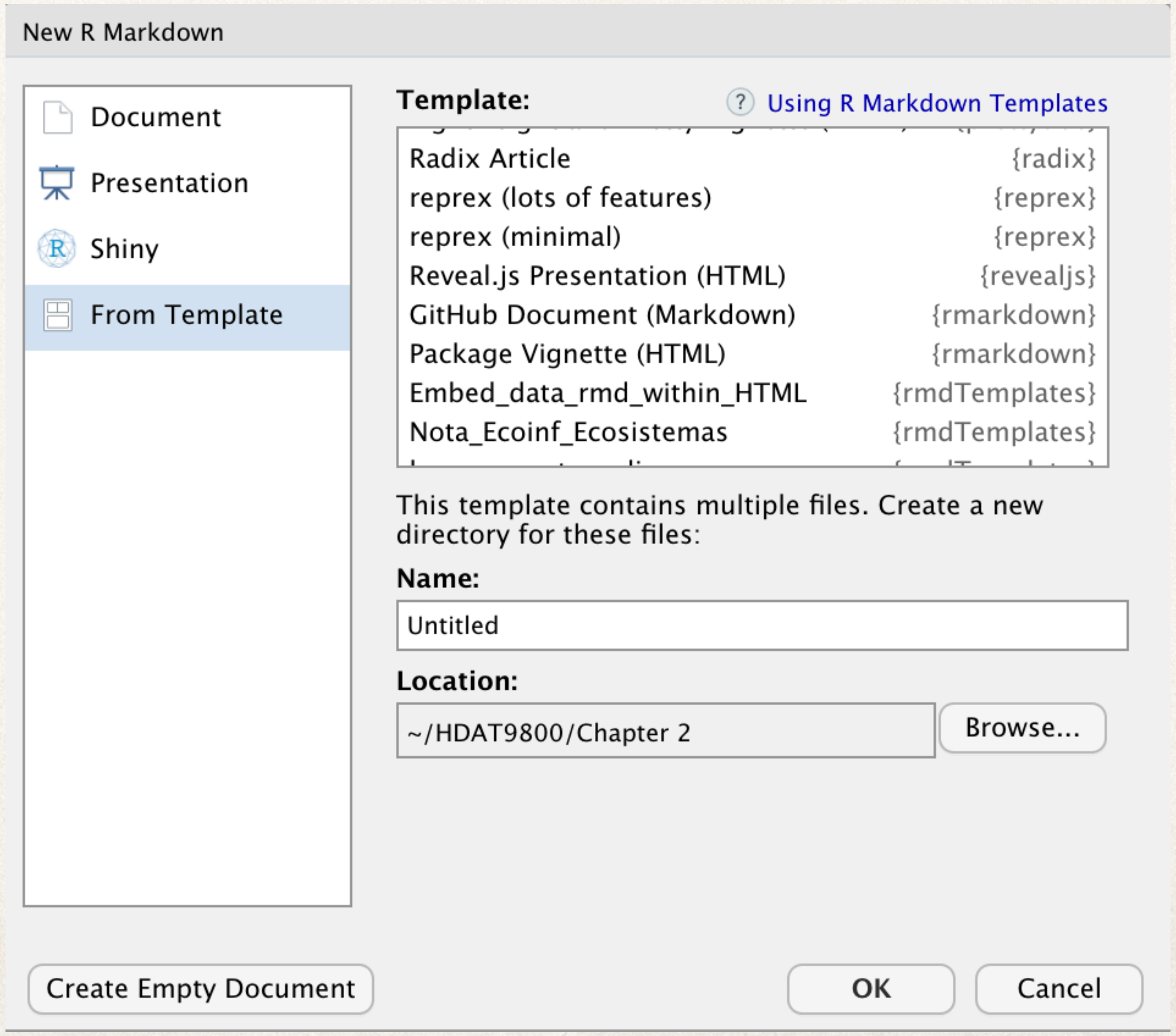
This template contains multiple files. Create a new directory for these files:

Name: Untitled

Location: ~/HDAT9800/Chapter 2

R Markdown templates

Installable packages of additional R Markdown document templates


New R Markdown

Template: Using R Markdown Templates

Radix Article	{radix}
reprex (lots of features)	{reprex}
reprex (minimal)	{reprex}
Reveal.js Presentation (HTML)	{revealjs}
GitHub Document (Markdown)	{rmarkdown}
Package Vignette (HTML)	{rmarkdown}
Embed_data_rmd_within_HTML	{rmdTemplates}
Nota_Ecoinf_Ecosistemas	{rmdTemplates}

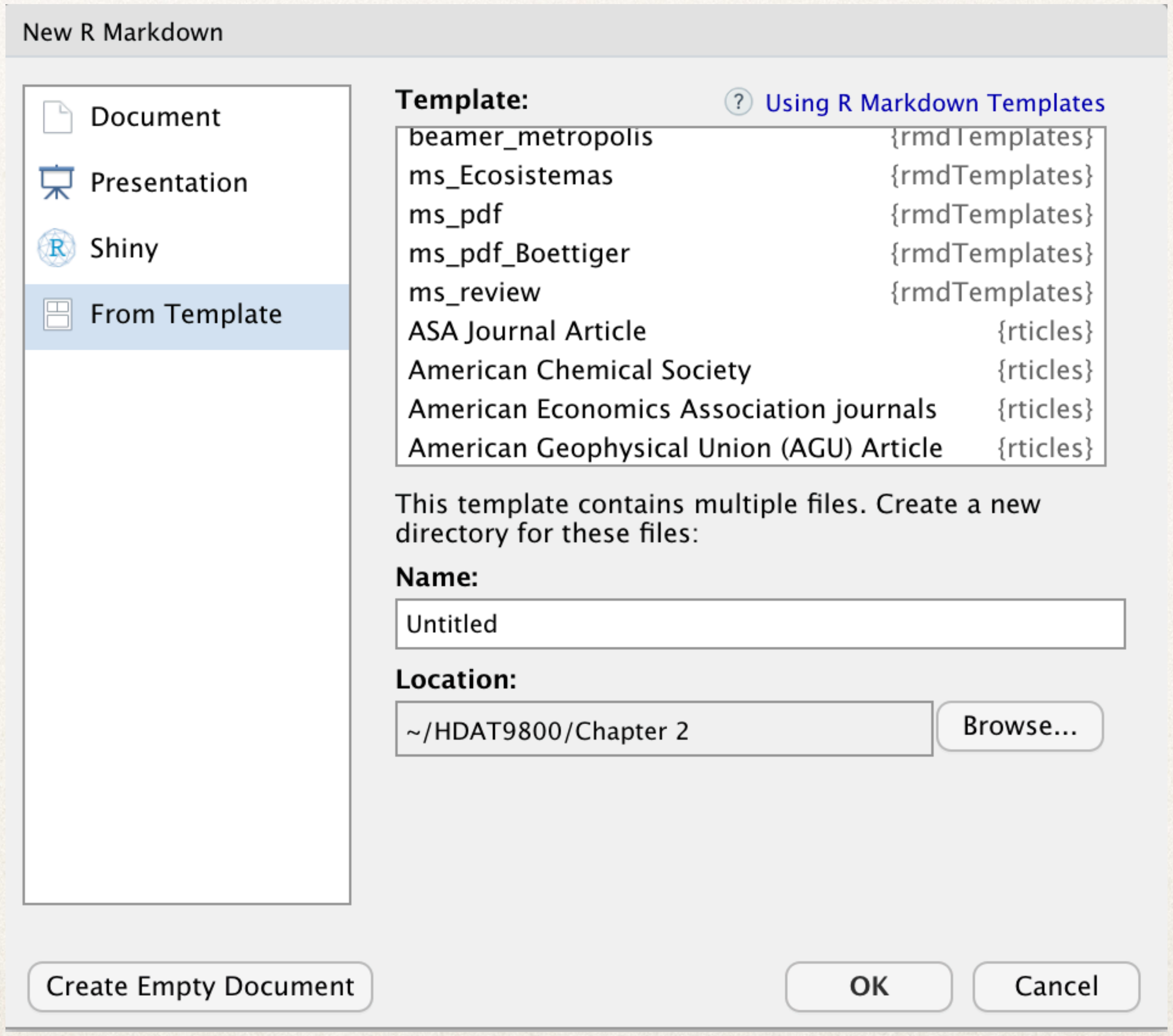
This template contains multiple files. Create a new directory for these files:

Name: Untitled

Location: ~/HDAT9800/Chapter 2

R Markdown templates

Installable packages of additional R Markdown document templates



Template:

beamer_metropolis	{rmdTemplates}
ms_Ecosistemas	{rmdTemplates}
ms_pdf	{rmdTemplates}
ms_pdf_Boettiger	{rmdTemplates}
ms_review	{rmdTemplates}
ASA Journal Article	{rticles}
American Chemical Society	{rticles}
American Economics Association journals	{rticles}
American Geophysical Union (AGU) Article	{rticles}

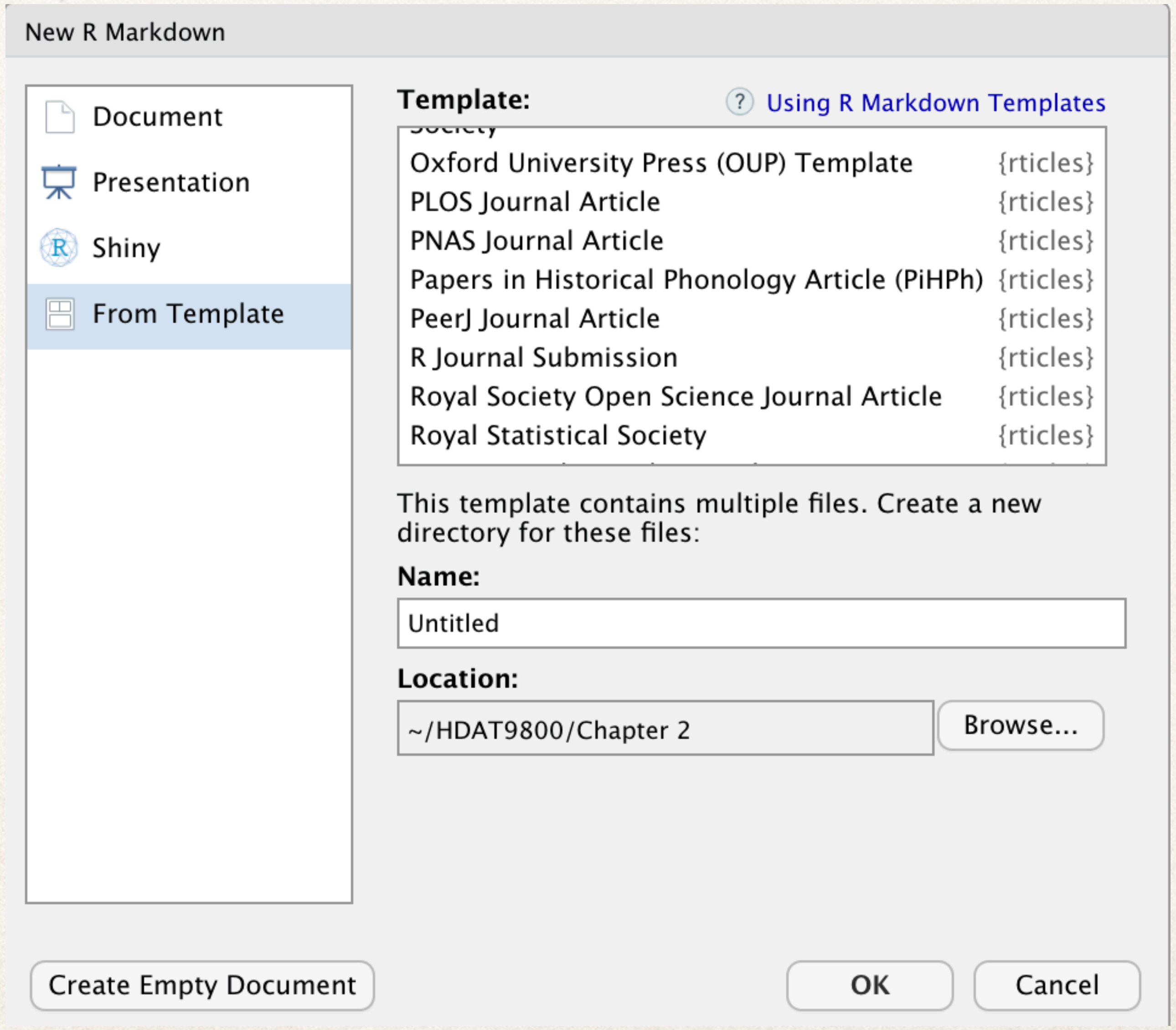
This template contains multiple files. Create a new directory for these files:

Name: Untitled

Location: ~/HDAT9800/Chapter 2

R Markdown templates

Installable packages of additional R Markdown document templates



Template:

Oxford University Press (OUP) Template	{rticles}
PLOS Journal Article	{rticles}
PNAS Journal Article	{rticles}
Papers in Historical Phonology Article (PiHPh)	{rticles}
PeerJ Journal Article	{rticles}
R Journal Submission	{rticles}
Royal Society Open Science Journal Article	{rticles}
Royal Statistical Society	{rticles}

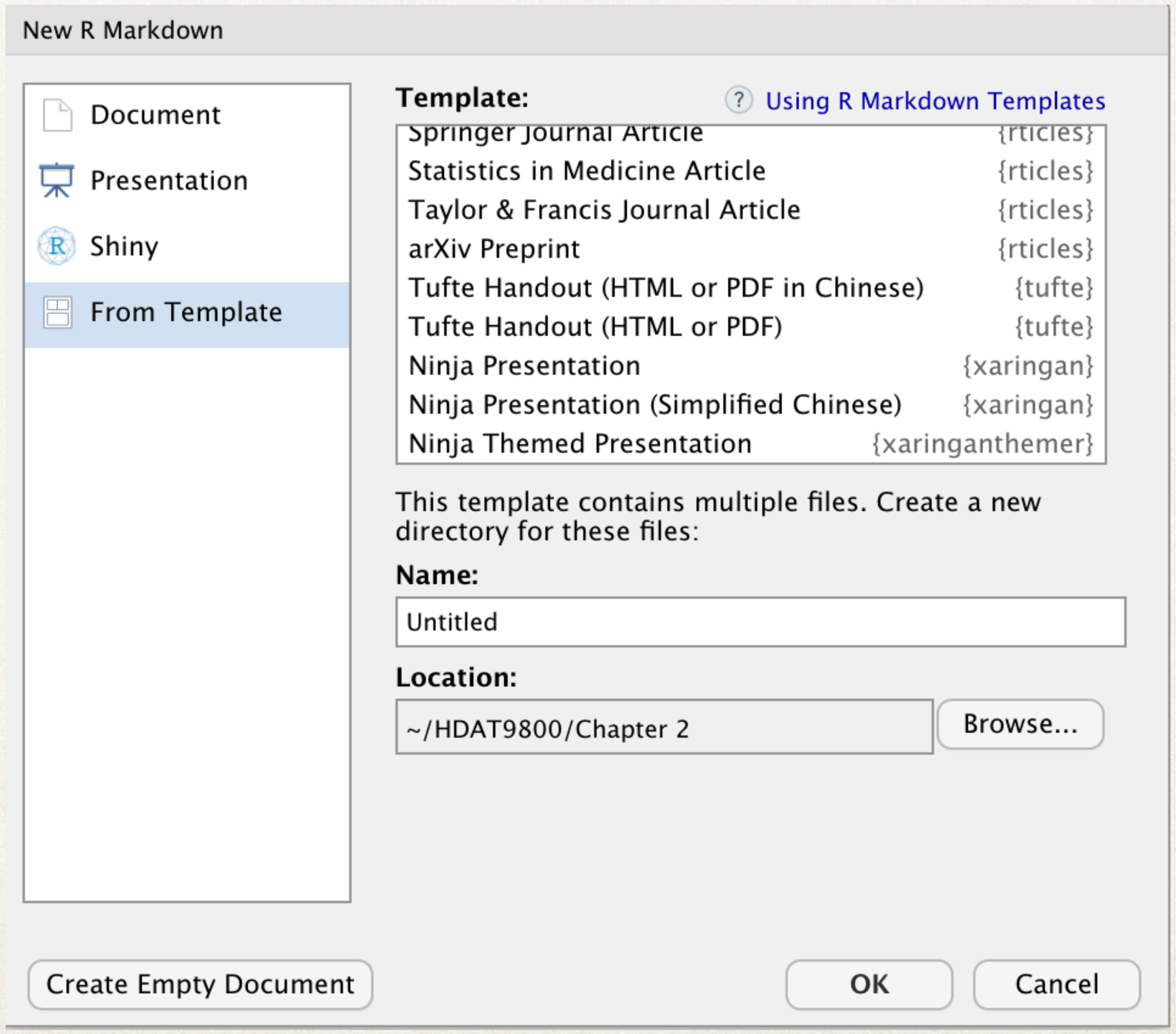
This template contains multiple files. Create a new directory for these files:

Name: Untitled

Location: ~/HDAT9800/Chapter 2

R Markdown templates

Installable packages of additional R Markdown document templates


New R Markdown

Template: Using R Markdown Templates

Springer Journal Article	{rticles}
Statistics in Medicine Article	{rticles}
Taylor & Francis Journal Article	{rticles}
arXiv Preprint	{rticles}
Tufte Handout (HTML or PDF in Chinese)	{tufte}
Tufte Handout (HTML or PDF)	{tufte}
Ninja Presentation	{xaringan}
Ninja Presentation (Simplified Chinese)	{xaringan}
Ninja Themed Presentation	{xaringanthemer}

This template contains multiple files. Create a new directory for these files:

Name: Untitled

Location: ~/HDAT9800/Chapter 2

Managing literate programming documents with *git*

.Rmd files are plain text files

As such they are ideal for management with *git*

Don't forget to initialise each project with *git*

Commit your work regularly

Rmd files

This file is different to a plain R script

- parts of the document are formatted text for reading
- parts of the document are R code for executing

The code sections are run and the output inserted into the document

The final documents can be in HTML, PDF or even Word documents

Text is marked up using a simple notation designed to conveniently indicate things like bold or italic text, or heading and lists

Document metadata

At the top of the document is a section that looks like this

```
---
title: "Untitled"
author: “Tim Churches"
date: “05/06/2021"
output: html_document
---
```

This is metadata or ‘data about the data in the document’

It can be a lot more sophisticated (it’s YAML) but these are the basics

Markdown

Markdown is a simple “mark-up” syntax. “Marking up” a document means indicating how you want the text and other features in it to appear on the printed pages (or in a PDF, or on a web page etc)

Markdown is quite expressive and can be used to embed hyperlinks, images, numbered and unnumbered lists.

There are many variants of Markdown, with slight differences. In R we use R Mrakdown (often written as Rmarkdown).

Look up *http://rmarkdown.rstudio.com* for more information

Markdown

Markdown is plain text with a little bit of markup to control formatting

- *…* or _…_ can be used to indicate *italic*
- **…** or __…__ can be used to indicate **bold**
- # Title Text can be used to format the heading ‘Title Text’
    - The number of ##’s indicates the heading level, start with 2
    - The title of the document (from the metadata) is usually rendered automatically as a first level heading

Lists

Lists are formatted using

- `*` or `+` for unordered list (the actual symbol doesn't matter)

```
* item
+ item
  * sub item
```

$\longrightarrow$

- item
- item
    - sub item

- `1.` for ordered lists (only the first number matters)

```
2. first item
5. second item
   1. first sub item
```

$\longrightarrow$

2. first item
3. second item
    1. first sub item

Hyperlinks

Hyperlinks can be included as follows

```
[the text of the link](http://example.com/page.html)
```

which will render appropriately for the output document format.

Images

Images are included using a similar notation to hyperlinks

```
![image caption text](http://example.com/logo.png)
```

For files local to a project a file relative URL can be specified

```
![image caption text](images/logo.png)
```

Equations

A good subset of LaTeX equations can be used

Inline equations are surrounded by $

```
Let $x = \sqrt{a_i^2 - b_j^2}$ as previously.
```

Let $x = \sqrt{a_i^2 - b_j^2}$ as previously.

Block equations are surrounded by $$ and get centred on a new line

```
$$ \sigma^2=\frac{\sum_{i=1}^{n}\left( x_i-\bar{x}\right)^2} {n-1} $$
```

$$\sigma^2=\frac{\sum_{i=1}^{n}\left(x_i-\bar{x}\right)^2} {n-1}$$

And more…

See the Rmarkdown and Pandoc markdown reference guides for more

- tables
- rules
- block quotes
- citations
- raw HTML (useful if your output format is HTML)

See also the Chapter 2 *learnr* tutorial, which goes into more detail.

Embedded code chunks

Code can be embedded as a paragraph surrounded by ``` marks (backticks)

The general format of these sections is

~~~
```{r <chunk-name>, <options>}
<programme-code>
```
~~~

The ‘r’ means this is R code (yes, you can embed python code, SQL and other types of code too)

The options control things like whether this code appears in the document, whether the code gets run and whether the output is included in the document

There are **many** options — see https://yihui.org/knitr/options/

The setup chunk

The first chunk is usually called something like 'setup')

~~~
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
~~~

The `include=FALSE` option says 'don't include this chunk in the final document'

The code in this example section sets the `echo` option default for all chunks to be `TRUE` meaning by default all chunks will have their output placed into the document

This saves us from putting `echo=TRUE` as an option on every chunk

All the chunks make one script

Consider all the chunks in a file to be one long script

The chunks are run one after the other

Changes in state and variables created and modified by earlier chunks are visible to later chunks

Technically: all the R code in chunks runs in its own environment for that particular Rmd document, which is **not** the R global environment, which is why objects created in the code chunks don't appear in the Environment tab in the top right of the RStudio window (it shows the global environment).

knitr

In RStudio, Rmarkdown document have a ‘Knit’ button on the toolbar

- format the markdown
- run the R code
- join everything together into a final document

The metadata options control whether we get HTML, PDF, or MS Word files as output by default

You can knit programmatically of course using

Developing a document

The code and text grow together

One we finish one we pretty much have finished the other

Everything is bundled together

Knitr runs the chunks of code like one big R programme and inserts the output results (and sometimes the input code) into the document

Don’t forget to use *add* and *commit* regularly to track your development

Resources

https://rmarkdown.rstudio.com

https://rmarkdown.rstudio.com/authoring_pandoc_markdown.html

https://yihui.org/knitr/