[Name of Workshop][Year]

Instructors: [list instructor names here]

[Insert dates of the workshop]

Contents

Part I

Introduction

Chapter 1

Welcome

Welcome to CBW's [workshop name, year] Workshop!

Put some introductory content here. (ex. links to bioinformatics.ca, general info)

Chapter 2

Course Schedule

Copy paste a table into https://www.tablesgenerator.com/markdown_tables (select convert to markdown) to create a table in markdown.

2.1 Pre-workshop Materials

Click here for your prework!

2.2 Computing Setup & Downloads

Insert downloads (ex. datasets) or other tech instructions here (ex. AWS Instructions)

2.3 Meet Your Faculty

Here's your team!

2.3.1 Instructor, TA, ...

Job Title Company/University/... Location

— contact information

[insert description of the person]

2.3.2 Michelle Brazas, PhD

Scientific Director Canadian Bioinformatics Workshops (CBW)
Toronto, ON, CA

— support@bioinformatics.ca

Dr. Michelle Brazas is the Associate Director for Adaptive Oncology at the Ontario Institute for Cancer Research (OICR), and acting Scientific Director at Bioinformatics.ca. Previously, Dr. Brazas was the Program Manager for Bioinformatics.ca and a faculty member in Biotechnology at BCIT. Michelle co-founded and runs the Toronto Bioinformatics User Group (TorBUG) now in its 11th season, and plays an active role in the International Society of Computational Biology where she sits on the Board of Directors and Executive Board.

2.3.3 Nia Hughes (she/her)

Program Manager, Bioinformatics.ca Ontario Institute for Cancer Research Toronto, ON, Canada

— nia.hughes@oicr.on.ca

Nia is the Program Manager for Bioinformatics.ca, where she coordinates the Canadian Bioinformatics Workshop Series. Prior to starting at OICR, she completed her M.Sc. in Bioinformatics from the University of Guelph in 2020 before working there as a bioinformatician studying epigenetic and transcriptomic patterns across maize varieties.

2.4 Class Photo

<- Replace the file address to your actual class photo file location

Part II

Modules

Chapter 3

Module 1

Welcome to module 1!

3.1 Lecture

Here is an example of a pdf embedded:

Sample PDF

This is a simple PDF file. Fun fun fun.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Phasellus facilisis odio sed mi. Curabitur suscipit. Nullam vel nisi. Etiam semper ipsum ut lectus. Proin aliquam, erat eget pharetra commodo, eros mi condimentum quam, sed commodo justo quam ut velit. Integer a erat. Cras laoreet ligula cursus enim. Aenean scelerisque velit et tellus. Vestibulum dictum aliquet sem. Nulla facilisi. Vestibulum accumsan ante vitae elit. Nulla erat dolor, blandit in, rutrum quis, semper pulvinar, enim. Nullam varius congue risus. Vivamus sollicitudin, metus ut interdum eleifend, nisi tellus pellentesque elit, tristique accumsan eros quam et risus. Suspendisse libero odio, mattis sit amet, aliquet eget, hendrerit vel, nulla. Sed vitae augue. Aliquam erat volutpat. Aliquam feugiat vulputate nisl. Suspendisse quis nulla pretium ante pretium mollis. Proin velit ligula, sagittis at, egestas a, pulvinar quis, nisl.

Pellentesque sit amet lectus. Praesent pulvinar, nunc quis iaculis sagittis, justo quam lobortis tortor, sed vestibulum dui metus venenatis est. Nunc cursus ligula. Nulla facilisi. Phasellus ullamcorper consectetuer ante. Duis tincidunt, urna id condimentum luctus, nibh ante vulputate sapien, id sagittis massa orci ut enim. Pellentesque vestibulum convallis sem. Nulla consequat quam ut nisl. Nullam est. Curabitur tincidunt dapibus lorem. Proin velit turpis, scelerisque sit amet, iaculis nec, rhoncus ac, ipsum. Phasellus lorem arcu, feugiat eu, gravida eu, consequat molestie, ipsum. Nullam vel est ut ipsum volutpat feugiat. Aenean pellentesque.

In mauris. Pellentesque dui nisi, iaculis eu, rhoncus in, venenatis ac, ante. Ut odio justo, scelerisque vel, facilisis non, commodo a, pede. Cras nec massa sit amet tortor volutpat varius. Donec lacinia, neque a luctus aliquet, pede massa imperdiet ante, at varius lorem pede sed sapien. Fusce erat nibh, aliquet in, eleifend eget, commodo eget, erat. Fusce consectetuer. Cras risus tortor, porttitor nec, tristique sed, convallis semper, eros. Fusce vulputate ipsum a mauris. Phasellus mollis. Curabitur sed urna. Aliquam nec sapien non nibh pulvinar convallis. Vivamus facilisis augue quis quam. Proin cursus aliquet metus. Suspendisse lacinia. Nulla at tellus ac turpis eleifend scelerisque. Maecenas a pede vitae enim commodo interdum. Donec odio. Sed sollicitudin dui vitae justo.

Morbi elit nunc, facilisis a, mollis a, molestie at, lectus. Suspendisse eget mauris eu tellus molestie cursus. Duis ut magna at justo dignissim condimentum. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Vivamus varius. Ut sit amet diam suscipit mauris ornare aliquam. Sed varius. Duis arcu. Etiam tristique massa eget dui. Phasellus congue. Aenean est erat, tincidunt eget, venenatis quis, commodo at, quam.

Here is an example of a YouTube video embedded:

^ HEIGHT HAS A BUG

3.1.1 Downloads

[insert your downloads for this module here (ex. datasets)]

3.2 Lab

[Your lab here]

```
# Your R code here

# For example:
x <- 42
x
```

```
## [1] 42
```

```
# Your python code here

# For example:
print("hello world")
```

```
## hello world
```

```
# Your bash code here

# For example:
pwd
```

```
## /Users/jqiu/Documents/CBWgithub/cbw-dev-templates-docs/bookdown-template
```

Try running these code "chunks" by pressing the green (left-pointing) triangle next to your code chunks.

You will see the code run in the console and the output provided below the code chunk.

The output of the code will also be produced under the code chunk on your website page.

Chapter 4

Module 2

4.1 Lecture

4.2 Lab

Chapter 5

How to Render/Compile Code

There are many ways to render and compile code using Bookdown!

Here are most of the "language engines" (programming languages) available to render, run and compile in Bookdown!

```
names(knitr::knit_engines$get())
```

```
##  [1] "awk"         "bash"        "coffee"      "gawk"        "groovy"
##  [6] "haskell"     "lein"        "mysql"       "node"        "octave"
## [11] "perl"        "php"         "psql"        "Rscript"     "ruby"
## [16] "sas"         "scala"       "sed"         "sh"          "stata"
## [21] "zsh"         "asis"        "asy"         "block"       "block2"
## [26] "bslib"       "c"           "cat"         "cc"          "comment"
## [31] "css"         "ditaa"       "dot"         "embed"       "eviews"
## [36] "exec"        "fortran"     "fortran95"   "go"          "highlight"
## [41] "js"          "julia"       "python"      "R"           "Rcpp"
## [46] "sass"        "scss"        "sql"         "stan"        "targets"
## [51] "tikz"        "verbatim"    "theorem"     "lemma"       "corollary"
## [56] "proposition" "conjecture"  "definition"  "example"     "exercise"
## [61] "hypothesis"  "proof"       "remark"      "solution"
```

5.1 Rendering Code

Just rendering code refers to Bookdown formatting code so that our users can view it. The code does not run or compile. This is very similar to what is shown

in 020-module-1.Rmd of the bookdown template. An example is shown below.

```
```
insert code here
```
```

which appears as

```
insert code here
```

These are called "code chunks".

5.2 Rendering Code with Highlights for Specific Languages

Bookdown also highlights (specific to the language) the outputted code for the ease of understanding. To do so, add the name of the language (of the many shown above) after the first 3 ` symbols. Note that the language is not case sensitive (so both "r" and "R" would render the same below). For example:

```
```r
x <- 42
x
```
```

renders into:

```r
x <- 42
x
```

Notice that since <- is how to assign values to variables in R, it is highlighted. If we were to replace "r" with a different language, this may not be highlighted (depending if "<-" is a relevant symbol in that language.)

5.3 Rendering and Compiling Code

To have the code actually compile, we need to put "{" and "}" brackets around the language, which we still put after the first 3 ` symbols. For example:

````
```{r}
x <- 42
x
htmltools::HTML('<b>LEFT`RIGHT</b>') htmltools::HTML('<b>LEFT`RIGHT</b>') htmltools::HTML('<b>LEF
```
````

renders and compiles into:

```
x <- 42
x
```

```
## [1] 42
```

The second bar you see is the output of R command!

You can run these "code chunks" without previewing or building the book! To the right of your code chunk, there are 3 symbols (as seen below).

The gear symbol leads to more chunk options click on it to see more options.

The down arrow symbol runs all previous chunks and the current chunk. This is helpful if previous chunks define a variable that you will need in your current chunk.

> **Note:** Hence, we can use variables from previous chunks!

The right-pointing arrow symbol runs the current code chunk. A rectangular box will appear below your code chunk, with the output of your code.

Additionally, notice that in your bottom left window, any code you run also runs in your console. Both of these ways of checking your code happens for all languages! If you are previewing your book, you must re-preview it, since running the code becomes the new action for your console, instead of previewing the book.

Note!

If you want to run a certain language, you must have the language installed! R and python are already setup for you. (Python relies on the RStudio interface option, reticulate - you can reconfigure this if you'd like.) You may have to configure new languages so that they can run in RStudio, but generally, they should be able to run automatically.

Exiting the Console

If you're running R, you don't have to exit the console (we already work in the R console when using Bookdown). However, if you're running a different language (for example, python), you will remain in the Python console, and must exit it to run Bookdown commands. You can look online for different ways to exit certain consoles, but generally running "quit" will return you to the R console.

Now we know how to both render (just show) and compile (see the output) of our code!

5.4 Code Chunk Options

Bookdown has options that we can include, to help with certain options for our code to appear and what it may produce.

Generally, the typical structure of adding code chunk options is shown below.

```
```{r, option=VALUE, option=VALUE, ...}
code
```
```

Some common code chunk options are:

- `echo`
  - `echo=TRUE` shows the code output (by default, it is on).
  - `echo=FALSE` hides the code output.
- `eval`
  - `eval=TRUE` runs the code (default).
  - `eval=FALSE` skips the chunk and does not execute the code.
- `results`
  - `results='markup'` runs the code (default).
  - `results='asis'` leaves the output with no additional formatting.
  - `results='hide'` does not show the output.
- `include`
  - `include=TRUE` includes both the code and the output on your website (default).

- `include=FALSE` excludes both the code and the output on your website.

Click here to find more code chunk options!

5.5 Code Chunks for Code-Generated Figures and Tables

We can use code chunks to help specify certain ways for code-generated figures and tables to appear, and how to link to them!

Generally, this will look like:

````
```{r reference-name, option=VALUE, option=VALUE, ...}
code that generates an image/table
```
````

Then, we can refer to the figure or table generated by the code in this chunk by using `\@ref(fig:reference-name)` or `\@ref(tab:chunk-label)`

For example, an R-generated image can be made using a similar code chunk to the one shown below:

````
See Figure \@ref(fig:nice-fig).

```{r nice-fig, fig.cap='Here is a nice figure!', out.width='80%', fig.asp=.75, fig.align='center
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```
````

This renders into:

See Figure 5.1.

```r
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

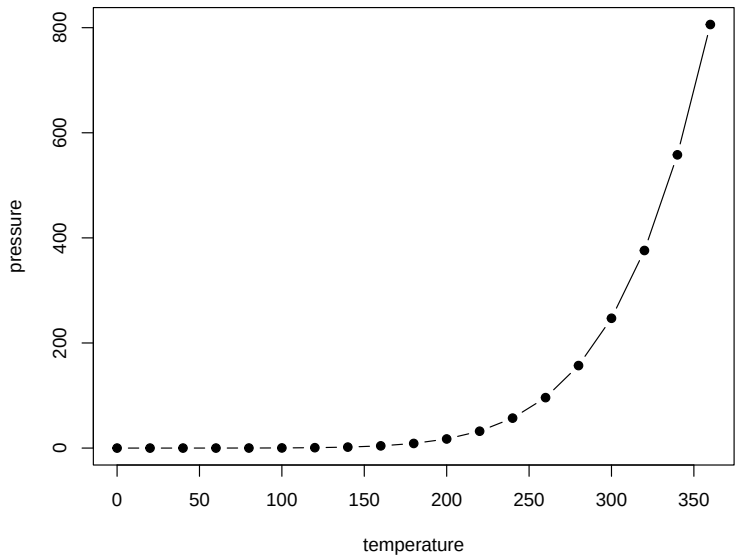

Figure 5.1: Here is a nice figure!

Here is an example of a table generated by R from a code chunk:

````
Don't miss Table \@ref(tab:nice-tab).

```{r nice-tab, tidy=FALSE}
knitr::kable(
  head(pressure, 10), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```
````

This renders into:

Don't miss Table 5.1.

```r
knitr::kable(
  head(pressure, 10), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

Table 5.1: Here is a nice table!

temperature	pressure
0	0.0002
20	0.0012
40	0.0060
60	0.0300
80	0.0900
100	0.2700
120	0.7500
140	1.8500
160	4.2000
180	8.8000

You can find more inforamtion on figures and tables in Bookdown here!