THE TITLE OF THE BOOK

AUTHOR NAME

Invalid Date

Table of contents

# Preface

This is a Quarto book, generated using the sipbs-compbiol-book-template GitHub template.

The index.qmd file provides this Preface page, which is intended as a frontispiece with brief introductory information about the book and its contents and scope and, maybe, its authors. To change the main text and content you see here (the content in the middle section of the page), you need to edit the index.qmd file.

* To learn more about writing Quarto books, visit <https://quarto.org/docs/books>.

|  |
| --- |
| Tip |
| To put links into Quarto pages, use the structure:  [Text to be displayed](URL of the link)  For example, [this link](https://www.youtube.com/watch?v=dQw4w9WgXcQ) is written as  [this link](https://www.youtube.com/watch?v=dQw4w9WgXcQ) |

## How this page is structured

### Section headers

Section headers are written in [Markdown](#markdown) using hash/pound signs:

## This is a (level two) section header.  
### This is a (level three) section header

* To learn more about sections and headers, see this [quick guide to Markdown basics](https://quarto.org/docs/authoring/markdown-basics.html).

|  |
| --- |
| Changing title and author information, or the cover image (click to expand) |
| The book title, author, date, and cover image are specified in the \_quarto.yml. To change this information, edit that file, and re-render the page.  book:  title: "THE TITLE OF THE BOOK"  author: "AUTHOR NAME"  date: "DD/MM/YY"  cover-image: sipbs\_compbio\_800.png |

### Callout blocks

Callout blocks, like the ones above, are highlighted regions of the document which carry a title, icon and colour, which may indicate the kind of information the callout contains. For example:

|  |
| --- |
| Warning |
| This is a warning block. To insert a block like this, use:  ::: { .callout-warning } The text of the callout goes here ::: |

|  |
| --- |
| Caution (click to expand) |
| This is a caution block, which is collapsible when rendered as HTML/webpage but appears in full in the Word document. To make a callout collapsible, use collapse="true" as in the example below. This places a dropdown icon to the right of the title bar, which controls expansion.  ::: { .callout-caution collapse="true" title="Caution (click to expand)"} This is a collapsible caution block. The callout content goes here. :::  Notice also that the title of the callout is specified between the curly braces as title="Caution (click to expand)". |

You can learn more about callout blocks in the [Quarto documentation](https://quarto.org/docs/authoring/callouts.html).

# 1. Introduction

The Introduction page is intended as a short introduction to the book.

Like most Quarto books, this is a book created from markdown and executable code.

This kind of book is an example of literate programming - the intertwining of nicely-formatted text and images, and executable code. For example, the R code cell below executes and produces output when the book is compiled:

1 + 1

[1] 2

But the R code cell below does not:

summary(cars)

See Knuth ([1984](#ref-knuth84)) for additional discussion of literate programming.

# 2. Getting started

|  |
| --- |
| Important |
| This page provides a step-by-step tutorial for using the sipbs-compbiol-book-template GitHub template. By following this guide you will:   * Create a new GitHub repository for your materials/project, using this book template. * Create a local copy of the repository on your machine, which you can edit using, for example, RStudio. * Set up the repository to use GitHub Pages as a platform for publishing your book/materials. * Make local changes to the files on your own machine, and update the public pages by *pushing* them to GitHub. |

* [sipbs-compbiol-book-template repository](https://github.com/sipbs-compbiol/sipbs-compbiol-book-template)

The basic steps we cover below are:

1. Use the GitHub page for this template to create a new GitHub repository
2. Clone the new repository to your local machine
3. Set up the new repository to display as a website on GitHub Pages
4. Edit your repository locally, and push changes back to GitHub (they will automatically update the webpage)

## Step 1: Create a new GitHub repository from the template.

1. Navigate to [this template’s repository at GitHub](https://github.com/sipbs-compbiol/sipbs-compbiol-book-template) using your web browser.
2. Click on the green Use This Template button at the top right of the sipbs-compbiol-book-template GitHub page ([Figure 2.1](#fig-use-this-template)).

|  |
| --- |
| Warning |
| You will only see this button if you are signed into your own GitHub account.  If you do not already have a GitHub account, the [GitHub documentation](https://docs.github.com/en/get-started/start-your-journey/creating-an-account-on-github) can walk you through the process of obtaining one, and logging in. |

|  |
| --- |
| <assets/videos/template-tutorial-video-1.mp4>  Figure 2.1: Video demonstration of using this template to create a new GitHub repository. |

1. A drop-down menu will appear. Click on the Create a new repository option. This will bring you to a page ([Figure 2.1](#fig-use-this-template)) for you to provide details about the new repository you want to create.

|  |
| --- |
| Tip |
| * Give your repository a unique, memorable, descriptive name. This will make it easier to find and help users understand what the repository is for. * Add a brief description, explaining the purpose of your new repository. This will help users understand what the repository is for. * Leave Include all branches unchecked. * Make sure the repository status is set to Public. |

1. When you are satisfied that the details for creating your new repository are correct, click the green Create repository button ([Figure 2.1](#fig-use-this-template)).

## Step 2: Clone the new repository to your personal machine

You will be editing your online material on your own machine, and *pushing* the repository to GitHub for version control and to create and update the webpage where people will read your material. You need to *clone* your new repository to your machine, so that you can work on it. The first part of this process is to copy the URL that will let you clone the repository.

|  |
| --- |
| Note |
| There are several equally valid methods for cloning a repository to your own computer. Covering all of these is beyond the scope of this tutorial, so we only present one method here. Any approach that works is fine. |

1. Click on the green Code button on your GitHub repository. This will reveal a drop-down box with a number of tabs and options ([Figure 2.2](#fig-get-repo-url))
2. With the SSH tab selected, click on the Copy URL to clipboard button.

|  |
| --- |
| <assets/videos/template-tutorial-video-1a.mp4>  Figure 2.2: Video demonstration: copying the URL for your repository so it can be cloned to your local computer. |

1. At the command-line/terminal, type the command git clone and paste the repository URL that you copied to clipboard from your new repository ([Figure 2.2](#fig-get-repo-url)).

## Step 3: Set up the local repository to render to GitHub Pages

1. In the terminal, change directory to your new repository (e.g. if your new repository is called my-new-repo, use cd my-new-repo as in [Figure 2.3](#fig-clone-repo)).
2. Run the command quarto publish gh-pages. You will be asked whether you want to publish the site to GitHub. Press the Y key to accept.

|  |
| --- |
| <assets/videos/template-tutorial-video-2.mp4>  Figure 2.3: Video demonstration of cloning the repository to your local computer, and configuring it to render the document at GitHub Pages. |

After a short pause, your browser should automatically open your new site pages at GitHub. You will be able to navigate around the material just like a normal webpage ([Figure 2.4](#fig-repo-site)).

|  |
| --- |
| <assets/videos/template-tutorial-video-3.mp4>  Figure 2.4: Video demonstration of the published web materials. |

|  |
| --- |
| Important |
| You should not need to modify any further settings for changes that you now push to your GitHub repository to be made live at your repository’s website.   * The quarto publish gh-pages command automatically created a gh-pages branch at GitHub, and placed a rendered version of your new repository website in it ([Figure 2.5](#fig-repo-config)). * Your repository is automatically set to use this gh-pages branch of your repository to host its webpages ([Figure 2.5](#fig-repo-config)). * The template includes a GitHub Action that will automatically rebuild your Quarto site when any changes are pushed to the main repository. Once pushed, the changes will go live following a short delay to build the pages.  |  | | --- | | <assets/videos/template-tutorial-video-4.mp4>  Figure 2.5: Video demonstration of the repository configuration. | |

## Step 4: Edit your local repository (in RStudio) and push changes to GitHub Pages

You can now edit your online materials in the repository using RStudio in the usual way. When you are ready to make changes to the public site showing the materials, you only need to *commit* your changes and *push* them to the repository. The rest is handled automatically. The process is demonstrated in [Figure 2.6](#fig-demo-commit), and the main actions are:

* Open your repository in RStudio
* Make an edit to any file that displays part of the materials online (e.g. index.qmd)
* Commit your changes in the git tab of the RStudio window
* Push your changes to the GitHub repository (and wait for the site to build)
* Check the repository and GitHub Pages site to confirm the changes are live

|  |
| --- |
| <assets/videos/template-tutorial-video-5.mp4>  Figure 2.6: Video demonstration of the process of updating the public site by editing a file and pushing changes to the repository. |

1. Open the template/R project using RStudio (e.g. double-click on the .Rproj file as in [Figure 2.6](#fig-demo-commit)).
2. Make the desired edits to any files (e.g. modify the index.qmd file as in [Figure 2.6](#fig-demo-commit)).
3. Save your changes locally.
4. Select the changed files in the git tab of RStudio and click the Commit icon. A new window will appear.
5. In the new window, enter a short commit string, and click the Commit button. The commit string will disappear. This *commits* your changes to the local repository.
6. Click on the Push icon (the up arrow) to *push* your changes to the GitHub repository. A short message will appear, confirming that the files have transferred.
7. To confirm the changes were pushed to the GitHub repository, you can visit the repository site and check the individual file contents.
8. To confirm that the website is being updated, you can click on the Actions link at the top of the repository page. This will summarise recent GitHub Actions and will tell you if your site has been built, is in the process of being built, or if an error occurred.
9. Once the site has been built by the GitHub Action, you will be able to visit the public GitHub Pages site for this repository, and inspect the changes, as in [Figure 2.6](#fig-demo-commit).

# 3. How do I…?

This page is intended as a FAQ/HowTo for some common questions that might come up when using this template for the first time.

Some [example pages](example_pages.qmd) dedicated to common tasks and specific kinds of presentation are included later in the template.

## 3.1 …create a new page?

|  |
| --- |
| tl;dr |
| 1. Create a new plain text file 2. Write Markdown content in your file 3. Save your file as a .qmd file (with an appropriate filename) 4. Add your filename to the \_quarto.yml file in an appropriate place 5. Commit your changes 6. Push the changes to your repository |

Assuming that you’re using RStudio, you need to create a new file to hold the Markdown for your page. I usually create a new text file (File -> New File -> Text File or click on the New File icon and select Text File). Write some content in the file and save it as a .qmd file with a meaningful filename (e.g. this file is called howto.qmd).

At this point, the file you just wrote could be rendered, but it won’t be linked as a page in the book. To link the file, open the \_quarto.yml file and look for the section that starts:

# Define chapters and sections here  
 # [E] CHAPTERS AND SECTIONS  
 chapters:  
 - index.qmd  
 - intro.qmd

Then enter the name of your file in the appropriate place, e.g. in the early-section.qmd part, as a chapter following tutorial.qmd in the example below:

- part: early-section.qmd  
 chapters:   
 - tutorial.qmd  
 - howto.qmd # new page added at this line

|  |
| --- |
| Warning |
| The \_quarto.yml file is written in a language called YAML (yet another markup language), and the syntax is important. If you follow the style of the file, you should be fine. |

and then render the file.

|  |
| --- |
| Important |
| To ensure your changes are not lost, *commit* the changes to your repository (your .qmd file, the \_quarto.yml file, and any supporting images or other files), and *push* them to GitHub. |

## 3.2 …set a value once to be used everywhere?

Quarto supports *global variables*, values that are defined once in a document, and that can be referred to at any point by using a special sequence of characters called a *shortcode*. The variables are defined in the \_variables.yml file, using the YAML markup language.

|  |
| --- |
| Tip |
| Global variables can be useful for defining details that might be repeated multiple times across your book:   * administrative information, like contact emails and numbers * the academic year for a module presentation (saves searching for every mention in all pages) |

To include the value held in a variable called myvar, you would define this in \_variables.yml as:

myvar: "The value of the variable"

and use the shortcode:

{{< var myvar >}}

Variables can be named directly, as with myvar above, but they can also be grouped by category as category.variable combinations. For instance, if you wanted to define a set of telephone numbers for three rooms HW312, HW313, and HW314, you could give these the category phone, defining them as, e.g.:

phone:  
 HW312: 01-234-657890  
 HW313: 01-234-657891  
 HW313: 01-234-657892

and use them with shortcode:

{{< var phone.HW313 >}}

## Appearance

This is global\_variable: This is an example of a global variable.

This is category.value1: This is an example of a category.value variable.

This is category.value2: This is another example of a category.value variable.

## Markdown

This is `global\_variable`: {{< var global\_variable >}}.  
  
This is `category.value1`: {{< var category.value1 >}}.  
  
This is `category.value2`: {{< var category.value2 >}}.

## \_variables.yml

# Example variables for the howto page  
global\_variable: "This is an example of a global variable"  
category:  
 value1: "This is an example of a `category.value` variable"  
 value2: "This is another example of a `category.value` variable"

# 4. Useful links

This page provides links to online resources that may be useful when working with Quarto or this template.

## 4.1 Quarto

* [Quarto guide](https://quarto.org/docs/guide/)
* [Using R with Quarto](https://quarto.org/docs/computations/r.html)

### 4.1.1 Markdown

* [Quarto Markdown basics](https://quarto.org/docs/authoring/markdown-basics.html)
* [kableExtra documentation](https://www.rdocumentation.org/packages/kableExtra/versions/1.4.0)
* [RMarkdown cookbook](https://bookdown.org/yihui/rmarkdown-cookbook/)

## 4.2 R

* [Programming with R (Carpentries)](https://swcarpentry.github.io/r-novice-inflammation/)
* [R for Reproducible Scientific Analysis (Carpentries)](https://swcarpentry.github.io/r-novice-gapminder/)

### 4.2.1 Data processing

* [readr documentation](https://readr.tidyverse.org/)
* [dplyr documentation](https://dplyr.tidyverse.org/)
* [tidyr documentation](https://tidyr.tidyverse.org/)
* [stringr documentation](https://stringr.tidyverse.org/)
* [lubridate documentation](https://lubridate.tidyverse.org/)

### 4.2.2 Data Visualisation

* [ggplot2 documentation](https://ggplot2.tidyverse.org/reference/)
* [ggally documentation](https://ggobi.github.io/ggally/reference/index.html)
* [ggpubr documentation](https://rpkgs.datanovia.com/ggpubr/index.html)

### 4.2.3 Shiny

* [Shiny documentation](https://shiny.posit.co/r/getstarted/shiny-basics/lesson1/)

## 4.3 RStudio

* [RStudio Getting Started](https://docs.posit.co/ide/user/ide/get-started/)
* [RStudio tutorial (DataCamp)](https://www.datacamp.com/tutorial/r-studio-tutorial)
* [Using git from RStudio (Carpentries)](https://opsis.eci.ox.ac.uk/swc-git-novice/14-supplemental-rstudio.html)

### 4.3.1 Shiny

* [r-shinylive` documentation](https://posit-dev.github.io/r-shinylive/)
* [shinylive documentation](https://shiny.posit.co/py/get-started/shinylive.html)

### 4.3.2 WebR

* [WebR documentation](https://docs.r-wasm.org/webr/latest/)

## 4.4 Version Control with git and GitHub

### 4.4.1 git

* [Version control with git (Carpentries)](https://swcarpentry.github.io/git-novice/)
* [About git (GitHub)](https://docs.github.com/en/get-started/using-git/about-git)
* [Using git with RStudio](https://r-bio.github.io/intro-git-rstudio/)

### 4.4.2 GitHub

* [Getting started with GitHub](https://docs.github.com/en/get-started)
* [Creating an account on GitHub](https://docs.github.com/en/get-started/start-your-journey/creating-an-account-on-github)
* [Using git/GitHub with R](https://rfortherestofus.com/2021/02/how-to-use-git-github-with-r/)

## 4.5 Cheatsheets

* [Quarto/RMarkdown cheatsheet](https://posit.co/wp-content/uploads/2022/10/rmarkdown-1.pdf)
* [readr cheatsheet](https://raw.githubusercontent.com/rstudio/cheatsheets/main/data-import.pdf)
* [dplyr cheatsheet](https://github.com/rstudio/cheatsheets/blob/main/data-transformation.pdf)
* [tidyr cheatsheet](https://github.com/rstudio/cheatsheets/blob/master/tidyr.pdf)
* [stringr cheatsheet](https://github.com/rstudio/cheatsheets/blob/main/strings.pdf)
* [lubridate cheatsheet](https://rawgit.com/rstudio/cheatsheets/main/lubridate.pdf)
* [ggplot2 cheatsheet](https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf)
* [Shiny cheatsheet](https://github.com/rstudio/cheatsheets/raw/main/shiny.pdf)
* [git cheatsheet](https://education.github.com/git-cheat-sheet-education.pdf)
* [GitHub cheatsheet](https://lahirumw.github.io/2025-01-26-github-cheat-sheet/)

# 5. Custom Callouts

This page demonstrates the use of the [quarto-custom-callout](https://quarto.thecoatlessprofessor.com/custom-callout/) extension to provide bespoke callout blocks.

* [quarto-custom-callout repository](https://github.com/coatless-quarto/custom-callout)
* [quarto-custom-callout examples and documentation](https://quarto.thecoatlessprofessor.com/custom-callout/)

## 5.1 Introduction

Quarto provides a number of built-in callout blocks, including:

|  |
| --- |
| Caution |
| The caution callout  ::: { .callout-caution } The caution callout ::: |

|  |
| --- |
| Important |
| The important callout  ::: { .callout-info } The important callout ::: |

|  |
| --- |
| Tip |
| The tip callout  ::: { .callout-tip } The tip callout ::: |

but these do not cover all the cases we might want to use callouts for. It is possible to change the titles of these callouts, such as:

|  |
| --- |
| Custom title |
| The tip callout with a custom title  ::: { .callout-tip title="Custom title"} The tip callout with a custom title ::: |

|  |
| --- |
| Not the caution callout |
| ::: { .callout-caution } ## Not the caution callout ::: |

but customising the colour and icon of a callout is more difficult. The quarto-custom-callout extension allows us to make our own callouts.

## 5.2 Defining a new callout

To make a new callout, add a new entry in \_quarto.yml in the custom-callout block with the name of the callout. For instance, to add a new case study callout, we could use the code below:

custom-callout:  
 callout-case-study:  
 title: "Case Study"  
 icon-symbol: "🔍"  
 color: "#FFA500"

and ensure that the custom-callout filter is called in \_quarto.yml:

filters:  
 - custom-callout

|  |
| --- |
| Callout naming convention |
| We have chosen to give all of our callouts for this template the prefix callout- for continuity with the builtin callouts, and to aid with literate programming/self-documentation. Please keep to this convention when you define your own callouts. |

|  |
| --- |
| Tip |
| Notice that the icon-symbol field accepts unicode icons and emojis. |

## 5.3 Custom callouts in this template

We have defined the custom callouts below:

|  |
| --- |
| Case Study |
| For describing case studies  ::: { .callout-case-study } For describing case studies ::: |

|  |
| --- |
| Challenge |
| For presenting a challenge to readers  ::: { .callout-challenge } For presenting a challenge to readers ::: |

|  |
| --- |
| Data Analysis |
| For describing datasets and data analysis  ::: { .callout-data } For describing datasets and data analysis ::: |

|  |
| --- |
| Discussion Questions: |
| For introducing discussion points  ::: { .callout-discussion } For introducing discussion points ::: |

|  |
| --- |
| Microbe Profile |
| For summarising the properties of a microbe  ::: { .callout-microbe } For summarising the properties of a microbe ::: |

|  |
| --- |
| Question |
| For presenting a single question to readers  ::: { .callout-question } For presenting a single question to readers ::: |

|  |
| --- |
| Questions |
| For presenting multiple questions to readers  ::: { .callout-questions } For presenting multiple questions to readers ::: |

|  |
| --- |
| Great job! |
| For general encouragement  ::: { .callout-thumbs-up } For general encouragement ::: |

|  |
| --- |
| Callout-todo |
| For highlighting where we still need to do some work  ::: { .callout-todo } For highlighting where we still need to do some work ::: |

# 6. Embedding video files

This page demonstrates how to embed video into a Quarto webpage. This is a native capability of Quarto, and requires no extensions to be installed.

* [Quarto video embedding documentation](https://quarto.org/docs/authoring/videos.html)

## 6.1 Embedding a video

To embed a video, use the {{< video >}} shortcode. You can embed local files or URLs of online videos:

{{< video local-video-file.mp4 >}}  
  
{{< video URL\_TO\_VIDEO >}}

### 6.1.1 Local file

|  |
| --- |
| Tip |
| This template provides an assets/videos folder, which can be used for adding video files to the repository. |

## Appearance

<assets/videos/template-tutorial-video-1a.mp4>

## Markdown

{{< video assets/videos/template-tutorial-video-1a.mp4 >}}

### 6.1.2 Remote video file

## Appearance

<https://youtu.be/dQw4w9WgXcQ?si=_BZIcKuz9U91pGqB>

## Markdown

{{< video https://youtu.be/dQw4w9WgXcQ?si=\_BZIcKuz9U91pGqB >}}

### 6.1.3 Cross-referencing videos

To use cross-references with videos, wrap the video in a div using three colons, as in the example below. They can then be referenced using the usual @fig-label method.

## Appearance

|  |
| --- |
| <assets/videos/template-tutorial-video-1a.mp4>  Figure 6.1: This is an example video with cross-referencing (and a legend!) |

And it’s cross-referenced like this ([Figure 6.1](#fig-example-1)).

## Markdown

::: {#fig-example-1}  
  
{{< video assets/videos/template-tutorial-video-1a.mp4 >}}  
  
This is an example video with cross-referencing (and a legend!)  
  
:::  
  
And it's cross-referenced like this (@fig-example-1).

# 7. Embedding PDF files

There are two ways to embed PDF files in a Quarto page: by using native Quarto support, or with the embedpdf extension.

|  |
| --- |
| Warning |
| This is useful only for HTML/webpage output, and embedding PDFs will not render as expected in Word document or other outputs. |

## 7.1 Native embedding

PDF files can be embedded in Quarto using the standard link syntax:

![An example PDF document](example.pdf)

## Appearance

![Kiepas et al. (2024)](data:application/pdf;base64,)

Kiepas *et al.* (2024)

## Code

![Kiepas \_et al.\_ (2024)](assets/papers/kiepas\_et\_al\_2024.pdf)

### 7.1.1 Controlling embedded document size

To set the height and width of the presentation of the embedded document, use brace notation:

![An example PDF document](example.pdf){width=80% height=300px}

## Appearance

![Kiepas et al. (2024)](data:application/pdf;base64,)

Kiepas *et al.* (2024)

## Code

![Kiepas \_et al.\_ (2024)](assets/papers/kiepas\_et\_al\_2024.pdf){width=100% height=300px}

## 7.2 embedpdf

embedpdf is a Quarto extension that enables embedding of PDF files directly into Quarto HTML files. The extension is included as part of this template.

* [embedpdf repository](https://github.com/jmgirard/embedpdf)
* [embedpdf examples and documentation](https://jmgirard.github.io/embedpdf/example.html)

With embedpdf, files are embedded using [shortcodes](https://quarto.org/docs/extensions/shortcodes.html), e.g.:

{{< pdf example.pdf >}}

## Appearance

## Code

{{< pdf assets/papers/kiepas\_et\_al\_2024.pdf >}}

### 7.2.1 Controlling embedded document size

To set the height and width of the presentation of the embedded document, add these variables to the shortcode:

{{< pdf example.pdf height=200px width=80%>}}

## Appearance

## Code

{{< pdf assets/papers/kiepas\_et\_al\_2024.pdf height=300px width=100% >}}

### 7.2.2 Installing embedpdf

The extension is packaged as part of this template, but if you do need to install embedpdf in other Quarto projects, use the command:

quarto add jmgirard/embedpdf

# 8. Interactive Multiple Choice Questions

This page demonstrates the setup of an [naquiz multiple-choice quiz](https://nareal.github.io/naquiz/example.html). The advantage of this extension is that we don’t need to use a Shiny server.

* [naquiz repository](https://github.com/nareal/naquiz)
* [naquiz examples and documentation](https://nareal.github.io/naquiz/example.html)

## 8.1 Multiple Choice Questions

### 8.1.1 Basic MCQ

A simple naquiz example looks something like the question below to the reader (click the Markdown tab to see how to write the markdown for it).

## Appearance

Bill Gates was the founder of:

Apple

Microsoft

Facebook

Google

## Markdown

:::::{.question}  
Bill Gates was the founder of:  
  
::::{.choices}  
  
:::{.choice}  
Apple  
:::   
  
:::{.choice .correct-choice}  
Microsoft  
:::  
  
:::{.choice}  
Facebook   
:::  
  
:::{.choice}  
Google   
:::  
  
::::  
:::::

|  |
| --- |
| Caution |
| The nested colon dividers and their counts can look confusing on the page, and missed/extra colons can break the layout of your page.   * The {.question} divider gets **five** colons. * The {.choices} divider that surrounds the choices presented to the reader gets **four** colons. * Each individual {.choice} divider gets **three** colons. |

It presents a series of *radio buttons* from which the reader can choose one option. On clicking the option, the reader receives very basic feedback: a red cross (incorrect) or green tick (correct).

### 8.1.2 Add a “clear answer” button

The naquiz MCQs can be decorated with useful buttons for reader interaction. The example below has a “Clear Answer” button, which unchecks the reader’s selected answer.

## Appearance

Bill Gates was the founder of:

Apple

Microsoft

Facebook

Google

## Markdown

:::::{.question}  
Bill Gates was the founder of:  
  
::::{.choices}  
  
:::{.choice}  
Apple  
:::   
  
:::{.choice .correct-choice}  
Microsoft  
:::  
  
:::{.choice}  
Facebook   
:::  
  
:::{.choice}  
Google   
:::  
  
:::{.button-clear title="Clear answer" button-class="btn btn-xs"}  
:::  
  
::::  
:::::

|  |
| --- |
| Important |
| The button to clear reader answers needs to be linked with the available choices, and so is placed **inside the {.choices} division** of the MCQ. |

### 8.1.3 Add a “hint” button

The example below includes a “hint” button, which the reader can use to get a clue to the answer.

## Appearance

Bill Gates was the founder of:

Apple

Microsoft

Facebook

Google

The company name starts with an ‘M’…

## Markdown

:::::{.question}  
Bill Gates was the founder of:  
  
::::{.choices}  
  
:::{.choice}  
Apple  
:::   
  
:::{.choice .correct-choice}  
Microsoft  
:::  
  
:::{.choice}  
Facebook   
:::  
  
:::{.choice}  
Google   
:::  
  
:::{.button-clear title="Clear answer" button-class="btn btn-xs"}  
:::  
  
::::  
  
::::{.btn-group}  
:::{.button-hint title="Show hint" button-class="btn btn-xs"}  
The company name starts with an 'M'...  
:::  
::::  
  
:::::

|  |
| --- |
| Important |
| We can place additional buttons within a *button group* divider, specified as ::::{.btn-group}, *inside* the question divider (the five colons :::::).  Here, we have linked a {.button-hint} that, when clicked, expands to show the hint text. |

### 8.1.4 Add an “answer” button

We can add a further button to provide students with the answer we’d like them to read, directly.

## Appearance

Bill Gates was the founder of:

Apple

Microsoft

Facebook

Google

The company name starts with an ‘M’…

Bill Gates and Paul Allen founded Microsoft on April 4, 1975.

## Markdown

:::::{.question}  
Bill Gates was the founder of:  
  
::::{.choices}  
  
:::{.choice}  
Apple  
:::   
  
:::{.choice .correct-choice}  
Microsoft  
:::  
  
:::{.choice}  
Facebook   
:::  
  
:::{.choice}  
Google   
:::  
  
:::{.button-clear title="Clear answer" button-class="btn btn-xs"}  
:::  
  
::::  
  
::::{.btn-group}  
:::{.button-hint title="Show hint" button-class="btn btn-xs"}  
The company name starts with an 'M'...  
:::  
::::  
  
:::{.button-answer title="Show Answer" button-class="btn btn-xs"}  
Bill Gates and Paul Allen founded Microsoft on April 4, 1975.  
:::  
:::::

|  |
| --- |
| Important |
| We can place more than one button in the *button group* divider. Here, we have linked a {.button-answer} that, when clicked, expands to show the hint text. |

### 8.1.5 Placing an MCQ in a custom callout

We have created a custom callout called callout-question which can be used to make MCQ stand out a bit more, with a consistent visual style.

## Appearance

|  |
| --- |
| Question |
| Bill Gates was the founder of:  Apple  Microsoft  Facebook  Google  The company name starts with an ‘M’…  Bill Gates and Paul Allen founded Microsoft on April 4, 1975. |

## Markdown

::: { .callout-question }  
  
:::::{.question}  
Bill Gates was the founder of:  
  
::::{.choices}  
  
:::{.choice}  
Apple  
:::   
  
:::{.choice .correct-choice}  
Microsoft  
:::  
  
:::{.choice}  
Facebook   
:::  
  
:::{.choice}  
Google   
:::  
  
:::{.button-clear title="Clear answer" button-class="btn btn-xs"}  
:::  
  
::::  
:::::  
  
::::{.btn-group}  
:::{.button-hint title="Show hint" button-class="btn btn-xs"}  
The company name starts with an 'M'...  
:::  
  
:::{.button-answer title="Show Answer" button-class="btn btn-xs"}  
Bill Gates and Paul Allen founded Microsoft on April 4, 1975.  
:::  
::::  
  
:::

### 8.1.6 Randomising question values

We can use R or Python to generate random values that can be used to compose questions that are different each time the page is rendered, by inserting a suitable R/Python code block and using the values in the question.

|  |
| --- |
| Caution |
| The values do **not** change when the page is reloaded by a reader. The values are calculated when the pages are rendered.  For values that change when the page is refreshed, or on demand, use an R [Shiny](https://shiny.posit.co/r/getstarted/shiny-basics/lesson1/) or [Shinylive](https://posit-dev.github.io/r-shinylive/) solution, or a solution like [Numbas](https://www.numbas.org.uk/). |

## Appearance

|  |
| --- |
| Question |
| What is the sum of 4 and 12?  -8  16  8  48  You need to add the two numbers 4 and 12  4 + 12 = 16 |

## Markdown

::: { .callout-question }  
  
:::::{.question}  
What is the sum of `r x\_val` and `r y\_val`?  
  
::::{.choices}  
  
:::{.choice}  
`r x\_val - y\_val`  
:::   
  
:::{.choice .correct-choice}  
`r x\_val + y\_val`  
:::  
  
:::{.choice}  
`r y\_val - x\_val`  
:::  
  
:::{.choice}  
`r y\_val \* x\_val`  
:::  
  
:::{.button-clear title="Clear answer" button-class="btn btn-xs"}  
:::  
  
::::  
:::::  
  
::::{.btn-group}  
:::{.button-hint title="Show hint" button-class="btn btn-xs"}  
You need to add the two numbers `r x\_val` and `r y\_val`  
:::  
  
:::{.button-answer title="Show Answer" button-class="btn btn-xs"}  
`r x\_val` + `r y\_val` = `r x\_val + y\_val`  
:::  
::::  
  
:::

## R code

#| show: false  
#| echo: false  
  
x\_val <- sample(1:10, 1)  
y\_val <- sample(11:20, 1)

# 9. WebR Playground

#| context: setup  
  
# Download reporter data  
download.file('https://raw.githubusercontent.com/sipbs-compbiol/BM214-Workshop-3/main/assets/data/reporter\_curves.csv', 'reporter\_curves.csv')  
  
library(palmerpenguins)  
library(tidyverse)

## 9.1 Introduction

This page provides a [WebR](https://docs.r-wasm.org/webr/latest/) cell for use as a playground to experiment with example datasets, and describes how to include WebR on other Quarto pages.

* [WebR documentation](https://docs.r-wasm.org/webr/latest/)

## 9.2 Playground

## Appearance

# Use this WebR cell to experiment with using R. You can do almost anything  
# by typing in this code cell that you can do in R, including plotting graphs.  
#  
# Why not try the command:  
#  
# plot(penguins)  
#  
# and click the "Run code" icon?

## Markdown

```{webr-r}  
# Use this WebR cell to experiment with using R. You can do almost anything  
# by typing in this code cell that you can do in R, including plotting graphs.  
#  
# Why not try the command:  
#  
# plot(penguins)  
#  
# and click the "Run code" icon?  
  
```

## 9.3 Examples to try

## Palmer Penguins

The penguins dataset contains data about three different species of penguins. You can use the R commands below to investigate the data:

glimpse(penguins) # look at the format of the dataset  
plot(penguins) # plot a visual overview of the dataset  
  
# Plot the count of penguins in each species as a bar chart  
ggplot(penguins, aes(species, fill=species)) + geom\_bar()  
  
# Break down the distribution by species in a facet plot  
ggplot(penguins, aes(species, fill=species)) + geom\_bar() + facet\_wrap(~sex)  
  
# Make a box and whisker plot of penguin body mass by species:  
ggplot(penguins, aes(x=species, y=body\_mass\_g, fill=species)) + geom\_boxplot()  
  
# Plot the body mass for each sex side-by-side  
ggplot(penguins, aes(x=species, y=body\_mass\_g, fill=sex)) + geom\_boxplot()  
  
# Investigate correlation between body mass and flipper length:  
ggplot(penguins, aes(x=body\_mass\_g, y=flipper\_length\_mm)) + geom\_point()  
  
# Colour datapoints by species:  
ggplot(penguins, aes(x=body\_mass\_g, y=flipper\_length\_mm, colour=species)) +  
 geom\_point()  
  
# Fit a linear regression to each species separately:  
ggplot(penguins, aes(x=body\_mass\_g, y=flipper\_length\_mm, colour=species)) +  
 geom\_point() +  
 geom\_smooth(method="lm")

## Plot reporter curves

One of our [BM214 workshops](https://sipbs-compbiol.github.io/BM214-Workshop-3/) involves a WebR-supported interactive exercise involving simulated reporter curves.

Here, we have downloaded some data from the online repository and made it available as though it was a dataset on the filesystem, reporter\_curves.csv.

# Load reporter experiment data  
data <- read.csv("reporter\_curves.csv")  
  
glimpse(data) # Summarise the data format  
  
# Plot absorbance against concentration as a facet plot  
ggplot(data, aes(x=conc, y=abs\_ratio, color=sample)) + geom\_point() +  
 geom\_line() + facet\_wrap(~sample)

## Other datasets

R comes with a number of example datasets you can practice with, including:

* mtcars: fuel consumption and other statistic for 32 automobiles
* Titanic: the fates of passengers from the maiden voyage of the ocean liner *Titanic*

You can see a full list by running the command

library(help = "datasets")

## 9.4 Setting up WebR

There are three important elements to including WebR on a Quarto page: the YAML header, the setup code block, and the WebR cell.

You can see an example of how to set up WebR on any Quarto page by inspecting the .qmd source for this page.

### 9.4.1 YAML header

To use WebR on a Quarto page, you need to include an appropriate YAML header at the top of the page:

---  
webr:  
 packages: ["tidyverse", "palmerpenguins"]  
filters:  
 - webr  
---

Packages to be imported into WebR should be specified in the list, as tidyverse and palmerpenguins are, here. The webr filter must be loaded.

### 9.4.2 WebR setup block

There is a special R code block needed to prepare the WebR instance. For this page, the code block

```{webr-r}  
#| context: setup  
  
# Download reporter data  
download.file('https://raw.githubusercontent.com/sipbs-compbiol/BM214-Workshop-3/main/assets/data/reporter\_curves.csv', 'reporter\_curves.csv')  
  
library(palmerpenguins)  
library(tidyverse)  
```

declares that it has the setup context, so will be run when the WebR instance starts. The download.file() call pulls the reporter\_curves.csv file from a GitHub repository and makes it available in WebR, as if it were a real file, with the name reporter\_curves.csv. The last three lines import R packages into the WebR environment.

### 9.4.3 The WebR code cell

The minimal WebR code cell looks like this:

```{webr-r}  
```

but it is more helpful, and more usual, to include some example code or comments, as in the example below:

## Appearance

# You can use `R` as a calculator  
  
1 \* sin(1)

## Markdown

```{webr-r}  
# You can use `R` as a calculator  
  
1 \* sin(1)  
```

# 10. Using shinylive

shinylive is an extension and R package that allows for embedding a *serverless* Shiny application in a Quarto page, as we do in [Figure 10.1](#fig-shinylive-example).

## Appearance

|  |
| --- |
| #| standalone: true #| viewerHeight: 600  library(shiny) library(bslib)  # Define UI for app that draws a histogram ---- ui <- page\_sidebar(  sidebar = sidebar(open = "open",  numericInput("n", "Sample count", 100),  checkboxInput("pause", "Pause", FALSE),  ),  plotOutput("plot", width=500) )  server <- function(input, output, session) {  data <- reactive({  input$resample  if (!isTRUE(input$pause)) {  invalidateLater(1000)  }  rnorm(input$n)  })    output$plot <- renderPlot({  op <- par(cex = 0.5)    hist(data(),  breaks = 40,  xlim = c(-2, 2),  ylim = c(0, 1),  lty = "blank",  xlab = "value",  freq = FALSE,  main = ""  )    x <- seq(from = -2, to = 2, length.out = 500)  y <- dnorm(x)  lines(x, y, lwd=1.5)    lwd <- 5  abline(v=0, col="red", lwd=lwd, lty=2)  abline(v=mean(data()), col="blue", lwd=lwd, lty=1)    legend(legend = c("Normal", "Mean", "Sample mean"),  col = c("black", "red", "blue"),  lty = c(1, 2, 1),  lwd = c(1, lwd, lwd),  x = -2,  y = 1  )  }, res=140) }  # Create Shiny app ---- shinyApp(ui = ui, server = server)  Figure 10.1: Interactive shinylive example showing random samples from a Normal distribution. Click the ‘Pause’ checkbox to freeze on a sample. Use the ‘Sample count’ option to choose the number of samples. The black line shows the Normal distribution being sampled from, and the dashed red line shows the population mean. The solid blue line shows the sample mean. The grey histogram shows the sample. |

## Markdown

```{shinylive-r}  
#| standalone: true  
#| viewerHeight: 600  
  
{CODE GOES HERE}  
```

## Code

library(shiny)  
library(bslib)  
  
# Define UI for app that draws a histogram ----  
ui <- page\_sidebar(  
 sidebar = sidebar(open = "open",  
 numericInput("n", "Sample count", 100),  
 checkboxInput("pause", "Pause", FALSE),  
 ),  
 plotOutput("plot", width=500)  
)  
  
server <- function(input, output, session) {  
 data <- reactive({  
 input$resample  
 if (!isTRUE(input$pause)) {  
 invalidateLater(1000)  
 }  
 rnorm(input$n)  
 })  
   
 output$plot <- renderPlot({  
 op <- par(cex = 0.5)  
   
 hist(data(),  
 breaks = 40,  
 xlim = c(-2, 2),  
 ylim = c(0, 1),  
 lty = "blank",  
 xlab = "value",  
 freq = FALSE,  
 main = ""  
 )  
   
 x <- seq(from = -2, to = 2, length.out = 500)  
 y <- dnorm(x)  
 lines(x, y, lwd=1.5)  
   
 lwd <- 5  
 abline(v=0, col="red", lwd=lwd, lty=2)  
 abline(v=mean(data()), col="blue", lwd=lwd, lty=1)  
   
 legend(legend = c("Normal", "Mean", "Sample mean"),  
 col = c("black", "red", "blue"),  
 lty = c(1, 2, 1),  
 lwd = c(1, lwd, lwd),  
 x = -2,  
 y = 1  
 )  
 }, res=140)  
}  
  
# Create Shiny app ----  
shinyApp(ui = ui, server = server)

## The fundamentals of a Shiny app

We don’t have space here for a tutorial on how to use Shiny. There is a learning curve, and we can recommend a number of online resources for you to get up to speed with this package, including:

* [Shiny documentation](https://shiny.posit.co/r/getstarted/shiny-basics/lesson1/)
  + [Shiny cheatsheet](https://github.com/rstudio/cheatsheets/raw/main/shiny.pdf)

|  |
| --- |
| The bare bones |
| To implement a Shiny app, you need to define three things in your R code:   1. a *user interface* object that defines how the user interacts with the app (sliders, checkboxes, plots, etc.) 2. a *server* function that returns values to be displayed by the *user interface* 3. a call to the shinyApp() function   library(shiny) # of course, you have to make the Shiny package available  ui <- {CODE DEFINING USER INTERFACE}  server <- function(input, output, session) {CODE DEFINING WHAT HAPPENS}  shinyApp(ui = ui, server = server) |

## What you need to do special for shinylive

Under normal circumstances, the Shiny code with the three elements above would run on a Shiny server, and you wouldn’t need to do much else. But with shinylive and Quarto you need to place the app on the page, and tell Quarto that you want to use the shinylive filter. Firstly, the header of the page should be defined in YAML as, e.g.:

---  
title: "Using `shinylive`"  
filters:  
 - shinylive  
format:   
 html:  
 css: assets/fix\_editor.css  
---

|  |
| --- |
| Note |
| The format: section is required to fix a layout problem with r-shinylive interactive editors, described in **?@sec-editor**. |

This requires a new kind of fenced block stating that it is {shinylive-r}:

```{shinylive-r}  
{CODE GOES HERE}  
```

and there are new block-level arguments controlling on-page display, e.g.

```{shinylive-r}  
#| standalone: true  
#| viewerHeight: 600  
  
{CODE GOES HERE}  
```

|  |
| --- |
| Important |
| The standalone:true setting is required for the compiled app to run in your public repository. This setting tells shinylive that the entire app is contained within the fenced code block. |

### 10.0.1 Example

The shinylive example in [Figure 10.2](#fig-shinylive-ggplot2) below is fairly short but includes some important points of difference from [Figure 10.1](#fig-shinylive-example).

## Appearance

|  |
| --- |
| #| standalone: true #| viewerHeight: 600  library(shiny) library(ggplot2) library(DT)  if (FALSE) {  library(munsell) }  ui <- fluidPage(  plotOutput("plot", brush = "plot\_brush"),  DTOutput("table") )  server <- function(input, output, session) {  output$plot <- renderPlot(  ggplot(mtcars) +  geom\_point(aes(x = mpg, y = disp))  )  output$table <- renderDT({  brushedPoints(mtcars, input$plot\_brush)  }) }  shinyApp(ui = ui, server = server)  Figure 10.2: shinylive example using ggplot2, DT, and allowing for interactive selection of points from the graph. |

## Markdown

```{shinylive-r}  
#| standalone: true  
#| viewerHeight: 600  
  
# Import required libraries  
library(shiny)  
library(ggplot2)  
library(DT)  
  
# ggplot2 will not work in shinylive without this - see callout  
if (FALSE) {  
 library(munsell)  
}  
  
# Define the user interface  
ui <- fluidPage(  
 plotOutput("plot", brush = "plot\_brush"),  
 DTOutput("table")  
)  
  
# Define the server code  
server <- function(input, output, session) {  
 output$plot <- renderPlot(  
 ggplot(mtcars) +  
 geom\_point(aes(x = mpg, y = disp))  
 )  
 output$table <- renderDT({  
 brushedPoints(mtcars, input$plot\_brush)  
 })  
}  
  
# Run the shinyapp  
shinyApp(ui = ui, server = server)  
```

## Code

# Import required libraries  
library(shiny)  
library(ggplot2)  
library(DT)  
  
# ggplot2 will not work in shinylive without this - see callout  
if (FALSE) {  
 library(munsell)  
}  
  
# Define the user interface  
ui <- fluidPage(  
 plotOutput("plot", brush = "plot\_brush"),  
 DTOutput("table")  
)  
  
# Define the server code  
server <- function(input, output, session) {  
 output$plot <- renderPlot(  
 ggplot(mtcars) +  
 geom\_point(aes(x = mpg, y = disp))  
 )  
 output$table <- renderDT({  
 brushedPoints(mtcars, input$plot\_brush)  
 })  
}  
  
# Run the shinyapp  
shinyApp(ui = ui, server = server)

|  |
| --- |
| Using ggplot2 with shinylive |
| There is a known issue with ggplot2 and shinylive such that no output is produced due to a missing suggested dependency. A workaround is indicated at [this StackOverflow page](https://stackoverflow.com/questions/79678683/cannot-deploy-shinylive-shiny-app-in-r-with-ggplot2-dependency) :  library(ggplot2)  if (FALSE) {  library(munsell) } |

## Adding the R editor

The examples above allow for user interaction, but not user editing of the code. With shinylive it is possible to include an editor so that users can modify the code of the app directly, in their own browser.

To do this, include the code block setting: #| components: [editor, viewer], as in [Figure 10.3](#fig-shinylive-editor).

|  |
| --- |
| Tip |
| Quarto book pages are tall rather than wide, so it is often helpful to stack the editor and viewer vertically, with the options:  #| components: [editor, viewer] #| layout: vertical |

|  |
| --- |
| Important |
| The layout for the r-shinylive editor inherits text alignment from the surrounding <div> and can be misaligned. This book template includes a .css file that fixes this, and which must be included in the YAML header for the page:  format:   html:  css: assets/fix\_editor.css |

## Appearance

|  |
| --- |
| #| standalone: true #| viewerHeight: 600 #| components: [viewer, editor] #| layout: vertical  library(shiny) library(ggplot2) library(DT)  if (FALSE) {  library(munsell) }  ui <- fluidPage(  plotOutput("plot", brush = "plot\_brush"),  DTOutput("table") )  server <- function(input, output, session) {  output$plot <- renderPlot(  ggplot(mtcars) +  geom\_point(aes(x = mpg, y = disp))  )  output$table <- renderDT({  brushedPoints(mtcars, input$plot\_brush)  }) }  shinyApp(ui = ui, server = server)  Figure 10.3: shinylive example using ggplot2, DT, and allowing for interactive selection of points from the graph. |

## Markdown

```{shinylive-r}  
#| standalone: true  
#| viewerHeight: 600  
#| components: [viewer, editor]  
#| layout: vertical  
  
# Import required libraries  
library(shiny)  
library(ggplot2)  
library(DT)  
  
# ggplot2 will not work in shinylive without this - see callout  
if (FALSE) {  
 library(munsell)  
}  
  
# Define the user interface  
ui <- fluidPage(  
 plotOutput("plot", brush = "plot\_brush"),  
 DTOutput("table")  
)  
  
# Define the server code  
server <- function(input, output, session) {  
 output$plot <- renderPlot(  
 ggplot(mtcars) +  
 geom\_point(aes(x = mpg, y = disp))  
 )  
 output$table <- renderDT({  
 brushedPoints(mtcars, input$plot\_brush)  
 })  
}  
  
# Run the shinyapp  
shinyApp(ui = ui, server = server)  
```

## Code

# Import required libraries  
library(shiny)  
library(ggplot2)  
library(DT)  
  
# ggplot2 will not work in shinylive without this - see callout  
if (FALSE) {  
 library(munsell)  
}  
  
# Define the user interface  
ui <- fluidPage(  
 plotOutput("plot", brush = "plot\_brush"),  
 DTOutput("table")  
)  
  
# Define the server code  
server <- function(input, output, session) {  
 output$plot <- renderPlot(  
 ggplot(mtcars) +  
 geom\_point(aes(x = mpg, y = disp))  
 )  
 output$table <- renderDT({  
 brushedPoints(mtcars, input$plot\_brush)  
 })  
}  
  
# Run the shinyapp  
shinyApp(ui = ui, server = server)

|  |
| --- |
| Challenge |
| What happens if you change line 17 of the code in the editor to:  geom\_point(aes(x = mpg, y = disp, color=as.factor(cyl)))  and run the code (click on the triangle, or press Shift-Return/Cmd-Shift-Return) |

## Installing the shinylive extension

* [shinylive extension repository](https://github.com/coatless-quarto/r-shinylive-demo)

For local use and development you will need to install the shinylive package:

install.packages("shinylive")

To install the shinylive Quarto extension, use the commmand:

quarto add quarto-ext/shinylive

|  |
| --- |
| Important |
| The shinylive extension is installed as part of this template, and the shinylive package is installed *via* the DESCRIPTION file. This extension should work automatically in the rendered GitHub pages. |

# 11. Embedding Numbas Questions

[Numbas](https://numbas.org.uk) is an open-source e-assessment system aimed at mathematics and other numerate disciplines. It generates [SCORM 2004](https://scorm.com/scorm-explained/technical-scorm/scorm-2004-overview-for-developers/)-compliant, self-contained assessment packages, and can present randomised questions for practice and assessment.

* Find out more about Numbas, including case studies and the latest blog posts, at <https://numbas.org.uk>.

## Embedding a Numbas question

To embed a Numbas question or exam, Run the question/exam from the Numbas editor, and click on the Share button. This will reveal a URL that can be embedded in your Quarto page.

## Appearance

## Markdown

<iframe src="[URL]" width="780" height="500"></iframe>

# References

Knuth, Donald E. 1984. “Literate Programming.” *Comput. J.* 27 (2): 97–111. <https://doi.org/10.1093/comjnl/27.2.97>.