# A Reproducible Research Compendium

*cf. list of contributors at*
*https://github.com/rr-mrc-bsu/reproducible-research/graphs/contributors*

*2018-12-13*

# Contents

# Chapter 1

# A 'Living Book' - aims and scope

This book is a little different from your ususal statistics foliant - it is written entirely using **Markdown** and rendered to html, pdf, and epub publishing formats using the R package **bookdown**. Its entire Markdown source code is publicly available on GitHub.com at https://github.com/rr-mrc-bsu/reproducible-research. A pre-build version is hosted as static html website using **GitHub pages** at https://rr-mrc-bsu.github.io/reproducible-research/. This structure allows to easily discuss changes using GitHub issues https://github.com/rr-mrc-bsu/reproducible-research/issues, organize further development using milestones and projects, contribute corrections or even entire chapters by creating pull requests, and to manage editions by GitHub releases. It also means that everybody - and yes, that does include you - can become a contributor by creating pull requests in the GitHub repository. Since the contents are thus evolving over time as long as there are active contributors to the project, the book is 'living'.

The overall purpose of the *Reproducible Research Compendium* is threefold:

1. Provide a platform for discussing aims and objectives as well as best practices for reproducible research with a clear focus on applications in biostatistics.
2. Build-up a lasting compendium for knowledge sharing around various issues and methods for coping with them that may broadly be subsumed under the term 'reproducible research'.
3. The book project itself acts as a learning-by-doing example for its contributors with the goal of anybody participating becoming knowlegable about organizing collaborative open-source [mostly coding] projects.

The complete documentation for **bookdown** can be found at https://bookdown.org/yihui/bookdown/. Note that R is a prerequisite but only for building the book - the contents itself are completely language-agnostic.

# Chapter 2

# Introduction

[taken from bookdown template!]

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 2. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter **??**.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

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

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 2.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 2.1.

```
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2018) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

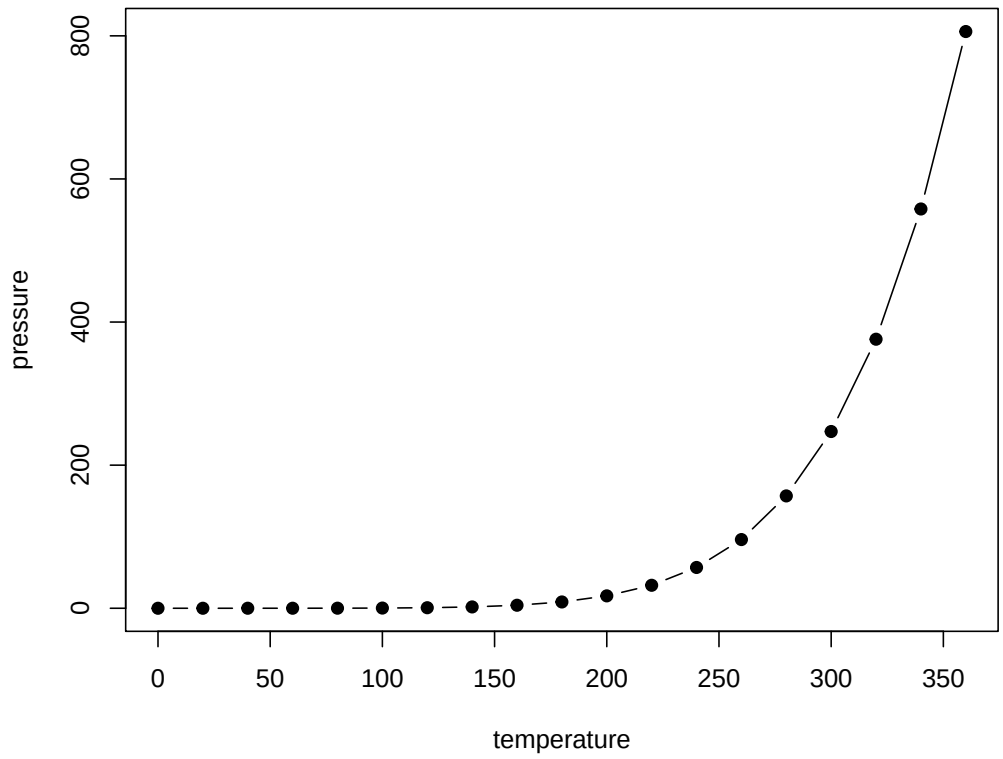


Figure 2.1: Here is a nice figure!

Table 2.1: Here is a nice table!

| Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | setosa |
| 4.9 | 3.0 | 1.4 | 0.2 | setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | setosa |
| 5.0 | 3.6 | 1.4 | 0.2 | setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | setosa |
| 5.0 | 3.4 | 1.5 | 0.2 | setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | setosa |
| 5.4 | 3.7 | 1.5 | 0.2 | setosa |
| 4.8 | 3.4 | 1.6 | 0.2 | setosa |
| 4.8 | 3.0 | 1.4 | 0.1 | setosa |
| 4.3 | 3.0 | 1.1 | 0.1 | setosa |
| 5.8 | 4.0 | 1.2 | 0.2 | setosa |
| 5.7 | 4.4 | 1.5 | 0.4 | setosa |
| 5.4 | 3.9 | 1.3 | 0.4 | setosa |
| 5.1 | 3.5 | 1.4 | 0.3 | setosa |
| 5.7 | 3.8 | 1.7 | 0.3 | setosa |
| 5.1 | 3.8 | 1.5 | 0.3 | setosa |

# Chapter 3

# Continuous Integration and Build Automation

Did you ever wonder what the green/yellow/red 'badges' in some Readme.md files on, e.g., Github.com actually mean? How are they created, what are they for and why should you care?

This chapter will hopeflly shed light on the meaning of some of these badges (those refering to a 'build status') and you will learn how to use these techniques for you own repositories. The key term here is 'continuous integration' (CI) which refers to a concept in software development where all working copies (in git one would refer to branches) of a project are frequently integrated into the mainline (in git terms: the master branch). The rationale being that frequent/continuous integration prevents diverging development branches. Since the philosophy of git is to create feature branches for small, contained changes of master which are to be merged back as soon as possible CI and git are a natural fit.

In practice, however, frequent changes to master are dangerous. After all, the master branch should maintain the last at least working if not stable version of the project that other feature branches can be started from. It is thus crucial to prevent erroneous changes to be merged into the master branch too often. This means that CI requires some kind of automated quality checks that preemptively check for new bugs/problems before a pull request from a feature branch on master is executed. It is this particular aspect of CI that is most interesting to collaborative work on scientific coding projects - being able to automatically run checks/tests on pull requests proposing changes to the master branch of the project.

[Random thought: we should have an example repository for demonstraiting the different states of PRs etc. instead of just including pictures. Readers could then inspect the 'frozen' repository directly and see the PRs etc.!]

To enable this kind of feature on Github, a cloud-based farm or build servers is required where users can run build scripts in virtual machines and retrieve reports on the build status (0 worked, 1 failed). It is these build-statuses that the green/yellow/red badges report visually (yellow/gray being a pending build)! There are multiple companies offering these services (within reasonable bounds) for free for public repositories and as of 2018 the free academic account for GitHub also enables free builds using TravisCI for private repositories. It must be stressed though that, since everything is running in the cloud, the same constraints as for storing data on GitHub servers apply to downloading or processing data in buildscripts for CI services. [point to Jenkins as on-premis solution]

The obvious setting to use automated builds in is package development. This is by far the most common application and the current tools are definitely geared towards that use case. We will later discuss how to extend the scope to non-package situations. For instance, the repository containing the source code for this book also uses TravisCI for build automation even though it is not an R-package itself.

## 3.1 Build automation for R package development

[Todo]

### 3.1.1 Other languages

[Todo]

## 3.2 Some advice on using build automation with analysis projects

[todo]

# Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2018). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.8.