The Live Textbook of Physical Chemistry 1

Dr. Roberto Peverati

2020-07-29

Contents

Chapter 1

Systems and Variables

1.1 Thermodynamic Systems

A thermodynamic system—or just simply a system—is a portion of space with defined boundaries that separate it from its surroundings. The surroundings may include other thermodynamic systems, or physical systems that are not thermodynamic systems. A boundary may be a real physical barrier or a purely notional one. Typical examples of systems are reported in Figure 1.1 below.

Figure 1.1: Examples of Thermodynamic Systems

In the first case, a liquid is contained in a typical erlenmeyer flask. The boundaries of the system are clearly the glass walls of the beaker. The second system is represented by the gas contained in a balloon. The boundary is a physical

barrier also in this case. The third case is a that of a thunder cloud, where the boundary is not a well-defined physical barriers, but rather some condition of pressure and chemical composition at the interface between the cloud and the atmosphere. Finally the fourth case is the case of an open flame. In this case the boundary is again non-physical, and possibly even harder to define than for a cloud. For example, we can chose to define the flame based on some temperature threshold, or some color criterion, or even some chemical one. despite the lack of physical boundaries, both the cloud and the flame—as portions of space containing matter—can be defined as a thermodynamic system and studied as such.

A system can exchange exclusively mass, exclusively energy, or both mass and energy with its surroundings. Depending on the abilities of the boundaries to exchange these quantities, a system is defined as open, closed, or isolated. An open system exchanges both mass and energy. A closed system exchanges only energy, but not mass. Finally, an isolated system does not exchange mass nor energy.

Type of System:	Mass	Energy (either heat or work)
Open	Y	Y
Closed	N	Y
Isolated	N	N

When a system exchanges mass or energy with its surroundings, some of its parameters (variables) change. For example, if a system loses mass to the surroundings, the number of molecules (or moles) in the system will decrease. Similarly, if a system is absorbing some energy one or more of its variables (such as for example its temperature) increase. Mass and energy can flow into the system or out of the system. Let's consider mass exchange only. If some molecules of a substance leave the system and then the same amount of molecules flow back into the system, the system will not be modified (Fig. XXX - system losing molecules, and then gaining them back). In other words, we can count 100 molecules leaving a system and assign them the value of –100 in an outgoing process, and then observe the same 100 molecules going back into the system and assign them a value of +100. Regardless of the number of molecules present in the system in the first place, the overall balance will be –100 (from the outgoing process) +100 (from the ingoing process) = 0, which brings the system to its initial situation (mass has not changed). However, from a mathematical standpoint, we could have equally assigned the label +100 to the outgoing process, and –100 to the ingoing one, and the overall total would have stayed exactly the same: +100–100 = 0. Which of the two labels is best? For this case it seems natural to define a mass going out of the system as negative (the system is losing it), and a mass going into the system as positive (the system is gaining it), but is it as straightforward for energy?

Here's another example. Let's consider a system that is composed of your body. When you are exercising you are losing mass in the form of water (sweat) and CO_2 (from respiration). This mass loss can be easily measured by stepping on a scale before and after exercise. The number you observe on the scale will go down, hence you've lost weight. After exercise, you will reintegrate the lost mass by drinking and eating. If you've reintegrated the same amount you've lost, your weight is going to be exactly the same as it was before the exercise (no weight loss). But which label do you attach to the amounts that you have lost and gained? Let's say that you're running a 5km race without drinking nor eating, and you measure your weight dropping 2 kg after the race. After the race you drink 1.5 kg of water and eat a 500 g energy bar. Overall you didn't lose any weight, and it would seem normal to label the 2 kg that you've lost as negative (–2) and the 1.5 kg of water that you drank and the 500 g bar that you ate as positive (+1.5 +0.5 = +2). But is it the only way? After all, you didn't gain nor lose any weight, so why not calling the 2 kg due to exercise +2 and the 2 that you've ingested as –2? It might seem silly, but mathematically it would not make any difference, the total would still be zero. Now, let's consider energy instead of mass. Let's say that in order to run the 5km race you've spent 500 kcal, which then you reintegrate exactly by eating the energy bar. Which sign would you put in front of the kilocalories that you "burned" during the race? In principle, you've lost them so if you want to be consistent you should use a negative sign. But if you think about it, you've put quite an effort to "lose" those kilocalories, so it might not feel bad to assign them a positive sign instead. After all, it's perfectly OK to say "I've done a 500 kcal run today", while it might sound quite awkward to say "I've done a –500 kcal run today." The results of our previous exercise with mass, in fact, demonstrate that it doesn't really matter which sign you put in front of the quantities. As long as you are consistent throughout the process, the signs will cancel out. That is, if you've done a +500 kcal run, then you've eaten a bar for –500 kcal, resulting in a total zero loss/gain. Alternatively, if you've done a –500 kcal run, you would have eaten a +500 kcal bar, for a total of again zero loss/gain.

These simple examples clearly demonstrate that the sign that we assign to quantities that flow through a boundary is arbitrary (i.e., we can define it any way we want, as long as we are always consistent with ourselves). There is no best way to assign those signs. If you ask two different people, you might obtain two different answers. But we are scientists, and we must make sure to be rigorous. For this reason, chemists have established a convention for the signs that we will follow throughout this course. If we are consistent in following the convention, we are guaranteed to never make any sign mistake.

The chemistry convention of the sign is system-centric.

- If something (energy or mass) goes **into** the system it has a **positive** sign + (system is gaining)
- If something (energy or mass) goes **out of** the system it has a **negative** sign – (system is losing)

If you want a trick to remember the convention, use the weight loss/gain during exercise example above. You are the system, if you lose weight the kilograms will be negative (–2 kg), while if you gain weight they will be positive (+2 kg). Similarly, if you eat an energy bar, you are the system and you will have increased your energy by +500 kcal (positive), while if you burned energy during exercise, you are the system and you will have lost energy, hence –500 kcal (negative). If the system is a balloon filled with gas, and the balloon is losing mass, you are the balloon and you are losing weight, hence the mass will be negative. If the balloon is absorbing heat (likely increasing its temperature, hence increasing its volume), you are the system and you are gaining heat, hence heat will be positive.

1.2 Thermodynamic Variables

The system is defined and studied using parameters that are called variables. These variables are quantities that we are able to measure, such as pressure and temperature, but don't be surprised if in some occasion youu encounter some variable that is a little harder to measure directly, such as entropy. The variables depends only on the current state of the system, and therefore they define it. In other words, if I know the values of all the "relevant variables" of a system, I know the state of the system. The relationship between the variables are described by mathematical functions that are called state functions, while the "relevant variables" are called natural variables. What are the "relevant variables" of a system? The answer to this question depends on the system, and it is not always straightforward. The simplest case is the case of an ideal gas, for which the natural variables are those that enter the ideal gas law and the corresponding equation:

$$PV = nRT \tag{1.1}$$

Therefore, the natural variables for an ideal gas are: the pressure P, the volume V, the number of moles n, and the temperature T, with R being the ideal gas constant. Recalling from the general chemistry courses, R is a universal dimensional constant which has the values of R = 8.31 kJ/mol in SI units.
We will use the ideal gas equation and its variables as an example to discuss about variables and functions in this chapter, and then we will analyze more complicated cases in the next chapters. Variables can be classified according to numerous criteria, each with its advantages and disadvantages. A typical classification is:

- **Physical variables (P, V, T in the ideal gas law):** independent on the chemical composition of the system.
- **Chemical variables (n in the ideal gas law):** dependent on the chemical composition of the system.

Another useful classification is:

- **Intensive variables (P, T in the ideal gas law):** independent on the physical size (extension) of the system.
- **Extensive variables (V, n in the ideal gas law):** dependent on the physical size (extension) of the system.

When we deal with thermodynamic systems, it is more convenient to work with intensive variables. Luckily, it is relatively easy to convert extensive variables into intensive ones by just taking the ratio between two of them. For an ideal gas, by taking the ratio between V and n, we obtained the intensive variable called molar volume:

$$V_m = \frac{V}{n}. \tag{1.2}$$

We can then recast eq. (1.1) as:

$$PV_m = RT, \tag{1.3}$$

which is the preferred equation that we will use for the remainder of this course. The ideal gas equation connects the 3 variables pressure, molar volume, and temperature, and reduces the number of independent variables to just 2. In other words, once 2 of the 3 variables are known, the other one can be easily obtained using these simple relations:

$$P(T, V_m) = \frac{RT}{V_m}, \tag{1.4}$$

$$V_m(T, P) = \frac{RT}{P}, \tag{1.5}$$

$$T(P, V_m) = \frac{PV_m}{R}. \tag{1.6}$$

These equations define three state functions, each one expressed in terms of two independent natural variables. For example, eq. (1.4) defines the state function called "pressure", expressed as a function of temperature and molar volume. Similarly, eq. (1.5) defines the "molar volume" as a function of temperature and pressure, and eq. (1.6) defines the "temperature" as a function of pressure and molar volume. When we know the natural variables that define a state function, we can express the function using its total differential, for example for the pressure $P(T, V_m)$:

$$dP = \left(\frac{\partial P}{\partial T}\right) dT + \left(\frac{\partial P}{\partial V_m}\right) dV_m \tag{1.7}$$

Recalling Schwartz's theorem, the mixed partial second derivatives that can be obtained from eq. 1.2.7 are the same:

$$\frac{\partial^2 P}{\partial T \partial V_m} = \frac{\partial}{\partial V_m}\frac{\partial P}{\partial T} = \frac{\partial}{\partial T}\frac{\partial P}{\partial V_M} = \frac{\partial^2 P}{\partial V_m \partial T} \tag{1.8}$$

Which can be easily verified considering that:

$$\frac{\partial}{\partial V_m}\frac{\partial P}{\partial T} = \frac{\partial}{\partial V_m}\left(\frac{R}{V_m}\right) = -\frac{R}{V_m^2} \tag{1.9}$$

and

$$\frac{\partial}{\partial T}\frac{\partial P}{\partial V_m} = \frac{\partial}{\partial T}\left(\frac{-RT}{V_m^2}\right) = -\frac{R}{V_m^2} \tag{1.10}$$

While for the ideal gas law all the variables are "well-behaved" and always satisfies Schwartz's theorem, we will encounter some variable for which Schwartz's theorem does not hold. Mathematically, if the Schwartz's theorem is violated (i.e., if the mixed second derivatives are not equal), then the corresponding function cannot be integrated, hence it is not a state function. The differential of a function that cannot be integrated cannot be defined exactly. For this reason, these functions are called path functions, that is, they depend on the path rather than the state. The most typical example of path functions that we will encounter in the next chapters are heat (Q) and work (W). For these functions we cannot define exact differentials dQ and dW, and we must introduce a new notation to define their "inexact differentials" δQ and δW.

We will return on exact and inexact differential when we discuss heat and work, but for this chapter it is important to notice the difference between a state function and a path function. A typical example to understand the difference between state and path function is to consider the distance between two geographical locations. Let's for example consider the distance between New York City and Los Angeles. If we fly straight from one city to the other there are roughly 4,000 km between them. This "air distance" depends exclusively on the geographical location of the two cities and it stays constant regardless of the method of transportation that I have accessibility to travel between them. Since the positions of the cities depend uniquely on their latitudes and longitudes, the "air distance" is a state functions, i.e., it is uniquely defined from a simple relationship between measurable variables. However, the "air distance" is not the distance that I will practically have to drive when I go from NYC to LA. Such "travel distance" depends on the method of transportation that I

decide to take (airplane vs. car vs. train vs. boat vs. ...), and it will depend on a plentiful amount of other factors such as the choice of road to be traveled (if going by car), the atmospheric conditions (if flying), and so on. A typical "travel distance" by car is, for example, about 4,500 km, which is about 12% more than the "air distance". Certainly we could even design a very inefficient road trip that avoids all highways and will result in a "travel distance" of 8,000 km or even more (200% of the "air distance"). The "travel distance" is a clear example of a path function because it depends on the specific path that I decide to travel to go from NYC to LA. See Figure 1.2.

Figure 1.2: State Functions vs. Path Functions

Chapter 2

Prerequisites

This GitBook is created in Rstudio, using the `bookdown` package. To get your system set up correctly, you have to install several software packages, and register on GitHub. You only have to perform these steps once, and the entire process should take approximately 1 hour if you start from scratch. In case some software is already installed on your system, you can skip related steps. Follow these steps in order:

1. Install R from cloud.r-project.org
2. Install Rstudio Desktop (Free) from rstudio.com
3. Install Git from git-scm.com. Use the default, recommended settings. It is especially important to leave these settings selected:
    - Git from the command line and also from third party software
    - Use the OpenSSL library
    - Checkout Windows-style, commit Unix-style line endings
    - Enable Git Credential Manager
    - If you run into any trouble, a more comprehensive tutorial on installing Git is available at happygitwithr.com.
4. Register on GitHub
    - Go to https://github.com/ and click "Sign up". Choose an "Individual", "Free" plan.
    - Request a free academic upgrade. This allows you to create *private repositories*, which are only visible to you and selected collaborators, and can be made public when your work is published.

5. Connect Rstudio to Git and Github (for more support, see this Rstudio article, and this blog post)
    a. Open Rstudio, open the Tools menu, click *Global Options*, and click *Git/SVN*
    b. Verify that *Enable version control interface for RStudio projects* is selected
    c. Verify that *Git executable:* shows the location of git.exe. If it is missing, manually fix the location of the file.
    d. Click *Create RSA Key*. Do not enter a passphrase. Press *Create*. A window with some information will open, which you can close.
    e. Click *View public key*, and copy the entire text to the clipboard.
    f. Close Rstudio (it might offer to restart by itself; this is fine)
    g. Go to https://github.com
    h. Click your user icon, click *Settings*, and then select the *SSH and GPG keys* tab.
    i. Click *New SSH key*. Give it an arbitrary name (e.g., your computer ID), and paste the public key from your clipboard into the box labeled "*Key*".
    j. Open Rstudio again (unless it restarted by itself)
6. Install all required packages by running the following code in the Rstudio console. Be prepared for three contingencies:
    - If you receive any error saying *There is no package called XYZ*, then run the code `install.packages("XYZ")`
    - If you are prompted to update packages, just press `3: None`. Updating packages this way in an interactive session sometimes leads to errors, if the packages are loaded.
    - If you see a pop-up dialog asking *Do you want to install from sources the package which needs compilation?*, click *No*. If this leads to errors, then please follow *Step 3* from this online guide, and run `install.packages("devtools")`. This will take a long time, but will allow you to install packages from source.

Run the following code to install the required packages:

```
install.packages("bookdown")
install.packages("tinytex")
tinytex::install_tinytex()
git2r::config(global = TRUE, user.name = "your.name", user.email = "your.email")
```

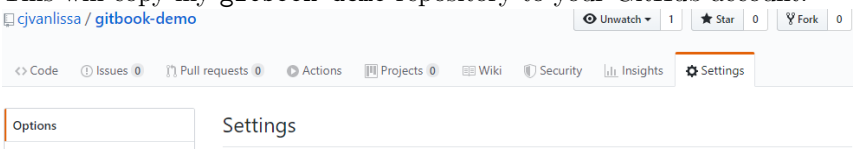
That's it! Everything should be installed and connected now.

Chapter 3

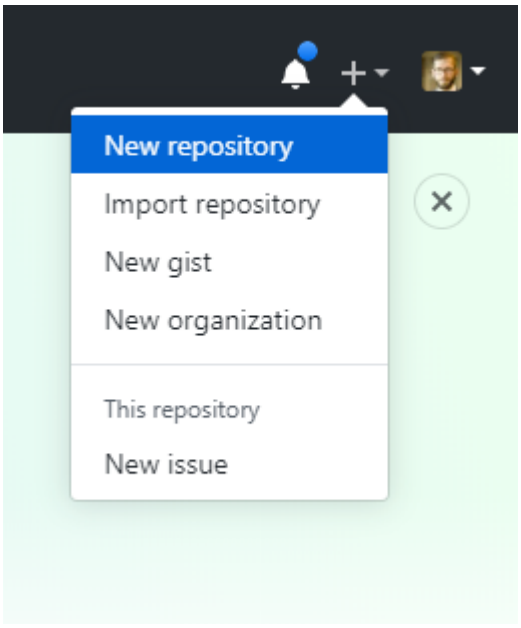
Get your GitBook

To get your GitBook, you should follow these steps:

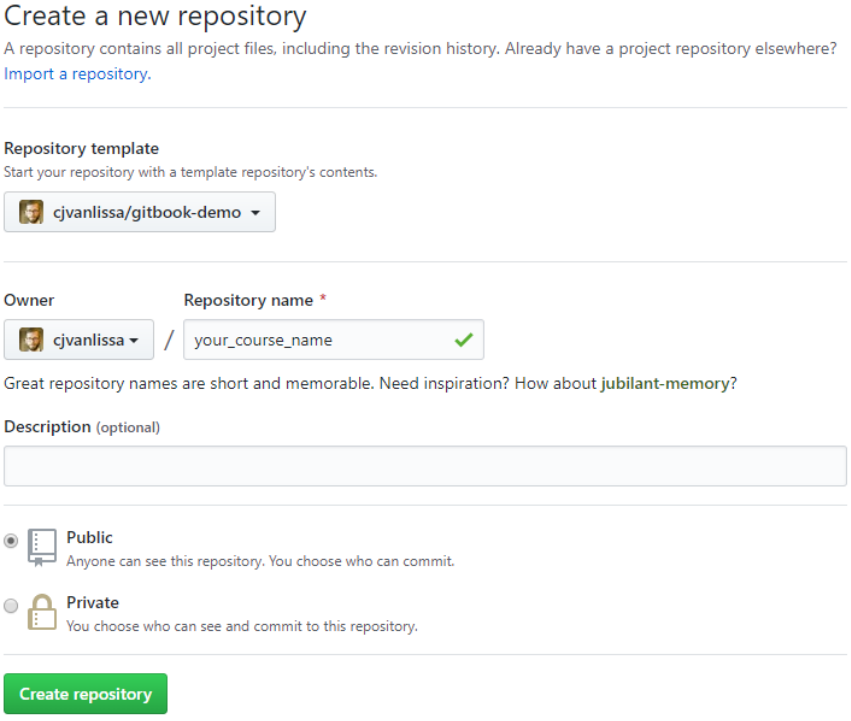
1. Go to https://github.com/cjvanlissa/gitbook-demo
2. In the top right of the page, click `Fork`.
   This will copy my `gitbook-demo` repository to your GitHub account.



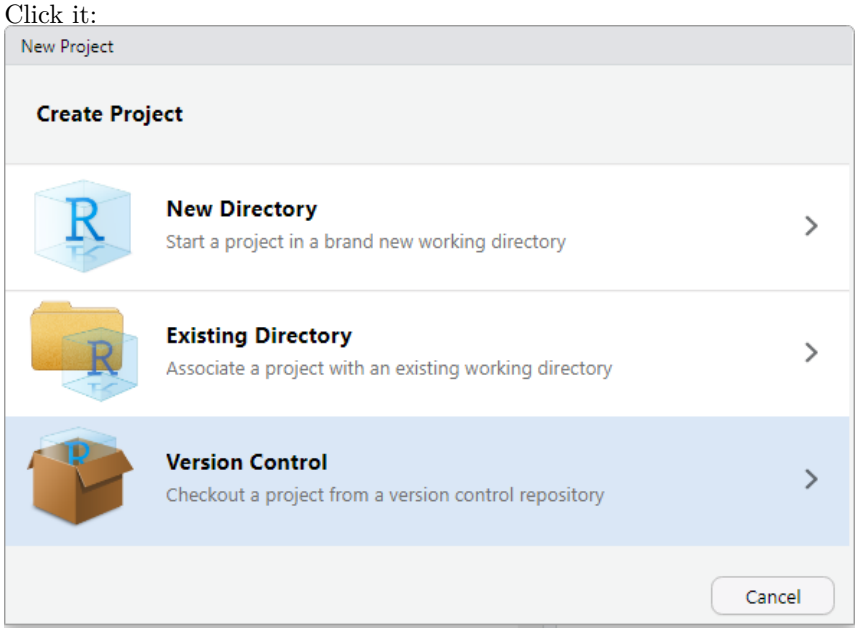
3. My repository is now copied to your account. It is a template repository, which means that you can create a *new repository* based on this one.
4. Create a new repository for your own GitBook. Create one for a course you've been wanting to update. In the top-right corner of the GitHub website, click the + icon, and select "New repository":



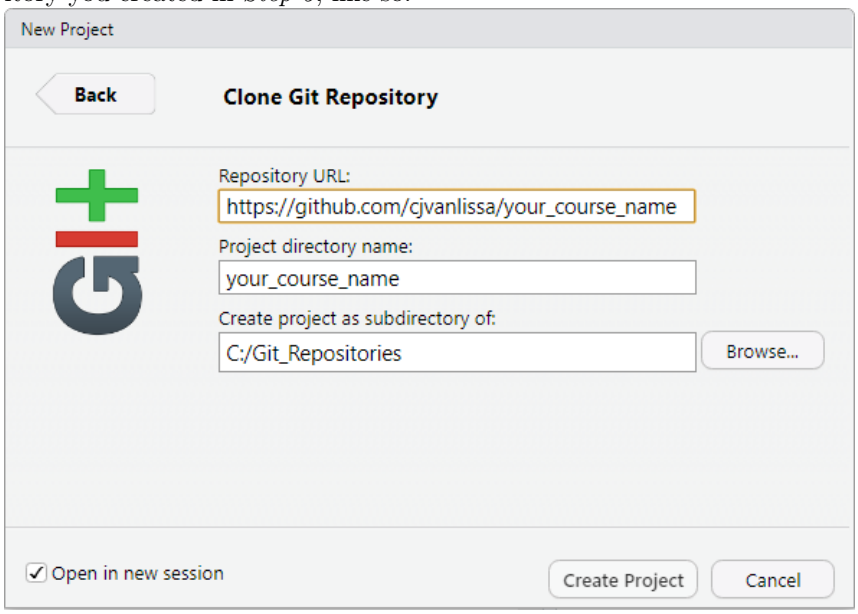
5. In the dialog, select the `gitbook-demo` as "Repository template", and give the repository an appropriate name for your course. Then, press `Create repository`:



6. Now, go back to Rstudio on your computer. In Rstudio, click `File > New Project`. A dialog will open. If you set up Rstudio with Git correctly, the dialog should have an option to create a new project from Version control. Click it:



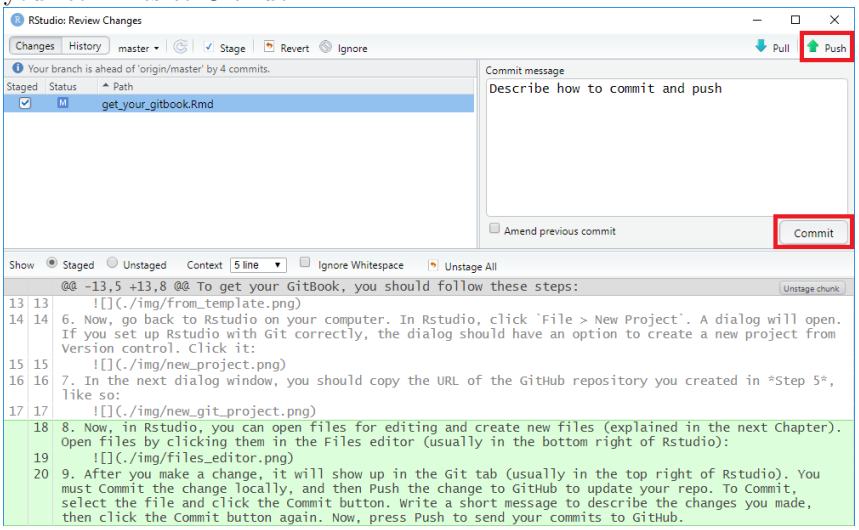
7. In the next dialog window, you should copy the URL of the GitHub repository you created in *Step 5*, like so:



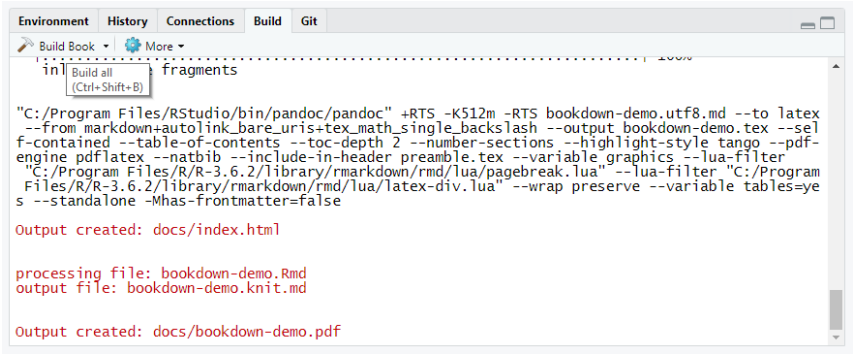
8. Now, in Rstudio, you can open files for editing and create new files (ex-

plained in the next Chapter). Open files by clicking them in the Files editor (usually in the bottom right of Rstudio):

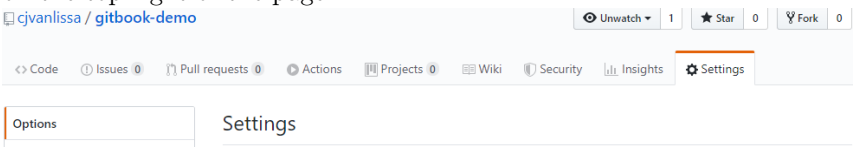
9. After you make a change, it will show up in the Git tab (usually in the top right of Rstudio). You must Commit the change locally, and then Push the change to GitHub to update your repo. To Commit, select the file and click the Commit button. Write a short message to describe the changes you made, then click the Commit button again. Now, press Push to send your commits to GitHub.



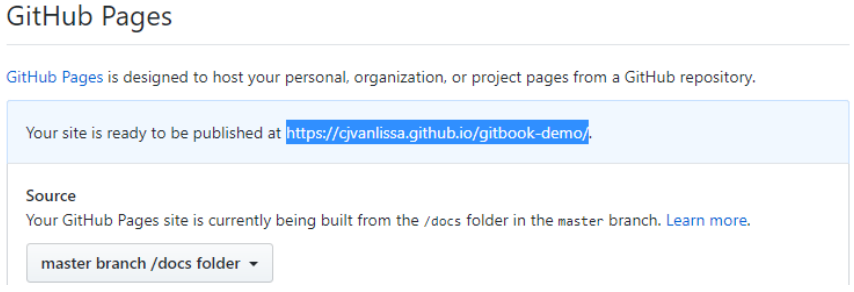
10. To render your book as a GitBook, you must "Build" it. In the top-right panel of Rstudio, you see a "Build" tab. In this tab, simply click the "Build Book" button to build your book. You should see a lot of rendering messages, until a window pops up with your brand new GitBook. If you get errors at this stage, you probably made a mistake in preparing your system (see the previous Chapter).



11. Building the book generated a lot of new files in the `./docs` directory. This directory contains the website files for your GitBook. Open the Git tab again, verify that the `./docs` directory is listed, and Commit and Push all of these new files as described in *Step 9*.
12. There is only one last remaining task: To publish your GitBook on GitHub pages. Once you do this, any change to the `./docs` folder that you push to GitHub will lead to an immediate update of your GitBook website. Go back to the GitHub page for your Repository. Click on the `Settings` tab on the top right of the page:



13. On the Settings page, scroll all the way down until you reach a section called `GitHub Pages`. There, under the "Source" heading, click the word `None`, and select `master branch /docs folder`. When you select it, the page will update, and if you scroll back down to the `GitHub Pages` section, you will see the URL where your GitBook is published. The first time, it will take a few minutes for your GitBook to come online. When you publish updates to the GitBook however (simply by following *Step 11* again), the update will be near-instantaneous. The Pages section should now look like this (and that is hopefully the link where you found this book):



Chapter 4

Editing the book

The contents of the book are written in **RMarkdown**. You can use any formatting code that Pandoc's Markdown supports, e.g., a math equation $a^2 + b^2 = c^2$. Moreover, you can include chunks of R-code, like this:

The results of these chunks can be rendered to the GitBook:

```
## [1] "This is an R-command!"
```

To edit the book, you can change the text in the `.Rmd` files. Each Rmd file should contain **one and only one** chapter. A chapter is defined by the first-level heading `#`, e.g., `# Editing the book`.

Any sub-headings within the chapter are indicated with several `#` signs, e.g., `##` (level 2) and `###` (level 3).

4.1 Creating new chapters

To create a new chapter, you must follow two steps: 1) Create the file, and 2) Include it in the list of chapters.

First, to create the file for a new chapter in Rstudio, click `File > New File > Text file`. At the top of the file, write your chapter heading, as explained above. Then, click `File > Save`. Save the file as `.Rmd`, without spaces in the file name, e.g.: `editing_the_book.Rmd`.

Second, to include it in the list of chapters, open the file `_bookdown.yml` (click it in the Files explorer in the bottom right of Rstudio). This file has a list of `.Rmd` files to be included in the book. In this example, the list looks like this:

```
tmp <- readLines("_bookdown.yml")
cat(tmp[grep("^rmd_files", tmp):grep("references\\.Rmd", tmp)], sep = "\n")
```

rmd_files: [“index.Rmd”, “Basis.Rmd”, “prerequisites.Rmd”, “get_your_gitbook.Rmd”, “editing_the_book.Rmd”, “figures_tables.Rmd”, “examples.Rmd”, “open_educational.Rmd”, “use_in_course.Rmd”, “licenses.Rmd”, “references.Rmd”]

Insert the file name of your new chapter in the desired position in this list.

4.2 Linking across chapters

You can label chapter and section titles using `{#label}` after them. The labels can be used as cross-references. For example, we can link to Chapter 5. If you do not manually label chapters, there will be automatic labels anyway, e.g., Chapter 6.

4.3 Advanced editing

The convenient Rmarkdown Cheat Sheet by Rstudio covers most of the knowledge required for advanced Rmarkdown editing. You can print it out and stick it to your wall!

Chapter 5

Figures and tables

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)
```

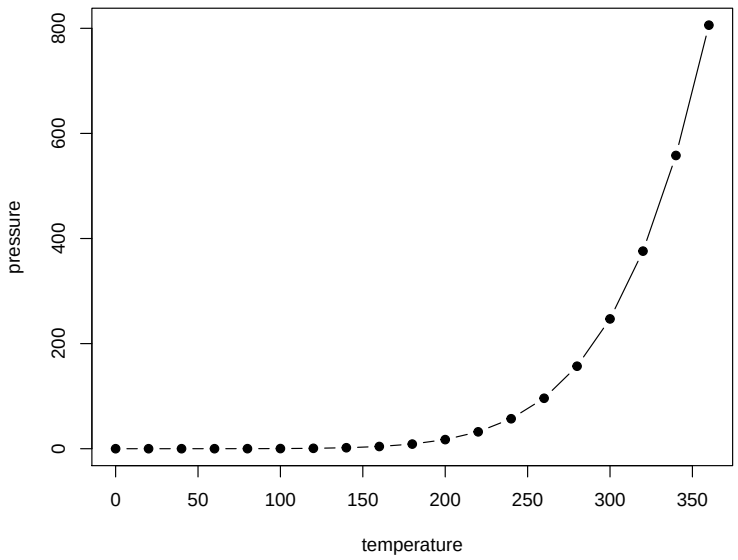


Figure 5.1: Here is a nice figure!

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 5.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 5.1.

Table 5.1: 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

```
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, 2020) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Chapter 6

Examples

Here are some examples of other GitBooks for courses; if you want to have your GitBook added to the list, please send a Pull Request (here's how to send a pull request).

6.1 Statistics with R (H. Quene)

https://hugoquene.github.io/emlar2020

A GitBook for a tutorial on *Statistics with R (Basics)*, held as part of the workshop on Experimental Methods in Language Acquisition Research (EMLAR, https://emlar.wp.hum.uu.nl/), Utrecht, on 17 April 2020. This compact introduction helps you with your first steps into R.

6.2 Theory Construction and Statistical Modeling (C. J. van Lissa)

http://cjvanlissa.github.io/TCSM

A GitBook for the course *"Theory Construction and Statistical Modeling"*, with some interesting code, for example: Blocks of answers to the tutorial questions that can be collapsed and expanded.

6.3 Doing Meta-Analysis in R (C. J. van Lissa)

http://cjvanlissa.github.io/Doing-Meta-Analysis-in-R

A GitBook on doing meta-analysis in R, based on the book 'Doing Meta-Analysis in R', by Mathias Harrer, Pim Cuijpers, & David Ebert, and adapted to focus on the metafor package, and exploring heterogeneity using metaforest. The original can be found here: https://bookdown.org/MathiasHarrer/Doing_Meta_Analysis_in_R/

6.4 Métodos quantitativos em Psicologia com R (L. Anunciação)

https://anovabr.github.io/mqt/

This book provides a short and to-the-point exposition on the essentials of statistics, and was written for undergraduate students at the Pontifical Catholic University of Rio de Janeiro (PUC-Rio). To a lesser degree, the mathematical modeling of statistical questions will be addressed. This book might be relevant for Portuguese-speaking students who enroll for laboratory-based statistics and anyone who wants to learn R.

Chapter 7

Open Educational Resources

UNESCO defines Open Educational Resources as *teaching, learning and research materials in any medium – digital or otherwise – that reside in the public domain or have been released under an open license that permits no-cost access, use, adaptation and redistribution by others with no or limited restrictions.*

Open Educational resources can help lighten the workload on individual teachers, who can collaborate with the development of high-quality open access resources, instead of having to develop their own proprietary materials from scratch. Moreover, Open Educational resources are inclusive, lowering the barrier to knowledge acquisition for learners around the world, and enabling lifelong learning for those outside academia.

Many universities support Open Educational Resources. Here are just a few (feel free to send a pull request with other relevant resources).

- **OER Commons**: A freely accessible online library of open educational resources.
- **Utrecht University Figshare**: Open learning objects from Utrecht University.
- **Johns Hopkins University OCW**: Open public health courses and materials.
- **University of Pittsburgh OER**: Big List of Open Educational Resources.
- **MERLOT**: Online learning and support materials and content creation tools, led by an international community of educators, learners and researchers.

Chapter 8

Compatibility with existing systems

Many universities offer digital platforms for learning. You might wish to embed your GitBook within these existing systems. Here are two ways in which you might do that. Currently, this section only discusses BlackBoard, but the same principles should apply to other platforms.

8.1 Add a hyperlink

You can add a link to your GitBook in the BlackBoard course menu by following this tutorial.

8.2 Embed the whole book

You can add a Blank Page to your BlackBoard course menu, and fill that page with a full-size "iframe" - a web page within the web page. This tutorial explains how to do it. It is possible that your university is blocking this feature, however.

Chapter 9

License your GitBook

In the spirit of Open Science, it is good to think about making your course materials Open Source. That means that other people can use them. In principle, if you publish materials online without license information, you hold the copyright to those materials. If you want them to be Open Source, you must include a license. It is not always obvious what license to choose.

The Creative Commons licenses are typically suitable for course materials. This GitBook, for example, is licensed under CC-BY 4.0. That means you can use and remix it as you like, but you must credit the original source.

If your project is more focused on software or source code, consider using the GNU GPL v3 license instead.

You can find more information about the Creative Commons Licenses here. Specific licenses that might be useful are:

- CC0 ("No Rights Reserved"), everybody can do what they want with your work.
- CC-BY 4.0 ("Attribution"), everybody can do what they want with your work, but they must credit you. Note that this license may not be suitable for software or source code!

For compatibility between CC and GNU licenses, see this FAQ.

Bibliography

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

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.20.