

# Collaborative Data Science Practices

*Will Beasley*

*2019-01-22*



# Contents

<b>1</b>	<b>Prerequisites</b>	<b>7</b>
<b>2</b>	<b>Architecture Principles</b>	<b>9</b>
2.1	Encapsulation . . . . .	9
2.2	Leverage team member's strenghts & avoid weaknesses . . . . .	9
2.3	Scales . . . . .	9
2.4	Consistency . . . . .	9
<b>3</b>	<b>Prototypical File</b>	<b>11</b>
3.1	Clear Memory . . . . .	11
3.2	Load Sources . . . . .	11
3.3	Load Packages . . . . .	11
3.4	Declare Globals . . . . .	11
3.5	Load Data . . . . .	11
3.6	Tweak Data . . . . .	11
3.7	(Unique Content) . . . . .	11
3.8	Verify Values . . . . .	11
3.9	Specify Output Columns . . . . .	11
3.10	Save to Disk or Database . . . . .	11
<b>4</b>	<b>Prototypical Repository</b>	<b>13</b>
4.1	Analysis . . . . .	13
4.2	Data Public . . . . .	13
4.3	Data Unshared . . . . .	13
4.4	Documentation . . . . .	13
4.5	Manipulation . . . . .	13
4.6	Stitched Output . . . . .	13
4.7	Utility . . . . .	13

<b>5</b>	<b>Data at Rest</b>	<b>15</b>
5.1	Data States . . . . .	15
5.2	Data Containers . . . . .	15
<b>6</b>	<b>Patterns</b>	<b>17</b>
6.1	Ellis . . . . .	17
6.2	Arch . . . . .	17
6.3	Ferry . . . . .	17
6.4	Scribe . . . . .	17
6.5	Analysis . . . . .	17
6.6	Presentation -Static . . . . .	17
6.7	Presentation -Interactive . . . . .	17
6.8	Metadata . . . . .	17
<b>7</b>	<b>Security &amp; Private Data</b>	<b>19</b>
7.1	File-level permissions . . . . .	19
7.2	Database permissions . . . . .	19
7.3	Public & Private Repositories . . . . .	19
<b>8</b>	<b>Automation</b>	<b>21</b>
8.1	Flow File in R . . . . .	21
8.2	Makefile . . . . .	21
8.3	SSIS . . . . .	21
8.4	cron Jobs & Task Scheduler . . . . .	21
8.5	Sink Log Files . . . . .	21
<b>9</b>	<b>Scaling Up</b>	<b>23</b>
9.1	Data Storage . . . . .	23
9.2	Data Processing . . . . .	23
<b>10</b>	<b>Parallel Collaboration</b>	<b>25</b>
10.1	Social Contract . . . . .	25
10.2	Code Reviews . . . . .	25
10.3	Remote . . . . .	25

<b>11 Documentation</b>	<b>27</b>
11.1 Team-wide . . . . .	27
11.2 Project-specific . . . . .	27
11.3 Dataset Origin & Structure . . . . .	27
11.4 Issues & Tasks . . . . .	27
11.5 Flow Diagrams . . . . .	27
11.6 Setting up new machine . . . . .	27
<b>12 Publishing Results</b>	<b>29</b>
12.1 To Other Analysts . . . . .	29
12.2 To Researchers & Content Experts . . . . .	29
12.3 To Technical-Phobic Audiences . . . . .	29
<b>13 Testing, Validation, &amp; Defensive Programming</b>	<b>31</b>
13.1 Testing Functions . . . . .	31
13.2 Defensive Programming . . . . .	31
13.3 Validator . . . . .	31
<b>14 Troubleshooting and Debugging</b>	<b>33</b>
14.1 Finding Help . . . . .	33
14.2 Debugging . . . . .	33
<b>15 Considerations when Selecting Tools</b>	<b>35</b>
15.1 Required Installation . . . . .	35
15.2 Recommended Installation . . . . .	35
15.3 Optional Installation . . . . .	35
15.4 Asset Locations . . . . .	35
<b>16 Considerations when Selecting Tools</b>	<b>37</b>
16.1 General . . . . .	37
16.2 Languages . . . . .	37
16.3 R Packages . . . . .	37
16.4 Database . . . . .	37
<b>17 Growing a Team</b>	<b>39</b>
17.1 Recruiting . . . . .	39
17.2 Training to Data Science . . . . .	39
17.3 Bridges Outside the Team . . . . .	39
<b>18 Introduction</b>	<b>41</b>

<b>19 Scratch Pad of Loose Ideas</b>	<b>43</b>
19.1 Chapters & Sections to Form . . . . .	43

# Chapter 1

## Prerequisites

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports, e.g., a math equation  $a^2 + b^2 = c^2$ .

The **bookdown** package can be installed from CRAN or Github:

```
install.packages("bookdown")  
# or the development version  
# devtools::install_github("rstudio/bookdown")
```

Remember each Rmd file contains one and only one chapter, and a chapter is defined by the first-level heading #.

To compile this example to PDF, you need XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): <https://yihui.name/tinytex/>.





## Chapter 2

# Architecture Principles

### 2.1 Encapsulation

### 2.2 Leverage team member's strenghts & avoid weaknesses

1. Focused code files
2. Metadata for content experts

### 2.3 Scales

1. Single source & single analysis
2. Multiple sources & multiple analyses

### 2.4 Consistency

1. Across Files {#consistency-files}
2. Across Languages
3. Across Projects



## Chapter 3

# Prototypical File

As stated before, in Consistency Files, using a consistent file structure can (a) improve the quality of the code because the structure has been proven over time to facilitate good practices and (b) allow your intentions to be more clear to teammates because they are familiar with the order and intentions of the chunks.

We use the term “chunk” for a section of code because it corresponds with knitr terminology (Xie, 2015), and in many cases, the chunk of our R file connects to a knitr Rmd file.

### 3.1 Clear Memory

### 3.2 Load Sources

### 3.3 Load Packages

### 3.4 Declare Globals

### 3.5 Load Data

### 3.6 Tweak Data

### 3.7 (Unique Content)

### 3.8 Verify Values

### 3.9 Specify Output Columns

### 3.10 Save to Disk or Database



## Chapter 4

# Prototypical Repository

<https://github.com/wibeasley/RAnalysisSkeleton>

### 4.1 Analysis

### 4.2 Data Public

1. Raw
2. Derived
3. Metadata
4. Database
5. Original

### 4.3 Data Unshared

### 4.4 Documentation

### 4.5 Manipulation

### 4.6 Stitched Output

### 4.7 Utility



# Chapter 5

## Data at Rest

### 5.1 Data States

1. Raw
2. Derived
  1. Project-wide File on Repo
  2. Project-wide File on Protected File Server
  3. User-specific File on Protected File Server
  4. Project-wide Database
3. Original

### 5.2 Data Containers

1. csv
2. rds
3. SQLite
4. Central Enterprise database
5. Central REDCap database
6. Containers to avoid for raw/input
  1. Proprietary like xlsx, sas7bdat





## Chapter 6

# Patterns

6.1 Ellis

6.2 Arch

6.3 Ferry

6.4 Scribe

6.5 Analysis

6.6 Presentation -Static

6.7 Presentation -Interactive

6.8 Metadata



## Chapter 7

# Security & Private Data

### 7.1 File-level permissions

### 7.2 Database permissions

### 7.3 Public & Private Repositories

#### 7.3.1 Scrubbing GitHub history

Occasionally files may be committed to your git repository that need to be removed completely. Not just from the current collections of files (*i.e.*, the branch's head), but from the entire history of the repo.

Scrubbing is required typically when (a) a sensitive file has been accidentally committed and pushed to GitHub, or (b) a huge file has bloated your repository and disrupted productivity.

The two suitable scrubbing approaches both require the command line. The first is the `git-filter-branch` command within git, and the second is the BFG repo-cleaner. We use the second approach, which is [recommended by GitHub]; it requires 15 minutes to install and configure from scratch, but then is much easier to develop against, and executes much faster.

The bash-centric steps below remove any files from the repo history called 'monster-data.csv' from the 'bloated' repository.

1. If the file contains passwords, change them immediately.
2. Delete 'monster-data.csv' from your branch and push the commit to GitHub.
3. Ask your collaborators to push any outstanding commits to GitHub and delete their local copy of the repo. Once scrubbing is complete, they will re-clone it.
4. Download and install the most recent Java JRE from the Oracle site.
5. Download the most recent jar file from the BFG site to the home directory.
6. Clone a fresh copy of the repository in the user's home directory. The `--mirror` argument avoids downloading every file, and downloads only the bookkeeping details required for scrubbing.

```
cd ~  
git clone --mirror https://github.com/your-org/bloated.git
```

7. Remove all files (in any directory) called ‘monster-data.csv’.

```
java -jar bfg-*.jar --delete-files monster-data.csv bloated.git
```

8. Reflog and garbage collect the repo.

```
cd bloated.git  
git reflog expire --expire=now --all && git gc --prune=now --aggressive
```

9. Push your local changes to the GitHub server.

```
git push
```

10. Delete the bfg jar from the home directory.

```
cd ~  
rm bfg-*.jar
```

11. Ask your collaborators to reclone the repo to their local machine. It is important they restart with a fresh copy, so the once-scrubbed file is not reintroduced into the repo’s history.
12. If the file contains sensitive information, like passwords or PHI, ask GitHub to refresh the cache so the file’s history isn’t accessible through their website, even if the repo is private.

#### 7.3.1.0.1 Resources

- BFG Repo-Cleaner site
- Additional BFG instructions
- GitHub Sensitive Data Removal Policy

## Chapter 8

# Automation

### 8.1 Flow File in R

### 8.2 Makefile

### 8.3 SSIS

### 8.4 cron Jobs & Task Scheduler

### 8.5 Sink Log Files



## Chapter 9

# Scaling Up

### 9.1 Data Storage

1. Local File vs Conventional Database vs Redshift
2. Usage Cases

### 9.2 Data Processing

1. R vs SQL
2. R vs Spark





## Chapter 10

# Parallel Collaboration

### 10.1 Social Contract

1. Issues
2. Organized Commits & Coherent Diffs
3. Branch & Merge Strategy

### 10.2 Code Reviews

1. Daily Reviews of PRs
2. Periodic Reviews of Files

### 10.3 Remote

1. Headset & sharing screens



# Chapter 11

## Documentation

11.1 Team-wide

11.2 Project-specific

11.3 Dataset Origin & Structure

11.4 Issues & Tasks

11.5 Flow Diagrams

11.6 Setting up new machine

(example)



## Chapter 12

# Publishing Results

12.1 To Other Analysts

12.2 To Researchers & Content Experts

12.3 To Technical-Phobic Audiences



## Chapter 13

# Testing, Validation, & Defensive Programming

### 13.1 Testing Functions

### 13.2 Defensive Programming

1. Throwing errors

### 13.3 Validator

1. Benefits for Analysts
2. Benefits for Data Collectors





## Chapter 14

# Troubleshooting and Debugging

### 14.1 Finding Help

1. Within your group (eg, Thomas and REDCap questions)
2. Within your university (eg, SCUG)
3. Outside (eg, Stack Overflow; GitHub issues)

### 14.2 Debugging

1. `traceback()`, `browser()`, etc



## Chapter 15

# Considerations when Selecting Tools

<https://github.com/OuhscBbmc/RedcapExamplesAndPatterns/blob/master/DocumentationGlobal/ResourcesInstallation.md>

### 15.1 Required Installation

### 15.2 Recommended Installation

### 15.3 Optional Installation

### 15.4 Asset Locations



## Chapter 16

# Considerations when Selecting Tools

### 16.1 General

#### 16.1.1 The Component's Goal

While discussing the advantages and disadvantages of tools, a colleague once said, “Tidyverse packages don’t do anything that I can’t already do in Base R, and sometimes it even requires more lines of code”. Regardless if I agree, I feel these two points are irrelevant. Sometimes the advantage of a tool isn’t to expand existing capabilities, but rather to facilitate development and maintenance for the same capability.

Likewise, I care less about the line count, and more about the readability. I’d prefer to maintain a 20-line chunk that is familiar and readable than a 10-line chunk with dense phrases and unfamiliar functions. The bottleneck for most of our projects is human time, not execution time.

#### 16.1.2 Current Skillset of Team

#### 16.1.3 Desired Future Skillset of Team

#### 16.1.4 Skillset of Audience

### 16.2 Languages

### 16.3 R Packages

### 16.4 Database



## Chapter 17

# Growing a Team

### 17.1 Recruiting

### 17.2 Training to Data Science

1. Starting with a Researcher
2. Starting with a Statistician
3. Starting with a DBA
4. Starting with a Software Developer

### 17.3 Bridges Outside the Team

1. Monthly User Groups
2. Annual Conferences





# Chapter 18

## Introduction

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 18. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter 2.

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 18.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 18.1.

```
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).

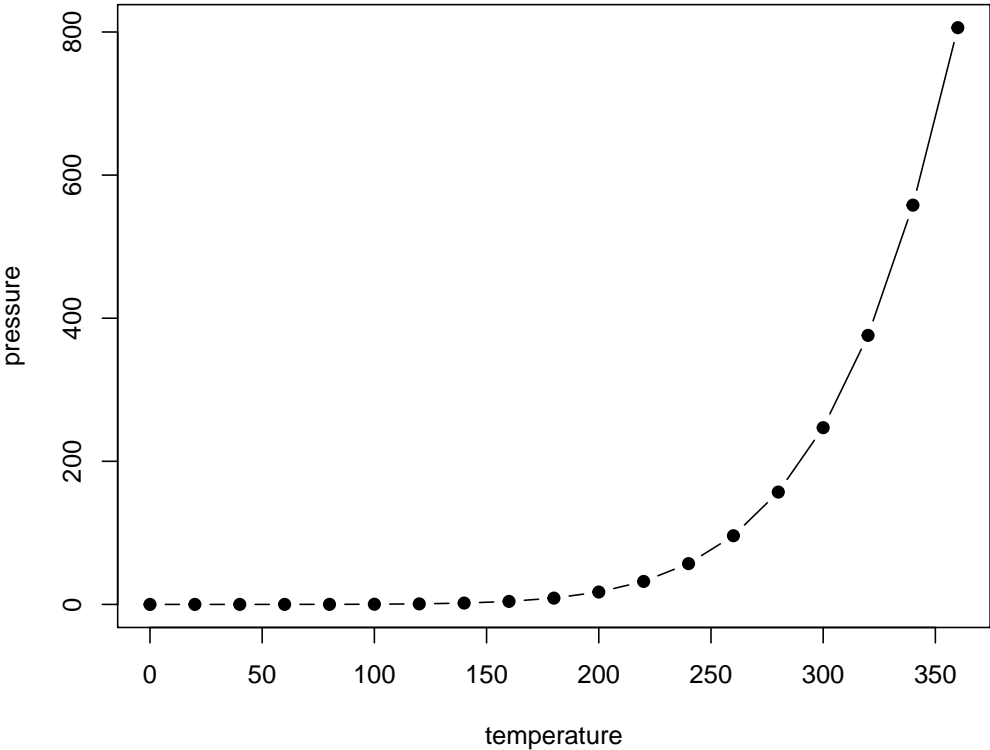


Figure 18.1: Here is a nice figure!

Table 18.1: 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 19

# Scratch Pad of Loose Ideas

### 19.1 Chapters & Sections to Form

1. Tools to Consider
  1. tidyverse
  2. odbc



# 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 0.9.