The Surgical Informatics Cookbook

Surgical Informatics, University of Edinburgh

2024-05-27

Contents


SURGICAL INFORMATI
Better surgical care through data and technology

Chapter 1

Introduction

1.1 How to contribute

(Steps 1. to 3. only need to be done once - to set-up.)

1. Connect your RStudio and GitHub using these instructions, only up to "Create new project" is necessary here (the repository/project already exists): https://www.datasurg.net/2015/07/13/rstudio-and-github/

2. Get your GitHub account added to the surgicalinformatics organisation on GitHub (ask Riinu/Ewen): https://github.com/SurgicalInformatics

3. In RStudio: New Project - Version Control - git, then copy the URL: https://github.com/SurgicalInformatics/cookbook

4. Add your thing by editing the appropriate .Rmd file - there's one for each chapter. In the `Build` pane (next to `Environment`) **click on More - Clean All** (if you don't do this you may be able to compile the book with code that won't work at a subsequent clean build which can be trickier to debug). Use the Build tab to Build your changes into a book.

5. If anyone has pushed since you cloned/last pulled (hopefully they've not been working on the exact same chapter): Make sure you **click on More - Clean All** (as above). Then Pull from the Git tab. This only cleans the output files - html and PDF, it will not touch the changes you've made in the .Rmd file.

6. Then Build Book again - this will include the new changes you pulled as well as your changes.

7. Git tab - commit everything, Push quickly before anyone else does or you'll have to go back to step 5. You can check for new pushed commits here: https://github.com/SurgicalInformatics/cookbook/commits/master Alternatively there's no harm in clicking the Pull button again - it should then say "Already up-to-date".

   Pro tip: instead of clicking on every single file in the Git tab, go to the terminal, `cd cookbook` to go to the project folder if still home, and do `git add .` which is the same thing. Still need to Commit though!

8. Have fun!

1.2 Indexing

1.2.1 Index

Bold index headings:
`\index{linear regression@\textbf{linear regression}}` (ticks in .Rmd file are excluded when actually using)

Sub-entries of bold headings:
`\index{linear regression@\textbf{linear regression}!diagnostics}`

Stand-alone entries:
`\index{linear regression}`

1.2.2 Chapter and section references

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter `\@ref(intro)` (ticks in .Rmd are excluded when actually using). If you do not manually label them, there will be automatic labels anyway, e.g., Chapter `\@ref(methods)`.

1.2.3 Figure and table references

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)
```

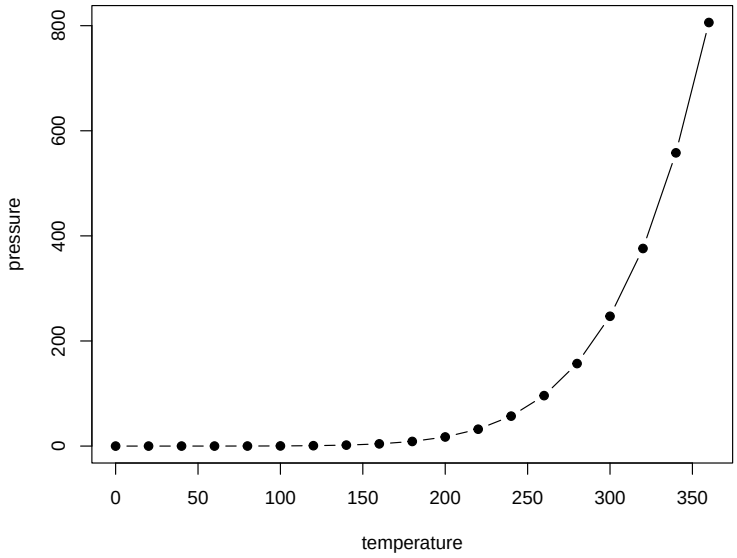


Figure 1.1: Here is a nice figure!

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

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

1.2.4 Citations

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2023) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

Table 1.1: 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 2

Snippets

Random useful snippets that do not fit anywhere else.

2.1 Useful RStudio functions

2.1.1 Clean restart

Cleans current environment and restarts R session (no code can run past this). Indicate objects to leave alone by supplying string to `leave` argument.

```
clean_restart = function(leave = NULL){
  rm(list=setdiff(ls(envir = .GlobalEnv), leave), envir = .GlobalEnv)
  invisible(.rs.restartR())
}
```

2.1.2 Remove duplicated packages between personal and system libraries

This removes any user library packages that are also present in the system library. Use this to minimise loading conflicts. Change `tdrake` in example to your rstudio username.

```
library(tidyverse)

tdrake_packages = installed.packages() %>% #change 'tdrake' to your rstudio username
  as_tibble() %>%
  filter(LibPath %>% str_detect("tdrake")) %>%
```

```
  pull(Package)

system_packages = installed.packages() %>%
  as_tibble() %>%
  filter(LibPath %>% str_detect("local")) %>%
  pull(Package)

to_remove = tdrake_packages[tdrake_packages %in% system_packages]

remove.packages(to_remove)
}
```

2.1.3 Remove large file accidentally committed to git

`git filter-branch -f --index-filter "git rm -rf --cached --ignore-unmatch filename.rda" HEAD`.

2.2 Memory and efficient RAM usage

1. Restart R often
2. Use `rm(...)` to get rid of big files after using them
3. Save large objects (and compress with "gz") that take a while to create but are needed often (`write_rds(x, "big_file.rds.gz", compress = "gz")`)
4. Run `gc()` (garbage collect)
5. Check memory usage: `pryr::mem_used()`

Most importantly if using big files (even if they aren't loaded in your environment!) do the following:

1. `save.image("temp.rda")`
2. Restart R
3. `load("temp.rda")`
4. Delete "temp.rda" from `Files` pane

2.3 Exporting tables that work in both PDF and Word

The `kable()` function from `library(knitr)` formats tables for all 3 output formats, but it is a bit limited. `library(kableExtra)` is great for table customisations that work in PDF and HTML. `kableExtra` functions do not work

for Word output, but `library(flextable)` does work for Word (but not for PDF).

We can define our own table printing function that uses kableExtra or flextable based on output type:

```
# This makes table resize or continue over multiple pages in all output types
# PDF powered by kableExtra, Word by flextable
mytable = function(x, caption = "", longtable = FALSE, ...){
  library(kableExtra)
  # if not latex or html then else is Word
  if (knitr::is_latex_output() | knitr::is_html_output()){
    knitr::kable(x, row.names = FALSE, align = c("l", "l", "r", "r", "r", "r", "r", "r", "r"),
          booktabs = TRUE, caption = caption, longtable = longtable,
          linesep = "", ...) %>%
    kableExtra::kable_styling(latex_options = c("scale_down", "hold_position"))
  }else{
    flextable::flextable(x) %>%
      flextable::autofit() %>%
      flextable::width(j = 1, width = 1.5) %>%
      flextable::height(i = 1, height = 0.5, part = "header")
  }

}
library(dplyr)
cars %>%
  head() %>%
  mytable()
```

2.3.1 Warning in kableExtra. Please specify format in kable. kableExtra can customize either HTML or LaTeX outputs.

This warning goes away when you load `library(kableExtra)` in your script, instead of just using `kableExtra::function()`.

What happens when you load `library(kableExtra)` is that it sets that specifies the format for the R session, so that Warning goes away. It does mean that the `render()` function may fail since it knits in the same environment you have, rather than a new one like the Knit button does.

Table 2.1:

speed	dist
4	2
4	10
7	4
7	22
8	16
9	10

2.4 Creating Reproducible R Examples to Share in the Group (binder, holepunch and docker)

When asking for help with R code having a reproducible example is crucial (some mock data that others can use along with your code to reproduce your error). Often this can be done easily with creation of a small tibble and posting of the code on slack but sometimes it requires more complex data or the error is due to something in the Linux system in which RStudio server is hosted. For example if the `cairo` package for Linux isn't installed then plots don't work. The `holepunch` package helps to reproduce examples like these (not suitable for projects with confidential data).

2.4.1 Create Basic Reproducible Examples

The three main parts of the reproducible example (reprex) in Surgical Informatics are 1. packages, 2. small dataset and 3. code. Other things like R version and Linux version can be assumed as we all use one of only a few servers.

If you have a small (and confidential) set of data in a tibble or data frame called `my_data` and want it to be easily copied run: `dput(droplevels(my_data))`. This will print out code in the console that can be copy-pasted to reproduce the data frame. Alternatively use the `tibble` or `tribble` functions to create it from scratch (this is preferable for simple datasets). Then copy in the packages and finally the code (ideally the least amount possible to generate the error) and share with the group e.g.:

```
library(tidyverse)

# Output generated from dput(droplevels(my_data))
data = structure(list(a = c(1, 2, 3), b = c("a", "b", "c"), c = 10:12), .Names = c("a",
"b", "c"), row.names = c(NA, -3L), class = c("tbl_df", "tbl",
"data.frame"))

data %>%
  mutate(newvar = a /b)
```

```
## Error in `mutate()`:
## i In argument: `newvar = a/b`.
## Caused by error in `a / b`:
## ! non-numeric argument to binary operator
```

2.4.2 `holepunch` - Complex Reproducible Examples

From your project with data you are happy to make public make sure you are backed up to `git` and `GitHub`. See the relevant chapter on how to do this. Then run the following:

```
# Holepunch testing

remotes::install_github("karthik/holepunch")

library(holepunch)
write_compendium_description(package = "Title of my project",
                             description = "Rough description of project or issue")

write_dockerfile(maintainer = "SurgicalInformatics")

generate_badge()

build_binder()
```

The file will generate some text to copy into the top of a `README.md` file. It will look like:

```
<!-- badges: start -->
[![Launch Rstudio Binder](http://mybinder.org/badge_logo.svg)](https://mybinder.org/v2/
<!-- badges: end -->
```

Now, whenever somebody clicks on the badge in the `README` on `GitHub` they will be taken to an RStudio server instance with all your files (excluding files listed in `.gitignore`), all the current versions of your package, all the current Linux packages and the current R version. They can then test your code in an near-identical environment to help identify the source of the error, their session will time out after 10 minutes of inactivity or 12 hours since starting and will not save anything so should only be used for bug-testing or quick examples.

As this is a free version of RStudio server there is a limit to what is supported and it shouldn't be used for computationally-intensive processes.

And, as mentioned: **No confidential data.**

2.5 Working with CHIs

Here are 4 functions for CHIs that could even be put in a small package. The Community Health Index (CHI) is a population register, which is used in Scotland for health care purposes. The CHI number uniquely identifies a person on the index.

2.5.1 chi_dob() - Extract date of birth from CHI

Note `cutoff_2000`. As CHI has only a two digit year, need to decide whether year is 1900s or 2000s. I don't think there is a formal way of determining this.

```
library(dplyr)
chi = c("1009701234", "1811431232", "1304496368")
# These CHIs are not real.
# The first is invalid, two and three are valid.

# Cut-off any thing before that number is considered 2000s
# i.e. at cutoff_2000 = 20, "18" is considered 2018, rather than 1918.
chi_dob = function(.data, cutoff_2000 = 20){
  .data %>%
    stringr::str_extract(".{6}") %>%
    lubridate::parse_date_time2("dmy", cutoff_2000 = cutoff_2000) %>%
    lubridate::as_date() # Make Date object, rather than POSIXct
}

chi_dob(chi)
```

```
## [1] "1970-09-10" "1943-11-18" "1949-04-13"
```

```
# From tibble
tibble(chi = chi) %>%
  mutate(
    dob = chi_dob(chi)
  )
```

```
## # A tibble: 3 x 2
##   chi        dob
##   <chr>      <date>
## 1 1009701234 1970-09-10
## 2 1811431232 1943-11-18
## 3 1304496368 1949-04-13
```

2.5.2 chi_gender() - Extract gender from CHI

Ninth digit is odd for men and even for women. A test for even is `x modulus 2 == 0`.

```
chi_gender = function(.data){
  .data %>%
```

```
    stringr::str_sub(9, 9) %>%
    as.numeric() %>%
    {ifelse(. %% 2 == 0, "Female", "Male")}
}

chi_gender(chi)
```

```
## [1] "Male"   "Male"   "Female"
```

```
# From tibble
tibble(chi = chi) %>%
  mutate(
    dob = chi_dob(chi),
    gender = chi_gender(chi)
  )
```

```
## # A tibble: 3 x 3
##   chi        dob        gender
##   <chr>      <date>     <chr>
## 1 1009701234 1970-09-10 Male
## 2 1811431232 1943-11-18 Male
## 3 1304496368 1949-04-13 Female
```

2.5.3 `chi_age()` - Extract age from CHI

Works for a single date or a vector of dates.

```
chi_age = function(.data, ref_date, cutoff_2000 = 20){
  dob = chi_dob(.data, cutoff_2000 = cutoff_2000)
  lubridate::interval(dob, ref_date) %>%
    as.numeric("years") %>%
    floor()
}

# Today
chi_age(chi, Sys.time())
```

```
## [1] 53 80 75
```

```
# Single date
library(lubridate)
chi_age(chi, dmy("11/09/2018"))
```

```
## [1] 48 74 69
```

```
# Vector
dates = dmy("11/09/2018",
            "09/05/2015",
            "10/03/2014")
chi_age(chi, dates)
```

```
## [1] 48 71 64
```

```
# From tibble
tibble(chi = chi) %>%
  mutate(
    dob = chi_dob(chi),
    gender = chi_gender(chi),
    age = chi_age(chi, Sys.time())
  )
```

```
## # A tibble: 3 x 4
##   chi        dob        gender   age
##   <chr>      <date>     <chr>  <dbl>
## 1 1009701234 1970-09-10 Male      53
## 2 1811431232 1943-11-18 Male      80
## 3 1304496368 1949-04-13 Female    75
```

2.5.4 `chi_valid()` - Logical test for valid CHI

The final digit of the CHI can be used to test that the number is correct via the modulus 11 algorithm.

```
chi_valid = function(.data){
  .data %>%
    stringr::str_split("", simplify = TRUE) %>%
    .[, -10] %>%              # Working with matrices hence brackets
    apply(1, as.numeric) %>%  # Convert from string
    {seq(10, 2) %*% .} %>%    # Multiply and sum step
    {. %% 11} %>%             # Modulus 11
    {11 - .} %>%              # Substract from 11
    dplyr::near(              # Compare result with 10th digit.
      {stringr::str_sub(chi, 10) %>% as.numeric()}
    ) %>%
    as.vector()
}

chi_valid(chi)
```

```
## [1] FALSE  TRUE  TRUE
```

```
# From tibble
tibble(chi = chi) %>%
  mutate(
    dob = chi_dob(chi),
    gender = chi_gender(chi),
    age = chi_age(chi, Sys.time()),
    chi_valid = chi_valid(chi)
  )
```

```
## # A tibble: 3 x 5
##   chi        dob        gender   age chi_valid
##   <chr>      <date>     <chr>  <dbl> <lgl>
## 1 1009701234 1970-09-10 Male      53 FALSE
## 2 1811431232 1943-11-18 Male      80 TRUE
## 3 1304496368 1949-04-13 Female    75 TRUE
```

2.6 Working with dates

2.6.1 Difference between two dates

I always forget how to do this neatly. I often want days as a numeric, not a lubridate type object.

```
library(lubridate)
date1 = dmy("12/03/2018", "14/05/2017")
date2 = dmy("11/09/2019", "11/04/2019")

interval(date1, date2) %>%
  as.numeric("days")
```

```
## [1] 548 697
```

2.6.2 Lags

This is useful for calculating, for instance, the period off medications. Lags are much better than long to wide solutions for this.

```
library(tidyverse)
library(lubridate)
id = c(2, 2, 2, 2, 3, 5)
```

```
medication = c("aspirin", "aspirin", "aspirin", "tylenol", "lipitor", "advil")
start.date = c("05/01/2017", "05/30/2017", "07/15/2017", "05/01/2017", "05/06/2017", "05/28/2017"
stop.date = c("05/04/2017", "06/10/2017", "07/27/2017", "05/15/2017", "05/12/2017", "06/13/2017")
df = tibble(id, medication, start.date, stop.date)
df
```

```
## # A tibble: 6 x 4
##      id medication start.date stop.date
##   <dbl> <chr>      <chr>      <chr>
## 1     2 aspirin    05/01/2017 05/04/2017
## 2     2 aspirin    05/30/2017 06/10/2017
## 3     2 aspirin    07/15/2017 07/27/2017
## 4     2 tylenol    05/01/2017 05/15/2017
## 5     3 lipitor    05/06/2017 05/12/2017
## 6     5 advil      05/28/2017 06/13/2017
```

```
df %>%
  mutate_at(c("start.date", "stop.date"), lubridate::mdy) %>% # make a date
  arrange(id, medication, start.date) %>%
  group_by(id, medication) %>%
  mutate(
    start_date_diff = start.date - lag(start.date),
    medication_period = stop.date-start.date
  )
```

```
## # A tibble: 6 x 6
## # Groups:   id, medication [4]
##      id medication start.date stop.date  start_date_diff medication_period
##   <dbl> <chr>      <date>     <date>     <drtn>          <drtn>
## 1     2 aspirin    2017-05-01 2017-05-04 NA days          3 days
## 2     2 aspirin    2017-05-30 2017-06-10 29 days         11 days
## 3     2 aspirin    2017-07-15 2017-07-27 46 days         12 days
## 4     2 tylenol    2017-05-01 2017-05-15 NA days         14 days
## 5     3 lipitor    2017-05-06 2017-05-12 NA days          6 days
## 6     5 advil      2017-05-28 2017-06-13 NA days         16 days
```

2.6.3 Pulling out "change in status" data

If you have a number of episodes per patient, each with a status and a time, then you need to do this as a starting point for CPH analysis.

2.6.4 Guess the Format of Dates

This function helps guess the date format from a dataset you have received. Add in new formats (as per lubridate) if they don't match. Once you work out what format it is then extract date with `mdy()` or `dmy()` etc. based on format.

```
library(tidyverse)
library(lubridate)
lakers = lakers %>%
  mutate(mm_date = paste0(str_sub(date, start =5, end = 6), "/",
                          str_sub(date, start = 7, end = 8), "/",
                          str_sub(date, start = 1, end = 4)),
         dd_date = paste0(str_sub(date, start =7, end = 8), "/",
                          str_sub(date, start = 5, end = 6), "/",
                          str_sub(date, start = 1, end = 4)))

guess_formats(lakers$mm_date, c("dmy", "mdy"), print_matches = TRUE)
guess_formats(lakers$dd_date, c("dmy", "mdy"), print_matches = TRUE)
```

2.6.4.1 Example data

```
library(dplyr)
library(lubridate)
library(finalfit)
mydata = tibble(
  id = c(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4,5,5,5,5),
  status = c(0,0,0,1,0,0,1,1,0,0,0,0,0,1,1,1,0,0,1,1),
  group = c(rep(0, 8), rep(1, 12)) %>% factor(),
  opdate = rep("2010/01/01", 20) %>% ymd(),
  status_date = c(
    "2010/02/01", "2010/03/01", "2010/04/01", "2010/05/01",
    "2010/02/02", "2010/03/02", "2010/04/02", "2010/05/02",
    "2010/02/03", "2010/03/03", "2010/04/03", "2010/05/03",
    "2010/02/04", "2010/03/04", "2010/04/04", "2010/05/04",
    "2010/02/05", "2010/03/05", "2010/04/05", "2010/05/05"
  ) %>% ymd()
)
mydata
```

```
## # A tibble: 20 x 5
##       id status group opdate     status_date
##    <dbl>  <dbl> <fct> <date>     <date>
##  1     1      0 0     2010-01-01 2010-02-01
```

```
##  2     1      0 0     2010-01-01 2010-03-01
##  3     1      0 0     2010-01-01 2010-04-01
##  4     1      1 0     2010-01-01 2010-05-01
##  5     2      0 0     2010-01-01 2010-02-02
##  6     2      0 0     2010-01-01 2010-03-02
##  7     2      1 0     2010-01-01 2010-04-02
##  8     2      1 0     2010-01-01 2010-05-02
##  9     3      0 1     2010-01-01 2010-02-03
## 10     3      0 1     2010-01-01 2010-03-03
## 11     3      0 1     2010-01-01 2010-04-03
## 12     3      0 1     2010-01-01 2010-05-03
## 13     4      0 1     2010-01-01 2010-02-04
## 14     4      1 1     2010-01-01 2010-03-04
## 15     4      1 1     2010-01-01 2010-04-04
## 16     4      1 1     2010-01-01 2010-05-04
## 17     5      0 1     2010-01-01 2010-02-05
## 18     5      0 1     2010-01-01 2010-03-05
## 19     5      1 1     2010-01-01 2010-04-05
## 20     5      1 1     2010-01-01 2010-05-05
```

2.6.4.2 Compute time from op date to current review

. . . if necessary

```
mydata = mydata %>%
  arrange(id, status_date) %>%
  mutate(
    time = interval(opdate, status_date) %>% as.numeric("days")
  )
mydata
```

```
## # A tibble: 20 x 6
##       id status group opdate     status_date  time
##    <dbl>  <dbl> <fct> <date>     <date>      <dbl>
##  1     1      0 0     2010-01-01 2010-02-01     31
##  2     1      0 0     2010-01-01 2010-03-01     59
##  3     1      0 0     2010-01-01 2010-04-01     90
##  4     1      1 0     2010-01-01 2010-05-01    120
##  5     2      0 0     2010-01-01 2010-02-02     32
##  6     2      0 0     2010-01-01 2010-03-02     60
##  7     2      1 0     2010-01-01 2010-04-02     91
##  8     2      1 0     2010-01-01 2010-05-02    121
##  9     3      0 1     2010-01-01 2010-02-03     33
## 10     3      0 1     2010-01-01 2010-03-03     61
```

```
## 11     3      0 1     2010-01-01 2010-04-03     92
## 12     3      0 1     2010-01-01 2010-05-03    122
## 13     4      0 1     2010-01-01 2010-02-04     34
## 14     4      1 1     2010-01-01 2010-03-04     62
## 15     4      1 1     2010-01-01 2010-04-04     93
## 16     4      1 1     2010-01-01 2010-05-04    123
## 17     5      0 1     2010-01-01 2010-02-05     35
## 18     5      0 1     2010-01-01 2010-03-05     63
## 19     5      1 1     2010-01-01 2010-04-05     94
## 20     5      1 1     2010-01-01 2010-05-05    124
```

2.6.4.3 Pull out "change of status"

```
mydata = mydata %>%
  group_by(id) %>%
  mutate(
    status_change = status - lag(status) == 1,                          # Mark TRUE if
    status_nochange = sum(status) == 0,                                 # Mark if no c
    status_nochange_keep = !duplicated(status_nochange, fromLast= TRUE) # Mark most re
  ) %>%
  filter(status_change | (status_nochange & status_nochange_keep)) %>%  # Filter out o
  select(-c(status_change, status_nochange, status_nochange_keep))      # Remove colum
mydata
```

```
## # A tibble: 5 x 6
## # Groups:   id [5]
##      id status group opdate     status_date  time
##   <dbl>  <dbl> <fct> <date>     <date>      <dbl>
## 1     1      1 0     2010-01-01 2010-05-01    120
## 2     2      1 0     2010-01-01 2010-04-02     91
## 3     3      0 1     2010-01-01 2010-05-03    122
## 4     4      1 1     2010-01-01 2010-03-04     62
## 5     5      1 1     2010-01-01 2010-04-05     94
```

2.6.4.4 Run CPH

```
mydata %>%
  finalfit("Surv(time, status)", "group")
```

```
##  Dependent: Surv(time, status)        all          HR (univariable)
##                           group 0 2 (40.0)                         -
```

```
##                                1 3 (60.0) 0.76 (0.10-5.51, p=0.786)
##         HR (multivariable)
##                          -
##  0.76 (0.10-5.51, p=0.786)
```

2.6.5 Isoweek and epiweek

This is frequently painful, when working with weeks in a year to aggregate data, how to best plot and how to deal with multiple years.

Easiest solution I have found (EMH) is just to work with the first date of each isoweek. Make a look up table and join.

```
# Generate random set of dates for demonstration purposes
## This is not required when using normally
library(tidyverse)
library(lubridate)
library(finalfit)
colon_s = colon_s %>%
  mutate(
    date = seq(ymd(20200101), ymd(20220601), by="day") %>%
      sample(929, replace = TRUE)
  )

# Make isoweek start date look-up table
## Make sure includes entire range of your dates of interest.
isoweek_lookup = tibble(date = seq(ymd(20200101), ymd(20220601), by = 1)) %>%
  mutate(isoweek = lubridate::isoweek(date),
         year = lubridate::year(date))

isoweek_lookup = isoweek_lookup %>%
  left_join(isoweek_lookup %>%
              group_by(isoweek, year) %>%
              slice(1) %>%
              select(isoweek_date = date, isoweek, year)
  )
```

```
## Joining with `by = join_by(isoweek, year)`
```

```
# Add isoweek_date (first date of a particular isoweek) and isoweek to analysis dataset.
colon_s %>%
  left_join(isoweek_lookup, by = c("date" = "date")) %>%
  select(date, isoweek, isoweek_date) %>%
  slice(1:6)
```

```
##         date isoweek isoweek_date
## 1 2021-10-11      41   2021-10-11
## 2 2022-02-11       6   2022-02-07
## 3 2021-08-19      33   2021-08-16
## 4 2020-11-28      48   2020-11-23
## 5 2021-03-07       9   2021-03-01
## 6 2020-02-24       9   2020-02-24
```

2.7 Labelling tables and figures in markdown chunks

2.7.1 Plots

I have made a plot and it looks nice (See Figure \ref{fig:a-nice-plot})

````
```{r fig.cap="A nice plot.\\label{fig:a-nice-plot}}
ggplot(mydata, aes(x = x)) +
  geom_histogram()
```
````

2.7.2 Tables

I have made a table and it looks nice (See Table \ref{tab:a-nice-table})

````
```{r}
library(knitr)
kable(mtcars[1:3,1:4], caption="A nice table.\\label{tab:a-nice-table}")
```
````

2.8 Join tables and overwrite values (that are often missing)

From here: https://stackoverflow.com/questions/35637135/left-join-two-data-frames-and-overwrite

```
library(dplyr)
.keys <- c("key1", "key2")

.base_table <- tribble(
    ~key1, ~key2, ~val1, ~val2,
```

```
    "A", "a", 0, 0,
    "A", "b", 0, 1,
    "B", "a", 1, 0,
    "B", "b", 1, 1)

.join_table <- tribble(
    ~key1, ~key2, ~val2,
    "A", "b", 100,
    "B", "a", 111)

# This works
df_result <- .base_table %>%
    # Pull off rows from base table that match the join table
    semi_join(.join_table, .keys) %>%
    # Drop cols from base table that are in join table, except for the key columns
    select(-matches(setdiff(names(.join_table), .keys))) %>%
    # Left join on the join table columns
    left_join(.join_table, .keys) %>%
    # Remove the matching rows from the base table, and
    # bind on the newly joined result from above.
    bind_rows(.base_table %>% anti_join(.join_table, .keys))

df_result %>%
    print()
```

```
## # A tibble: 4 x 4
##   key1  key2   val1  val2
##   <chr> <chr> <dbl> <dbl>
## 1 A     b         0   100
## 2 B     a         1   111
## 3 A     a         0     0
## 4 B     b         1     1
```

2.9 Read in multiple files

As of July 2021, library(readr) is happy to read in multiple files into the same tibble without you having to use `purrr::map_dfr()`.

```
library(tidyverse)

exampledata = read_csv(fs::dir_ls(path = "data_folder", glob = "*.csv"))
```

It's also happy to automatically uncompress files ending with .zip, .gz or similar.

2.10 Averaging multiple AUCs (say from imputed datasets)

Final method here probably makes most sense: ‘

```
auc = c(0.5, 0.6, 0.7, 0.8, 0.9, 0.99999)
mean(auc)                          # Mean
median(auc)                        # Median
prod(auc)^(1/length(auc))          # Geometric mean
exp(mean(log(auc)))                # Geometric mean again
log(auc)                           # Logs
log(auc / (1-auc) )                # Put values on scale from 0 (0.5) to Inf (1)
exp(mean(log(auc/(1-auc)))) /      # Mean on the above scale.
  (1 + exp(mean(log(auc/(1-auc)))))
```

```
auc = c(0.5, 0.6, 0.7, 0.8, 0.9, 0.99999)
mean(auc)                          # Mean
```

```
## [1] 0.7499983
```

```
median(auc)                        # Median
```

```
## [1] 0.75
```

```
prod(auc)^(1/length(auc))          # Geometric mean
```

```
## [1] 0.7298908
```

```
exp(mean(log(auc)))                # Geometric mean again
```

```
## [1] 0.7298908
```

```
log(auc)                           # Logs
```

```
## [1] -6.931472e-01 -5.108256e-01 -3.566749e-01 -2.231436e-01 -1.053605e-01
## [6] -1.000005e-05
```

```
log(auc / (1-auc) )                # Put values on scale from 0 (0.5) to Inf (1)
```

```
## [1]  0.0000000  0.4054651  0.8472979  1.3862944  2.1972246 11.5129155
```

```
exp(mean(log(auc/(1-auc)))) /   # Mean on the above scale.
  (1 + exp(mean(log(auc/(1-auc)))))
```

```
## [1] 0.9384781
```

2.11 Formating thousands automatically

Works in R Markdown and Quarto.

Before: There are 1.2345×10^4 patients in this dataset.

```
inline_hook = function(x) {
  if (is.numeric(x)) {
    format(x,big.mark = ",")
  } else x
}
knitr::knit_hooks$set(inline = inline_hook)
```

After: There are 12,345 patients in this dataset.

This only works on numbers entered by inline R.

2.12 DataTable function

```
mydt = function(df, pagelength = 20){
  DT::datatable(df,
                rownames = FALSE,
                extensions = c('Buttons'),
                filter = "top",
                options = list(
                  pageLength = pagelength,
                  autoWidth = TRUE,
                  scrollX = TRUE,
                  searchHighlight = TRUE))
}
```

Usage:

```
palmerpenguins::penguins %>%
  slice(1:5) %>%
  mydt()
```

Chapter 3

Data manipulation

3.1 Collapse multiple "no" and "yes" options

Common to have to do this in GlobalSurg projects:

```
library(dplyr)
mydata = tibble(
  ssi.factor = c("No", "Yes, no treatment/wound opened only (CD 1)",
                 "Yes, antibiotics only (CD 2)", "Yes, return to operating theatre (CD 3)",
                 "Yes, requiring critical care admission (CD 4)",
                 "Yes, resulting in death (CD 5)",
                 "Unknown") %>%
    factor(),
  mri.factor = c("No, not available", "No, not indicated",
                 "No, indicated and facilities available, but patient not able to pay",
                 "Yes", "Unknown", "Unknown", "Unknown") %>%
    factor()
)

# Two functions make this work
fct_collapse_yn = function(.f){
  .f %>%
    forcats::fct_relabel(~ gsub("^No.*", "No", .)) %>%
    forcats::fct_relabel(~ gsub("^Yes.*", "Yes", .))
}

is.yn = function(.data){
  .f = is.factor(.data)
  .yn = .data %>%
    levels() %>%
```

```
    grepl("(No.+)|(Yes.+)", .) %>%
    any()
  all(.f, .yn)
}

# Raw variable
mydata %>%
  count(ssi.factor)
```

```
## # A tibble: 7 x 2
##   ssi.factor                                          n
##   <fct>                                           <int>
## 1 No                                                  1
## 2 Unknown                                             1
## 3 Yes, antibiotics only (CD 2)                        1
## 4 Yes, no treatment/wound opened only (CD 1)          1
## 5 Yes, requiring critical care admission (CD 4)       1
## 6 Yes, resulting in death (CD 5)                      1
## 7 Yes, return to operating theatre (CD 3)             1
```

```
# Collapse to _yn version
mydata %>%
  mutate(across(where(is.yn), fct_collapse_yn, .names = "{col}_yn")) %>%
  count(ssi.factor_yn)
```

```
## # A tibble: 3 x 2
##   ssi.factor_yn     n
##   <fct>         <int>
## 1 No                1
## 2 Unknown           1
## 3 Yes               5
```

3.2 Filtering best practice

3.2.1 From Jamie Farrell

Particularly useful in OpenSAFELY, but should be adopted by us all.

This creates inclusion flags, and then summarises those inclusion flags in a table. This allows us to see exactly how many rows are removed at each filter step.

The inclusion flags are then used to run the final filter.

```
library(finalfit)
library(tidyverse)

# Utility function ----
## Ignore, just needed for the example.
fct_case_when <- function(...) {
  # uses dplyr::case_when but converts the output to a factor,
  # with factors ordered as they appear in the case_when's  ... argument
  args <- as.list(match.call())
  levels <- sapply(args[-1], function(f) f[[3]])  # extract RHS of formula
  levels <- levels[!is.na(levels)]
  factor(dplyr::case_when(...), levels=levels)
}

# Load data
colon_s = colon_s

# Create inclusion flags:
# Age over 50, male, obstruction data not missing
colon_s = colon_s %>%
  mutate(
    is_age_abover_50 = age > 50,
    is_male = sex.factor == "Male",
    obstruction_not_missing = !is.na(obstruct.factor)
  )

# Create exclusion table
flowchart = colon_s %>%
  transmute(
  c0 = TRUE,
  c1 = c0 & is_age_abover_50,
  c2 = c1 & is_male,
  c3 = c2 & obstruction_not_missing,
) %>%
  summarise(
    across(.fns=sum)
  ) %>%
  mutate(pivot_col = NA) %>%
  pivot_longer(
    cols=-pivot_col,
    names_to="criteria",
    values_to="n"
  ) %>%
  select(-pivot_col) %>%
  mutate(
```

```
    n_exclude = lag(n) - n,
    pct_all = (n/first(n)) %>% scales::percent(0.1),
    pct_exclude_step = (n_exclude/lag(n)) %>% scales::percent(0.1),
    crit = str_extract(criteria, "^c\\d+"),
    criteria = fct_case_when(
      crit == "c0" ~ "Original dataset",
      crit == "c1" ~ "  with age over 50 years",
      crit == "c2" ~ "  is male",
      crit == "c3" ~ "  obstruction data not missing",
      TRUE ~ NA_character_
    )
  ) %>%
  select(criteria, n, n_exclude, pct_all, pct_exclude_step)
flowchart
```

```
## # A tibble: 4 x 5
##   criteria                              n n_exclude pct_all pct_exclude_step
##   <fct>                             <int>     <int> <chr>   <chr>
## 1 "Original dataset"                  929        NA 100.0%  <NA>
## 2 "  with age over 50 years"          732       197 78.8%   21.2%
## 3 "  is male"                         385       347 41.4%   47.4%
## 4 "  obstruction data not missing"    375        10 40.4%   2.6%
```

Now filter the dataset in one step.

```
# Filtered dataset
colon_filtered = colon_s %>%
  filter(
    is_age_abover_50,
    is_male,
    obstruction_not_missing
  )
```

3.3 Filter NA: Dropping rows where all specified variables are NA

I want to keep rows that have a value for `important_a` and/or `important_b` (so rows 1, 3, 4). I don't care whether `whatever_c` is empty or not, but I do want to keep it.

```
library(tidyverse)
```

```
mydata  = tibble(important_a = c("Value", NA, "Value", NA, NA),
                 important_b = c(NA, NA, "Value", "Value", NA),
                 whatever_c  = c(NA, "Value", NA, NA, NA))

mydata
```

```
## # A tibble: 5 x 3
##   important_a important_b whatever_c
##   <chr>       <chr>       <chr>
## 1 Value       <NA>        <NA>
## 2 <NA>        <NA>        Value
## 3 Value       Value       <NA>
## 4 <NA>        Value       <NA>
## 5 <NA>        <NA>        <NA>
```

Functions for missing values that are very useful, but don't do what I want are:

(1) This keeps complete cases based on all columns:

```
mydata %>%
  drop_na()
```

```
## # A tibble: 0 x 3
## # i 3 variables: important_a <chr>, important_b <chr>, whatever_c <chr>
```

(Returns 0 as we don't have rows where all 3 columns have a value).

(2) This keeps complete cases based on specified columns:

```
mydata %>%
  drop_na(important_a, important_b)
```

```
## # A tibble: 1 x 3
##   important_a important_b whatever_c
##   <chr>       <chr>       <chr>
## 1 Value       Value       <NA>
```

This only keeps the row where both a and b have a value.

(3) This keeps rows that have a value in any column:

```
mydata %>%
  filter_all(any_vars(! is.na(.)))
```

```
## # A tibble: 4 x 3
##   important_a important_b whatever_c
##   <chr>       <chr>       <chr>
## 1 Value       <NA>        <NA>
## 2 <NA>        <NA>        Value
## 3 Value       Value       <NA>
## 4 <NA>        Value       <NA>
```

The third example is better achieved using the janitor package:

```
mydata %>%
  janitor::remove_empty()
```

```
## # A tibble: 4 x 3
##   important_a important_b whatever_c
##   <chr>       <chr>       <chr>
## 1 Value       <NA>        <NA>
## 2 <NA>        <NA>        Value
## 3 Value       Value       <NA>
## 4 <NA>        Value       <NA>
```

Now, (3) is pretty close, but still, I'm not interested in row 2 - where both a and b are empty but c has a value (which is why it's kept).

(4) Simple solution

A quick solution is to use `! is.na()` for each variable inside a `filter()`:

```
mydata %>%
  filter(! is.na(important_a) | ! is.na(important_b))
```

```
## # A tibble: 3 x 3
##   important_a important_b whatever_c
##   <chr>       <chr>       <chr>
## 1 Value       <NA>        <NA>
## 2 Value       Value       <NA>
## 3 <NA>        Value       <NA>
```

And this is definitely what I do when I only have a couple of these variables. But if you have tens, then the filtering logic becomes horrendously long and it's easy to miss one out/make a mistake.

(5) Powerful solution:

A scalable solution is to use `filter_at()` with `vars()` with a select helper (e.g., `starts with()`), and then the `any_vars(! is.na(.))` that was introduced in (3).

```
mydata %>%
  filter_at(vars(starts_with("important_")), any_vars(! is.na(.)))
```

```
## # A tibble: 3 x 3
##   important_a important_b whatever_c
##   <chr>       <chr>       <chr>
## 1 Value       <NA>        <NA>
## 2 Value       Value       <NA>
## 3 <NA>        Value       <NA>
```

3.4 Vectorising rowwise procedures

We frequently have to aggregate across columns. `dplyr::rowwise()` is the `group_by()` equivalent for rows. But this is painfully slow for large datasets.

The following works beautifully and allows tidyselect of variable names.

```
library(tidyverse)
mydata = tibble(a = c(1,2,NA),
       b = c(2,3,NA),
       c = c(3,4,NA),
       d = c(NA,NA,NA))

mydata %>%
  mutate(s = pmap_dbl(select(., a:d), pmax, na.rm=TRUE))
```

```
## # A tibble: 3 x 5
##       a     b     c d         s
##   <dbl> <dbl> <dbl> <lgl> <dbl>
## 1     1     2     3 NA        3
## 2     2     3     4 NA        4
## 3    NA    NA    NA NA       NA
```

3.5 Multiple imputation and IPW for missing data

Two approaches are commonly used for dealing with missing data.

Multiple imputation (MI) "fills in" missing values. Inverse probability weighting (IPW) "weights" observed values to reflect characteristics of missing data.

Missing data is discussed in detail here:https://finalfit.org/articles/missing.html

The IPW methods are taken from here: https://www.hsph.harvard.edu/miguel-hernan/causal-inference-book/

3.5.1 Data

```
library(finalfit)
library(tidyverse)
library(mice)

## Inspect dataset
# colon_s %>% ff_glimpse()

## Below we will use these variables to model,
## so remove missings from them and from outcome for this demonstration
explanatory = c("age",
                "sex.factor",
                "obstruct.factor",
                "perfor.factor",
                "adhere.factor",
                "nodes",
                "differ.factor",
                "rx")


## Remove existing missings
colon_s = colon_s %>%
  select(mort_5yr, explanatory) %>%
  drop_na()
```

Here is the truth for mortality at 5 years in these data.

```
## Truth
colon_s %>%
  summary_factorlist(explanatory = "mort_5yr")
```

```
##             label levels        all
##  Mortality 5 year  Alive 476 (55.7)
##                     Died 378 (44.3)
```

Let's conditionally delete some outcomes.

```
set.seed(1234)
colon_rs = colon_s %>%
  mutate(mort_5yr = ifelse(mort_5yr == "Alive",
                           sample(c("Alive", NA_character_),
                                  size = sum(colon_s$mort_5yr == "Alive"),
                                  replace = TRUE, prob = c(0.9, 0.1)),
                           sample(c("Died", NA_character_),
                                  size = sum(colon_s$mort_5yr == "Died"),
                                  replace = TRUE, prob = c(0.85, 0.15))) %>%
           factor()
  )
```

The proportion that died, is now lower due to our artifically introduced reporting bias:

```
colon_rs %>%
  summary_factorlist(explanatory = "mort_5yr")
```

```
##     label levels        all
##  mort_5yr  Alive 423 (57.2)
##             Died 316 (42.8)
```

3.5.2 Multiple imputation

First, let's use multiple imputation to "fill in" missing outcome using the existing information we have about the patient.

```
## Choose which variables in dataset to include in imputation.
predM = colon_rs %>%
  select(mort_5yr, explanatory) %>%
  missing_predictorMatrix(
    drop_from_imputed = c(""),
    drop_from_imputer = c(""))

# This is needed to drop from imputed
# Include for completeness, but not used here.
m0 = mice(colon_rs %>%
            select(mort_5yr, explanatory), maxit=0)
# m0$method[c("study_id")] = ""
# m0$method[c("redcap_data_access_group")] = ""

colon_rs_imputed = colon_rs %>%
```

```
  select(mort_5yr, explanatory) %>%
  mice(m = 10, maxit = 10, predictorMatrix = predM, method = m0$method,
       print = FALSE) # print false just for example

## Check output here:
# summary(colon_rs_imputed)
# plot(colon_rs_imputed )
```

3.5.2.1 Reduce imputed sets (mean)

As can be seen, the imputed proportion approaches that of the "true" value.

```
colon_rs_imputed %>%
  complete("all") %>%
  map(~ (.) %>% count(mort_5yr) %>%
        mutate(nn = sum(n),
               prop = n/nn) %>%
        select(-mort_5yr)) %>%
  reduce(`+`) / 10
```

```
##       n  nn      prop
## 1 484.8 854 0.5676815
## 2 369.2 854 0.4323185
```

3.5.3 IPW

Rather than filling in the missing values, we can weight the observed values to reflect the characteristics of the missing data.

```
## Define observed vs non-observed groups
colon_rs = colon_rs %>%
  mutate(
    group_fu = if_else(is.na(mort_5yr), 0, 1)
  )

colon_rs %>%
  count(group_fu)
```

```
##   group_fu   n
## 1        0 115
## 2        1 739
```

This method uses "stabilised weights", meaning weights should have a mean of 1.

```
# Numerator model
fit_num = colon_rs %>%
  glmmulti("group_fu", 1)

# Denominator model
fit_dem = colon_rs %>%
  glmmulti("group_fu", explanatory)

# Check denominator model
fit_dem %>%
  fit2df()
```

```
## Waiting for profiling to be done...
```

```
##              explanatory                          OR
## 1                    age  1.00 (0.99-1.02, p=0.766)
## 2         sex.factorMale  0.93 (0.62-1.38, p=0.710)
## 3     obstruct.factorYes  1.07 (0.65-1.83, p=0.803)
## 4       perfor.factorYes 4.71 (0.97-85.05, p=0.133)
## 5       adhere.factorYes  0.72 (0.43-1.26, p=0.235)
## 6                  nodes  0.96 (0.91-1.01, p=0.083)
## 7  differ.factorModerate  1.61 (0.85-2.91, p=0.128)
## 8      differ.factorPoor  1.49 (0.69-3.15, p=0.301)
## 9                  rxLev  1.01 (0.63-1.63, p=0.961)
## 10            rxLev+5FU  1.11 (0.68-1.82, p=0.687)
```

```
colon_rs = colon_rs %>%
  mutate(
    p = predict(fit_num, type = "response"),
    pi = predict(fit_dem, type = "response"),
    weights = case_when(
      group_fu == 1 ~ p / pi,
      group_fu == 0 ~ (1-p) / (1-pi)
    )
  )

mean(colon_rs$weights)
```

```
## [1] 0.9987887
```

```
# Whoop whoop if ~1.0
```

A weighted proportion can then be generated, which gives a very similar result to the multiple imputation.

```
colon_rs %>%
  mutate(
    mort_5yr_wgt = (as.numeric(mort_5yr) - 1) * weights
  ) %>%
  summary_factorlist(explanatory = c(" mort_5yr", " mort_5yr_wgt"), digits = c(4, 4, 3
```

```
##         label    levels             all
##      mort_5yr     Alive    423.0 (57.2)
##                    Died    316.0 (42.8)
##  mort_5yr_wgt Mean (SD) 0.4317 (0.5008)
```

Chapter 4

Machine learning

4.1 Deep learning

4.1.1 Pulling images from REDCap directly to argodeep

4.1.1.1 Original file names

```
library(REDCapR)
uri = "https://redcap.cir.ed.ac.uk/api/"
token = "" # API token here
record_list = 1:318
field_list = c("photo", "photo_2", "photo_3", "photo_4")
event_list = c("wound_concerns_arm_2", "questionnaire_1_arm_2",
               "questionnaire_2_arm_2", "questionnaire_3_arm_2")
directory = "wound_raw" # destination directory must exist already


for(record in record_list){
  for(field in field_list){
    for(event in event_list){
      result =
        tryCatch({      # suppress breaking error when no image in slot
          redcap_download_file_oneshot(
            record       = record,
            field        = field,
            redcap_uri   = uri,
            token        = token,
            event        = event,
```

```
            overwrite     = TRUE,
            directory     = directory
          )
        }, error=function(e){})
    }
  }
}
```

4.1.1.2 Named from REDCap record ID and event

```
library(REDCapR)
uri = "https://redcap.cir.ed.ac.uk/api/"
token = "" # API token here
record_list = 1:318
field_list = c("photo", "photo_2", "photo_3", "photo_4")
event_list = c("wound_concerns_arm_2", "questionnaire_1_arm_2",
               "questionnaire_2_arm_2", "questionnaire_3_arm_2")
directory = "wound_named" # destination directory must exist already

for(record in record_list){
  for(field in field_list){
    for(event in event_list){
      file_name = paste0(record, "_", field, "_", event, ".jpg")
      result =
        tryCatch({
          redcap_download_file_oneshot(
            record        = record,
            field         = field,
            redcap_uri    = uri,
            token         = token,
            event         = event,
            overwrite     = TRUE,
            directory     = directory,
            file_name     = file_name
          )
        }, error=function(e){})
    }
  }
}
```

Chapter 5

Data Transfer and Eddie

Often it is necessary to transfer data into and out of RStudio server. This can be done from personal laptops (as long as you have permission for the data!), university supported desktops, Eddie and a number of other devices or servers.

5.1 Uploading and Downloading Data the Easy Way

Often the GUI in RStudio is sufficient. In the `Files` pane click upload to move data from your current computer into RStudio server. To download select the file and then click `More > Export`. As always this should be done only when appropriate.

Sometimes this isn't possible either with large files or files stored remotely on other servers such as Eddie.

5.2 Alternative Methods when the Easy Way won't work

5.2.1 What is Eddie?

Eddie Mark 3 is the third iteration of the University's compute cluster and is available to all University of Edinburgh researchers. It consists of some 7000 Intel® Xeon® cores with up to 3 TB of memory available per compute node.

The surgical informatics group has a shared folder in the Eddie cluster which can be accessed to store data and perform analyses which might be too large

or complex for RStudio server. Every task in Eddie is controlled through the command line interface - those familiar with Linux/Unix will be familiar with this.

5.3 Using the Command Line

Sometimes the command line is the only possible way to copy data to and from RStudio server. Argonaut and Argosafe are `SSH`-enabled (Secure SHell) meaning that you can securely copy data to and from them using the command line editor on another device. It is also possible to use the RStudio terminal to copy to another device or server that is `SSH`-enabled although this isn't currently recommended due to issues with RStudio server's websockets (when you type in the terminal you have to do so very slowly for characters to appear in the right order or you have to use a script - described below).

First, if you don't have a command line editor or SSH client installed (often the case for earlier Windows versions although there is talk of a native client becoming default) then you will need to install one. For working on a Windows device generally PuTTY is the recommended SSH client (allows you connect to other servers) and a reasonable command line editor to work with files on the local device is GitBash.

5.3.1 Downloads and Setup - Eddie Example

1. Make sure you have the University VPN downloaded for your own computer to access Eddie if needed. The link is at: https://www.ed.ac.uk/information-services/computing/desktop-personal/vpn/vpn-service-using. It is the Cisco Connect Any Client which logs into the VPN (the password should be different to your EASE password).

2. Download the PuTTY terminal software: https://www.putty.org/.

Open up PuTTY and you will see a configuration screen. On this screen make sure to enter `eddie.ecdf.ed.ac.uk` into the box and tick SSH as your method for connection. The PuTTY terminal will launch (assuming you are connected to the University VPN already) and ask `login as` at which point you should enter your EASE username. You will then be asked for your EASE password and you should now see that you are logged into Eddie.

If you are logging in from RStudio Server or another terminal software you should enter on the command line:

```
ssh <UUN>@eddie.ecdf.ed.ac.uk
```

is your university username for EASE. You will then be asked for your password.

More information on Eddie basics can also be found at: https://www.wiki.ed.ac.uk/display/ResearchServices/Quickstart.

5.3.2 Copy and Paste with PuTTY

Like many other command line interfaces PuTTY can be made to work more easily with copy and paste. This is done simply through highlighting and clicking and not with traditional `ctrl-c` and `ctrl-v` commands like typical word processors.

To copy text from PuTTY: Highlight the text. That's it! No need to click anything or type / press anything, highlighting is enough. You can then paste the text elsewhere.

To copy text into PuTTY: Once you've copied text (either by highlighting in PuTTY itself or by using `ctrl-c` in another programme, just right-click. The text will appear at the command line. If you copy several lines separated by `\newline` then PuTTY will run each line up until the last one copied and leave the last line at the command line (if you highlight large sections of text in PuTTY and right-click it will try to run all of them).

5.3.3 Closing a Session in PuTTY

To close a session use `ctrl-d`.

5.3.4 Eddie File Structure

Once you have logged into Eddie there are several directories (folders/places) where you can store and manipulate files. Moving between these directories is usually done using the `cd` command. The same idea is applicable to your own machine

When you first log in you will default to your home directory. In order to see the "path" to that directory enter the command:

```
pwd
```

The terminal should print out something like:

```
/home/<UUN>
```

To get back to this directory at any point enter one of the following (they are both equivalent):

```
cd /home/<UUN>
```

or

```
cd ~
```

When inside your home folder to see any of the files or subdirectories in your home directory enter:

```
ls -a
```

The `-a` argument to ls shows hidden files which begin with a `.` such as `.Renviron` if you have created this. There are several other arguments which can be passed to ls such as `-l` which will should the permissions for the file or subdirectory. When using the `-l` argument the files start with `-` and are followed by 7 characters or `-` which explain whether different groups of people can write, read or execute the files. The first three are for the file owner, the next three for the group and the next three for any else with access to the directory. For example the following printed after running `ls -l` would indicate that the owner could read, write and execute a file, the group could read and execute it and that others could only read it:

```
-rwxr-xr-- ... ... ... ... my_file.txt
```

5.4 Copying Data from Eddie into RStudio server

5.4.1 Method 1: Using PuTTY (or another terminal/shell connected to Eddie)

To copy data from another server or device into RStudio Server use the following code in the command line editor (e.g. PuTTY, GitBash etc.):

```
scp <file_path_on_other_device_or_server>new_file.txt <RStudio_Server_Username>@argonau
```

You will be asked for your RStudio server (Argonaut etc.) password. If you are unsure of the file path enter `pwd` when you are working in the directory with the file before copying and pasting.

If you are not sure of the path to the directory in which you wish to copy the data to or from then use `getwd()` in RStudio when inside the project you wish

to copy to (you many need to add a subdirectory folder to the `getwd()` output if you are using subdirectories in your RStudio projects).

The `scp` command will work with Argonaut and Argosafe as the ability to allow SSH connections has been activated. This may need to be established separately for other RStudio servers within the department.

5.4.2 Method 2: Using the RStudio Server Terminal

To use the RStudio server terminal to copy data in and out of Eddie do not SSH into Eddie as described above but instead use the `scp` command. If you are using RStudio's terminal for accessing Eddie instead of PuTTY or something equivalent then you will need to either temporarily disconnect from Eddie (`ctrl-d`) or open a new terminal if using RStudio server Pro which allows multiple terminals.

In the terminal enter the following command to copy a single file from Eddie into RStudio server from a project subdirectory in the SurgInfGrp directory:

```
scp <UUN>@eddie.ecdf.ed.ac.uk:/exports/cmvm/eddie/edmed/groups/SurgInfGrp/<project_subdirectory>/

# Or adapt to copy entire directory contents (drop the * to copy the directory/folder as well as
scp -r <UUN>@eddie.ecdf.ed.ac.uk:/exports/cmvm/eddie/edmed/groups/SurgInfGrp/<project_subdirector
```

On entering the command to the terminal you will be asked to enter your EASE password (type this slowly if using RStudio Server Pro prior to web sockets issue being fixed). In order to move data in the other direction simply change the order of the file paths.

5.5 Group Folders on Eddie and DataStore

Use the `quota` command to check your disk space, this will include the shared space, e.g., :

```
[rots@login03(eddie) ~]$ quota

---------------------------------------------------------------
                  DISK QUOTA AND USAGE
/home/rots:
     84.62 MB of 10240.00 MB (.82%) used

/exports/eddie/scratch/rots:
     0 GB of 2048.00 GB (0%) used
```

```
/exports/cmvm/eddie/edmed/groups/SurgInfGrp:
     918.89 GB of 3072.00 GB (29.91%) used

----------------------------------------------------------------
```

To get to the Surgical Informatics shared group directory enter:

```
cd /exports/cmvm/eddie/edmed/groups/SurgInfGrp
```

If you don't have access then Riinu or Ewen can provide this as they have admin rights.

This directory should be the place to store any group projects or apps or other files which are somewhat (but not very) large.

The group space should be used to store bulk files which can be staged into Eddie for an active session but will not be permanently stored in Eddie. Consider storing your files on DataStore or other services for longer term needs. DataStore can be checked from the quota.txt file that's in the main folder.

5.5.1 The Scratch Space

Your own personal scratch space is where you should work on active projects during a session after staging them in from the datastore and/or shared Surgical Informatics group directory. The scratch space has 2TB of storage per user but this is cleaned up after one month. This means large datasets can be analysed here but not stored in the long-term. To find your own personal scratch space enter:

```
cd /exports/eddie/scratch/<UUN>
```

Finally there is also a temporary directory ($TMPDIR) which is only present and accessible whilst a job is running in Eddie. This has 1TB of available storage.

5.5.2 DataStore

Our shared DataStore folder is at:

```
/exports/cmvm/datastore/edmed/groups/SurgInfGrp
```

It should be mounted in argosafe at /home/common.

5.5.3 Running Eddie from RStudio Server

It is possible to log in to Eddie from RStudio server and use Shell Scripts stored in RStudio to perform tasks in Eddie. This may be helpful if you plan to modify shell scripts a lot and want to have the benefit of the RStudio interface. `ctrl-alt-enter` sends data to the terminal in the same way that it would the console without the `alt`.

To connect initially enter:

```
ssh <UUN>@eddie.ecdf.ed.ac.uk
```

You will then be asked for your password which has to be typed (currently slowly and carefully!) into the terminal. Afterwards you can send any commands from a script using `ctrl-alt-enter`. This will not work with `Rmd` or `notebooks`.

5.5.4 Modules (Applications) in Eddie

Eddie has several modules (applications) which can be run such as R, python, cuda, java, intel, fastqc etc. etc. The best way to see which modules you have available is to run `module avail`. To see which modules are loaded is to run `module list`.

To load in a new module to use run the following:

```
module load <module>

# For example:
module load R/3.5.3

# Note that the default R version is 3.3.2 (as of 13th June 2019) and is loaded using:
module load R
```

Once you have loaded modules that you will need for your analysis you may need to create new files or establish a library of packages for those modules. There are a default set of packages available for R when loaded from either the main applications library or from other installations on the IGMM paths but these are read-only meaning that for any customisation and installation of new packages, a separate library must be maintained and that the R options must be amended to point to this library. This can be done by creating a personal `.Rprofile` file in your home directory in Eddie which points to the Surgical Informatics Group Rlibrary.

It is not advised to create a separate library in your own Eddie space as this will quickly use up your disc quota. Theoretically a separate installation of a package could be stored in your own space if working on developer edition or github branch / fork of a package.

5.5.5 Working with R in Eddie

5.5.5.1 .Rprofile File

When you load up R an `.Rprofile` file is sourced and anything contained within the file will be run for the current R session. This is particularly useful in Eddie because the defaults are not very helpful:

- The R package library is the Eddie default library which cannot be added to
- The CRAN mirror is not specified so every session will ask you to select a new CRAN mirror to install packages in a temporary library

The `.Rprofile` file also has other options which can be customised to improve how your current R session will run on Eddie. There are a number of possible customisations but be careful as not all are helpful if you end up collaborating with other research teams, some of these customisations such as `stringsAsFactors=FALSE` by default could be used in a project-specific `.Rprofile` file with caution but if you are working with another research group who rely on `read.csv` and have scripts established which assume that all strings are factors then having that customisation in your home directory may generate bugs when collaborating on projects.

R will automatically look for a `.Rprofile` file when R is started during each Eddie session and will look for this file in three places with an order of preference. The first place is the current working directory of a project so an `.Rprofile` file could be stored here to generate very specific customisations for a project if necessary. The next place it will look is in your own home directory (`/home/<UUN>/` or `/~/`), this is where you should create a `.Rprofile` file which will be your default for all sessions, it is recommended to use this sparingly but that certain key features are used such as setting up your main library location as the Surgical Informatics Group and setting up a default CRAN mirror e.g. the RStudio CRAN mirror. Finally, if there is not a `.Rprofile` file here then R will attempt to source a file in the R home directory (found in R using `.R.home()`) and if there is no file here then no customisation will occur. Some servers also have a `.Rprofile.site` file although this is not currently present in Eddie.

Important: Always leave the final line of an `.Rprofile` file blank.

A possible `.Rprofile` file for using in Eddie as part of the Surgical Informatics Group is:

```
## Rprofile template


## Stop being asked for CRAN mirror every time
options("repos" = c(CRAN = "http://cran.rstudio.com/"))
```

```
## Change the default editor to nano
options(editor="nano")

## Change the terminal prompt to make it clear R is loaded
options(prompt="R > ")

## Prevent default saving of workspace image (similar to recommended RStudio server settings)
q <- function(save="no", ...) {
        quit(save=save, ...)
}


## Clever code to allow tab-completion of package names used in library()
utils::rc.settings(ipck=TRUE)


## Add some colour to the console if colourout is available
if(Sys.Getenv("TERM") == "xterm-256color")
        library("colorout")

## Create a new envisible environment which can be used to create new functions
## Benefit of this is that all new functions here are hidden in environment and not rmeoved by rm
.env <- new.env()


## Set library path to make sure packages are loaded from SurgInfGrp
.libPaths("/exports/cmvm/eddie/edmed/groups/SurgInfGrp/R/Rlibrary")


## If working on a particular project e.g. the gwas_pipeline from IGMM best to create a new libra
## Just copy the .Rprofile file from your own home directory into the project working directory a


## Attach all the variables created
attach(.env)


## Remember that an .Rprofile file always silently ignores the last line so don't forget to leave
```

The above configuration should generate minimal portability issues when working on other projects.

There are a few explanations of the benefits and side effects of having a `.Rprofile` file as well as ways to temporarily mask them for specific

projects which will be shared with other collaborators in this blog post: https://www.r-bloggers.com/fun-with-Rprofile-and-customizing-r-startup/.

Should you wish to have an entire directory of startup files (e.g. one file for CRAN, one file for library, one file for custom function etc. etc.) so that segments of the customisation can be shared quickly without modifying / dropping all of the other elements of the `.Rprofile` file then this is described here: https://cran.r-project.org/web/packages/startup/vignettes/startup-intro.html. This relies on the `startup` package. Other options such as having secure directories with GitHub tokens and other content which is protected from access by other Eddie users is also discussed there.

5.5.5.2 Creating an .Rprofile File

When you first use Eddie there will not be any `.Rprofile` file generated by default and a blank file will need to be created. This can be done by entering the following into the terminal after loading R. To load R enter `module load R/3.5.3` (or the currently available R versions seen on `module avail`) into the terminal followed by `R`. This will start an active R session. Then enter the following code (please do not change the path to the shared group folder! If you need to change the path to a project be sure to include the subdirectory otherwise R will use your `.Rprofile` file for all SurgInfGrp users which won't be popular):

```
file.create("~/.Rprofile")
```

This will create a blank `.Rprofile` file in your home directory on Eddie which should be edited to customise the R configuration.

In order to edit the file it is suggested that you use the nano Unix-based editor. Firstly quit the current R session as that has not yet been configured to use the editor. Enter `q()` as you would normally do when closing R and enter `n` if asked about saving workspace image.

Navigagte to your home directory and using `ls -a` you should see the `.Rprofile` file which is currently blank.

5.5.5.3 .Renviron File

In addition to changing the `.Rprofile` file you will most likely need to change the `.Renviron` file. The default for this is probably located in the R home directory which may be `/exports/applications/apps/SL7/R/3.5.3/lib64/R/etc/`. The `.Renviron` file is searched for in the same order by R as the `.Rprofile` file with R first aiming to find the file in the project directory and if not able to find it there looking in the user home directory and finally looking to the

R home directory. To copy the `Renviron` file in the R home directory into a `.Renviron` file in your home directory enter the following code in the terminal:

```
# Path will need edited if future installations of R replace 3.5.3
cp /exports/applications/apps/SL7/R/3.5.3/lib64/R/etc/Renviron ~/.Renviron
```

The following is a reasonable starting point for the `.Renviron` file in your own profile:

```
### etc/Renviron.  Generated from Renviron.in by configure.
###
### ${R_HOME}/etc/Renviron
###
### Record R system environment variables.

R_PLATFORM=${R_PLATFORM-'x86_64-pc-linux-gnu'}
## Default printer paper size: first record if user set R_PAPERSIZE
R_PAPERSIZE_USER=${R_PAPERSIZE}
R_PAPERSIZE=${R_PAPERSIZE-'a4'}
## Default print command
R_PRINTCMD=${R_PRINTCMD-''}
# for Rd2pdf, reference manual
R_RD4PDF=${R_RD4PDF-'times,hyper'}
## used for options("texi2dvi")
R_TEXI2DVICMD=${R_TEXI2DVICMD-${TEXI2DVI-'texi2dvi'}}
## used by untar and installing grDevices
R_GZIPCMD=${R_GZIPCMD-'/usr/bin/gzip'}
## Default zip/unzip commands
R_UNZIPCMD=${R_UNZIPCMD-'/usr/bin/unzip'}
R_ZIPCMD=${R_ZIPCMD-'/usr/bin/zip'}
R_BZIPCMD=${R_BZIPCMD-'/usr/bin/bzip2'}
## Default browser
R_BROWSER=${R_BROWSER-'/usr/bin/xdg-open'}
## Default pager
PAGER=${PAGER-'/usr/bin/less'}
## Default PDF viewer
R_PDFVIEWER=${R_PDFVIEWER-'/usr/bin/xdg-open'}
## Used by libtool
LN_S='ln -s'
MAKE=${MAKE-'make'}
## Prefer a POSIX-compliant sed on e.g. Solaris
SED=${SED-'/usr/bin/sed'}
## Prefer a tar that can automagically read compressed archives
TAR=${TAR-'/usr/bin/gtar'}
```

```
## System and compiler types.
R_SYSTEM_ABI='linux,gcc,gxx,gfortran,?'


## Change the default R libraries and directories
R_DOC_DIR=/exports/applications/apps/SL7/R/3.5.3/lib64/R/doc
R_INCLUDE_DIR=/exports/applications/apps/SL7/R/3.5.3/lib64/R/include
R_SHARE_DIR=/exports/applications/apps/SL7/R/3.5.3/lib64/R/share
RBIN=/exports/applications/apps/SL7/R/3.5.3/bin
RDIR=/exports/applications/apps/SL7/R/3.5.3/3.3.3
R_LIBS=/exports/cmvm/eddie/edmed/groups/SurgInfGrp/R/Rlibrary
R_LIBS_SITE=/exports/cmvm/eddie/edmed/groups/SurgInfGrp/R/Rlibrary
R_LIBS_USER=/exports/cmvm/eddie/edmed/groups/SurgInfGrp/R/Rlibrary




### Local Variables: ***
### mode: sh ***
### sh-indentation: 2 ***
```

The `.Renviron` file can be copied to specific projects when specific R files / directories (or files / directories for other languages) are needed to run a particular analysis. For example, to use the GCC compiler from IGMM the following could be added to the file:

```
## Change the Clang variables
CC=igmm/compilers/gcc/5.5.0 -fopenmp
CXX=igmm/compilers/gcc/5.5.0 -fopenmp
```

The `.Rprofile` files and `.Renviron` files should be as generic as possible in your own profile so as to avoid issues when sharing scripts. They can be tailored for specific projects when needed in the local working directory. Typically this should be to alter the paths to directories etc. and not to create bespoke functions which should be done in scripts.

5.5.6 Specific R Package Versions

When working with other teams it is often necessary to work with particular R package versions. One effective way to ensure you do this is using the remotes package:

```
# Install the GenABEL package from CRAN but as no longer supporte need the 1.8-0 version from the
remotes::install_version("GenABEL", version = "1.8-0")
```

5.5.7 General Eddie Etiquette

Don't overwrite or change other users files without permission and set up your own files to prevent others doing that if appropriate. Please don't leave configuration files like `.Renviron` files in the shared space as this can play havoc with other users R tasks in Eddie - if you need a particular configuration (e.g. a default package library for a project) then best to leave the config files in the directory for that project (or in your own home directory).

5.6 Additional Resources

The following resource covers some of the commands needed for the Univa Grid Enginge scheduler which is used to submit jobs on Eddie:

https://angus.readthedocs.io/en/2019/Intro_to_Cluster.html#

5.7 Using RStudio directly in Eddie

Eddie is not meant for GUIs, but the good people at IGMM have installed RStudio installed on Eddie. It is terribly slow, but better than nothing and can be useful when you need to have a quick look and are really tired of doing command line R.

Make sure you have X11 forwarding installed/enabled. On a Windows computer use MobaXterm. In macOS, install XQuarts, but then use the Terminal as usual.

```
ssh -Y username@eddie.ecdf.ed.ac.uk
qlogin -l h_vmem=8G
module load igmm/apps/RStudio/1.1.383
module load igmm/apps/R/3.6.3
rstudio
```

Of the qlogin line requesting an interactive session times out then try `qlogin -l h_vmem=4G` instead. 2GB is the absolute minimum that will work.

Use `module avail` to check what the latest installed RStudio and R versions are.

5.8 Configuring Python Deep Learning Environment in Eddie

In this section, we will show how to configure the deep learning development environment in Eddie with Anaconda, PyTorch and Python.

5.8.1 Package installation

Anaconda is a package management tool that can easily help us manage different python versions and package reliance. To use anaconda, simply try `module load anaconda`.

Then we need to redirect the package and environment installation folder to the group folder to prevent these folders from becoming too huge and occupying your personal storage:

```
conda config --add envs_dirs /exports/cmvm/eddie/edmed/groups/SurgInfGrp/anaconda/envs
conda config --add pkgs_dirs /exports/cmvm/eddie/edmed/groups/SurgInfGrp/anaconda/pkgs
```

We already have a configured pytorch environment in these folders. To use that, try `conda activate torch`.

If you want to build your own env from scratch, use `conda create -n mypython python=3.7.4`, where 3.7.4 can be any python version you want to use and mypython is the env name. Then you can activate your environment by using `conda activate mypython` and installing packages with `pip install` or `conda install`.

5.8.2 GPU usage

To use GPU computation, please refer to the https://www.wiki.ed.ac.uk/display/ResearchServices/GPUs for the detailed documents. Generally I recommend using an interactive session during model development phase and using a normal job session for long-term training.

A sample code to request an interactive session with a Tesla K80 GPU and 16G memory is `qlogin -pe gpu 1 -l h_vmem=16g`. You also need to `module load cuda` to enable cuda acceleration.

5.9 Configuring Jupyter Notebook Access

Jupyter notebook can be a very efficient tool to develop python analytics pipeline. You can run a notebook server on the remote Eddie node and access

the notebook from your local browser or editor. First, request an interactive session and activate the conda environment you need. Then use the following code to activate a jupyter notebook server:

```
#!/usr/bin/bash
## get tunneling info
ipnport=8515
ipnip=$(hostname -i)
## print tunneling instructions to jupyter-log-{jobid}.txt
echo -e "
        Copy/ Paste this in your local terminal to ssh tunnel with remote
        -----------------------------------------------------------------
        ##ssh -N -L $ipnport:$ipnip:$ipnport [USERNAME]@eddie.ecdf.ed.ac.uk
    ssh -N -L $ipnport:$ipnip:$ipnport
        -----------------------------------------------------------------

        Then open a browser on your local machine to the following address
        -----------------------------------------------------------------
        localhost:$ipnport (prefix w/ https:// if using password)
        -----------------------------------------------------------------
        "

## launch jupyter server
jupyter-notebook --no-browser --port=$ipnport --ip=$ipnip
```

If you want to have the access to the node after you close your terminal, you can use `screen`.

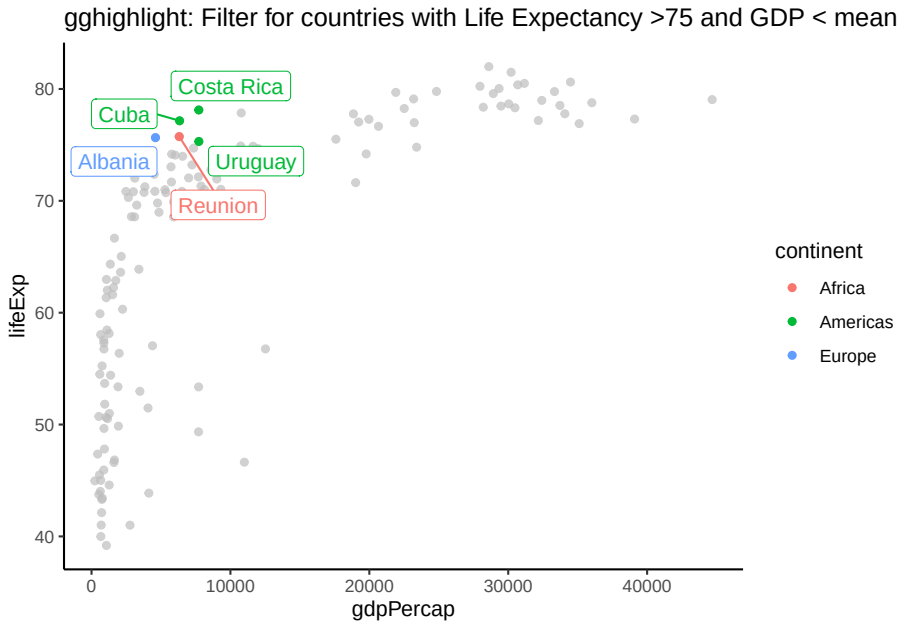
Chapter 6

Plotting

6.1 GGHighlight Example

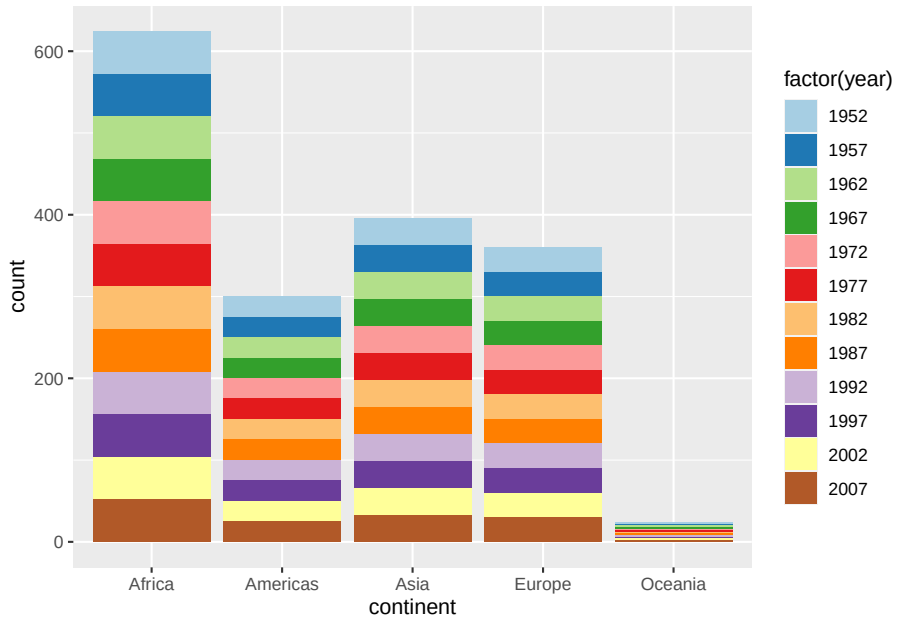
Plotting with gghighlight is pretty awesome allowing you to filter on any variable. It seems that gghighlight overwrites any 'colour' variable you put in the main aes. To get round this and have labels, save as a plot and add geom_label_repel separately.

```
library(gghighlight)
library(ggrepel)

mydata=gapminder

plot = mydata %>%
  filter(year == "2002") %>%
  ggplot(aes(x = gdpPercap, y = lifeExp, colour=continent)) +
  geom_point()+
  gghighlight(lifeExp > 75 & gdpPercap < mean(gdpPercap), label_key = country, use_direct_label =
  theme_classic()+
  labs(title= "gghighlight: Filter for countries with Life Expectancy >75 and GDP < mean" )

plot + geom_label_repel(aes(label= country), show.legend = FALSE) #only needed if you use  use_di
```

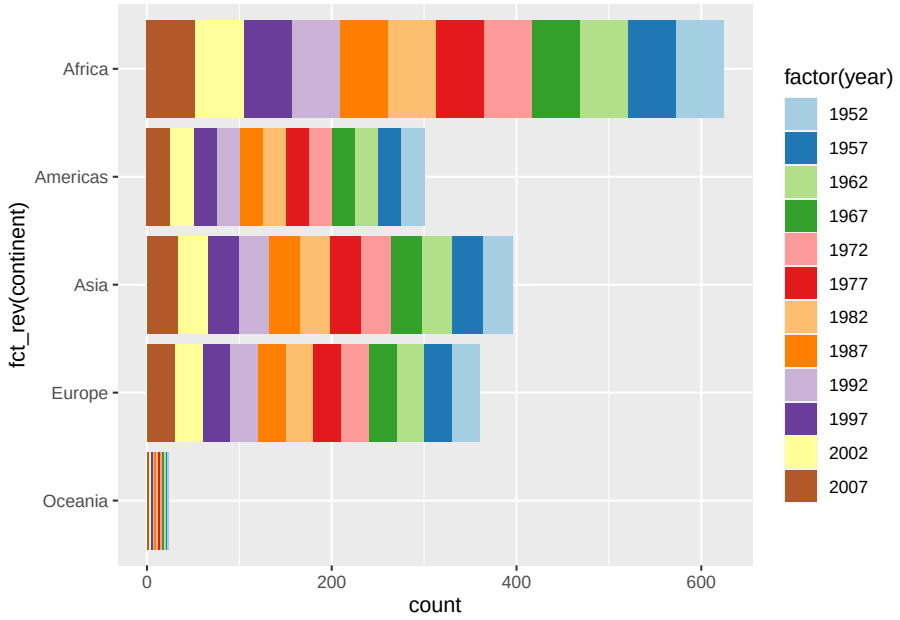


6.2 `coord_flip()` factor orders

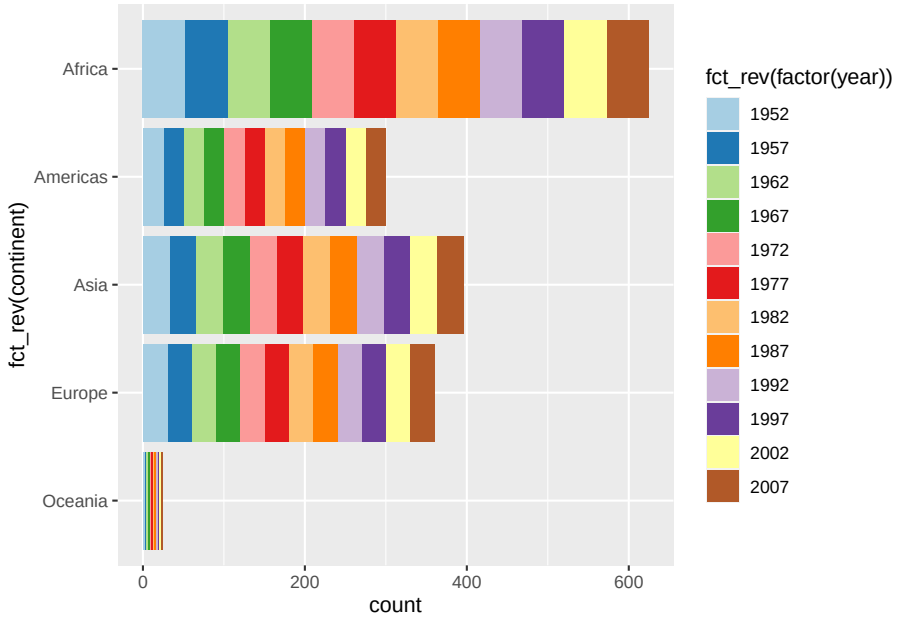
```
library(ggplot2)

# unflipped (yes this plot has no purpose :)
gapminder %>%
  ggplot(aes(x = continent, fill = factor(year))) +
  geom_bar() +
  scale_fill_brewer(palette = "Paired")
```




```
# flipped
gapminder %>%
  ggplot(aes(x = fct_rev(continent), fill = factor(year))) +
  geom_bar() +
  coord_flip() +
  scale_fill_brewer(palette = "Paired", breaks = rev) +
  guides(fill = guide_legend(reverse = TRUE))
```

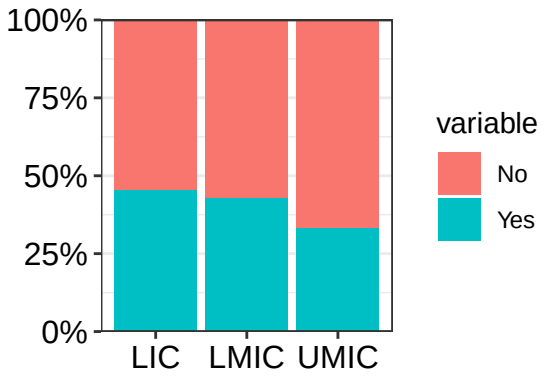



```
## This is actually the same as the previous plot so is achieving the aim.
## But the unflipped plot isn't that great given the order of the year
## Hence why wanting to flip

# Better flipped
# This way, new fill levels get added on the end
gapminder %>%
  ggplot(aes(x = fct_rev(continent), fill = fct_rev(factor(year)))) +
  geom_bar() +
  coord_flip() +
  scale_fill_brewer(palette = "Paired", breaks = rev, direction = -1)
```



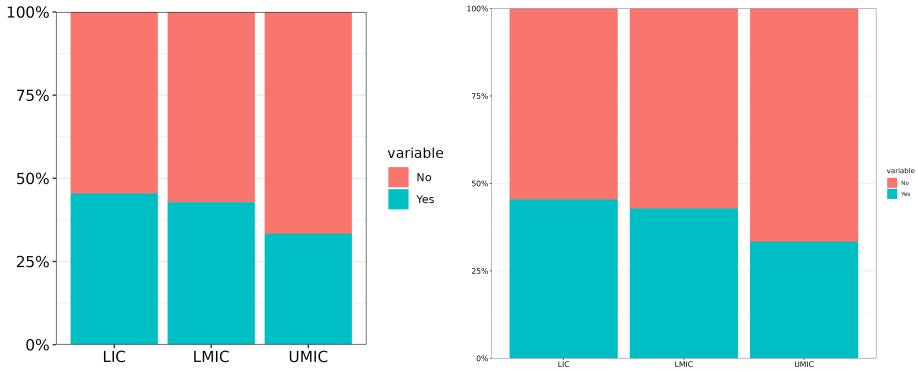
6.3 Axis font size

```
# OPTION 1: theme(axis.text = element_text(size = 12, colour = "black"))
# OPTION 2: width and height arguments of ggsave()

library(tidyverse)
library(scales)

# made-up example data
mydata = tibble(group    = c("UMIC", "LMIC", "LIC") %>% rep(each = 2),
                value    = 1:6,
                variable = c("Yes", "No") %>% rep(3))

mydata %>%
  ggplot(aes(x = group, y = value, fill = variable)) +
  geom_col(position = "fill") +
  scale_y_continuous(labels = percent, expand = c(0, 0)) +
  theme_bw() +
  # OPTION 1: change font with theme()
  theme(axis.text  = element_text(size = 12, colour = "black"),
        axis.title = element_blank())
```




```
# OPTION 2: play around with export size. Since PDF will always have max resolution an
# but changing width and height modifies text size
mywidth  = 5
myheight = 4
#ggsave("barplot_5x4.pdf", width = mywidth, height = myheight)

mywidth  = 10
myheight = 8
#ggsave("barplot_10x8.pdf", width = mywidth, height = myheight)
```

Same plot 5x4 inches vs 10x8 inches:

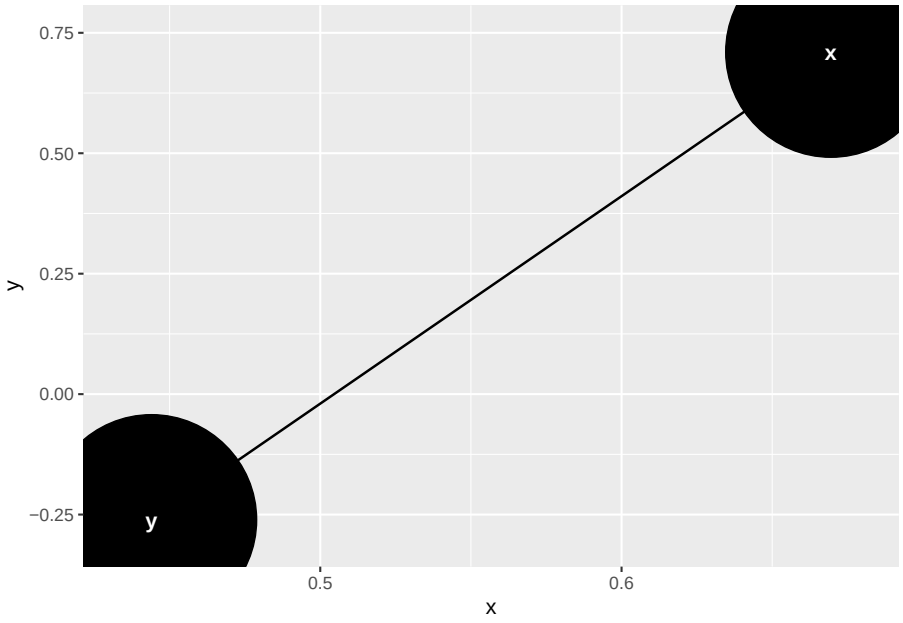


6.4 Shorten Arrows on a DAG

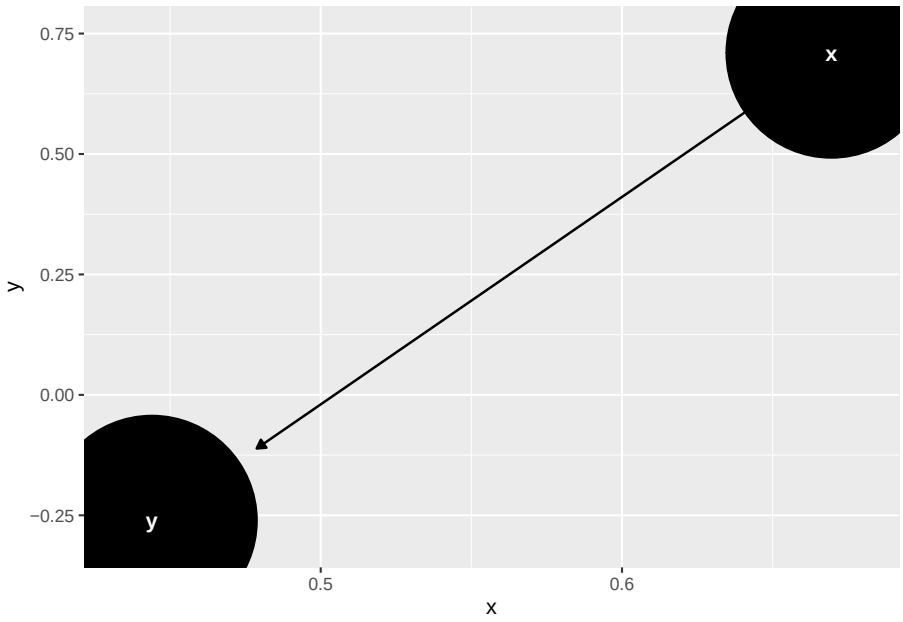
Plot 1 with large nodes and arrows not shortened:

```
DAG <- ggdag::dagify(y ~ x)
DAG <- ggdag::tidy_dagitty(DAG)
```

```
PLOT1 <- ggdag::ggdag(DAG, node_size = 50)
PLOT1
```




```
shorten_dag_arrows <- function(tidy_dag, shorten_distance){
  # Update underlying ggdag object
  tidy_dag$data = dplyr::mutate(tidy_dag$data, slope = (yend - y) / (xend - x), # Calculate slope
                  distance = sqrt((xend-x)^2 + (yend - y)^2), # Calculate total distance of line
                  proportion = shorten_distance/distance, # Calculate proportion by which to be s
                  xend = (1-proportion)*xend + (proportion*x), # Shorten xend
                  yend = (1-proportion)*yend + (proportion*y)) %>% # Shorten yend
    dplyr::select(-slope, -distance, -proportion) # Drop intermediate values
  return(tidy_dag)
}
DAG <- shorten_dag_arrows(DAG, shorten_distance = 0.1)
PLOT2 <- ggdag::ggdag(DAG, node_size = 50)
PLOT2
```


0.75
0.50
0.25
0.00
−0.25
y
0.5
0.6
x
x
y

Chapter 7

Genomics

7.1 Single Cell Analysis

7.1.1 Minimising the size of a Seurat Object

Single cell analyses are ofetn collated in Seurat objects which can be huge. We reduced the size of our Seurat object by pulling the relevant sections using the code below reducing the object to <20% of its original size. This works by only including the sparse matrices. This is not a workable example, but the PBMC dataset could be used if necessary.

```
#library(Seurat)
#library(dplyr)


#Load in the data

pathtoglobaldata <-"mac_shiny/data/Global_sc_object"
object_global<-readRDS(pathtoglobaldata)


# Get sparse assay data
sizetest_global<-GetAssayData(object = object_global, slot = "data") # counts don't work, scaled
object_global_small<- CreateSeuratObject(sizetest_global)

# Add in classifications.

object_global_small@reductions$tsne<-object_global@reductions$tsne # reduction embeddings for tsn
object_global_small$Phenotype<- object_global$Phenotype
```

```
object_global_small@meta.data$final_classification<- object_global@meta.data$final_clas
object_global_small$final_classificaiton<- object_global$final_classification #cell cl
Idents(object = object_global_small) <- "final_classificaiton" #set forever


#same
saveRDS(object_global_small,"mac_shiny/data/global_object_sml")
```

7.2 Nextflow

Nextflow enables scalable and reproducible scientific workflows using software containers. These can either work using singularity (HPC and secure compute), docker (insecure compute - requires root) and using conda. Singularity is the preferred option.

The core workflows can be found at here.

For a basic alignment job for RNASeq or DNASeq, you will require a machine with around 200GB memory and 24 cores. Fast scratch space is advantageous (NVMe fast disk is ideal). Fast disk space is needed for the working directory (cached files and staged files) and for the results storage. At least three times the size of the input data is ideal.

7.2.1 RNASeq example with sbatch (slurm)

```
#!/bin/bash --login
#
#SBATCH --job-name=single-node-multicore ### Replace with the name of your job
#SBATCH --mail-user=thomas.drake@glasgow.ac.uk ### Your email address
#SBATCH --mail-type=END,FAIL # Conditions when an email is sent
#SBATCH --output=%x.%j.out ### output file: <job-name.job-id.out>
#SBATCH --nodes=1 ### Keep the job on one computer/node.
#SBATCH --ntasks=1 ### single task
#SBATCH --cpus-per-task=64  ### each task uses up to 64 cpu cores (defaults to 1 if ab
#SBATCH --time=10-00:00:00 ### WallTime (10 days)
#SBATCH --mem=480G
#SBATCH --partition=compute ### Partition Name
date
hostname
# Set OMP_NUM_THREADS to the same value as --cpus-per-task
# with a fallback option in case it isn't set.
module load nextflow-22.04.0
```

```
module load singularity/singularity-3.8.7
nextflow run nf-core/rnaseq -w "/nfs/sharedscratch2/users/tdrake/nf_work/" --input "/nfs/sharedsc
date
```

7.2.2 DNASeq example with sbatch (slurm)

```
#!/bin/bash --login
#
#SBATCH --job-name=single-node-multicore ### Replace with the name of your job
#SBATCH --mail-user=thomas.drake@glasgow.ac.uk ### Your email address
#SBATCH --mail-type=END,FAIL # Conditions when an email is sent
#SBATCH --output=%x.%j.out ### output file: <job-name.job-id.out>
#SBATCH --nodes=1 ### Keep the job on one computer/node.
#SBATCH --ntasks=1 ### single task
#SBATCH --cpus-per-task=64  ### each task uses up to 64 cpu cores (defaults to 1 if absent)
#SBATCH --time=10-00:00:00 ### WallTime (10 days)
#SBATCH --mem=480G
#SBATCH --partition=compute ### Partition Name
date
hostname
# Set OMP_NUM_THREADS to the same value as --cpus-per-task
# with a fallback option in case it isn't set.
module load nextflow-22.04.0
module load singularity/singularity-3.8.7
nextflow run nf-core/sarek -w "/nfs/sharedscratch2/users/tdrake/nf_work/" --input "/mnt/data/R09/
date
```

7.2.3 ATACSeq example with sbatch (slurm)

```
#!/bin/bash --login
#
#SBATCH --job-name=single-node-multicore ### Replace with the name of your job
#SBATCH --mail-user=thomas.drake@glasgow.ac.uk ### Your email address
#SBATCH --mail-type=END,FAIL # Conditions when an email is sent
#SBATCH --output=%x.%j.out ### output file: <job-name.job-id.out>
#SBATCH --nodes=1 ### Keep the job on one computer/node.
#SBATCH --ntasks=1 ### single task
#SBATCH --cpus-per-task=64  ### each task uses up to 64 cpu cores (defaults to 1 if absent)
#SBATCH --time=10-00:00:00 ### WallTime (10 days)
#SBATCH --mem=480G
#SBATCH --partition=compute ### Partition Name
```

```
date
hostname
# Set OMP_NUM_THREADS to the same value as --cpus-per-task
# with a fallback option in case it isn't set.
module load nextflow-22.04.0
module load singularity/singularity-3.8.7
nextflow run nf-core/atacseq -w "/nfs/sharedscratch2/users/tdrake/nf_work/" --input "/
date
```

Chapter 8

Programming in rlang

8.1 rlang

8.1.1 What is rlang?

rlang is a low-level programming API for R which the tidyverse uses (meaning it speaks to R in as R like way as possible, rather than a 'high-level' - high level is more user orientated and interpretable). It enables you to extend what the tidyverse can do and adapt it for your own uses. It's particularly good to use if you're doing lots of more 'programming' type R work, for example, building a package, making a complex shiny app or writing functions. It might also be handy if you're doing lots of big data manipulation and want to manipulate different datasets in the same way, for example.

In this chapter we'll discuss some uses of it.

8.1.2 Dynamic calling of variables using dplyr

In this example, say we have a tibble of variables, but we want to apply dynamic changes to it (so we feed R a variable, that can change, either using another function like `purr::map` or in a ShinyApp). In this instance, specifying each variable and each different possible consequence using different logical functions would take forever and be very clunky. So we can use rlang to simply put a dynamic variable/object through the same function.

We'll use an example where we want to summarise by different outcomes in a dynamic way.

```
library(tidyverse)
#Make data - here nonsense numbers on deaths per 100k from guns say

example_data = tibble(countries = c('UK', 'USA', 'Pakistan', 'Mexico', 'Ireland', 'Esto
                      region = c('Europe', 'Americas', 'Asia', 'Americas', 'Europe', 'E
                      Death_from_guns = c(1, 200, 150, 450, 3, 3.5),
                      Death_from_smoking = c(100, 300, 140, 150, 120, 300))


#example function for summarising using dynamic variables and bang bangs
#note metric must be numeric
summarise_feature = function(df, col_var, ...){
  require(tidyverse)

  wurly_curly = function(.){#The wurly_curly function makes things nicer
  require(rlang)
  !!quo_name(enquo(.))}

  summary_nm_sum <- paste0("metric_", wurly_curly(col_var))#The new LHS variable must b

  df %>%
    group_by(...) %>%
    summarise(!!summary_nm_sum := sum({{col_var}}))
}


#output new variable
example_data %>%
  summarise_feature(Death_from_guns, region)

#How to dynamically change names using mutate and rlang
#Bang bang variables into mutate/tidyverse(so they can be dynamically changed)

#Select metric
metric = 'metric_Death_from_guns' #example but can be any named column

metric_mutate = paste0('prefix_', metric) #some reason doesn't take these within the m

example_data %>%
  summarise_feature(Death_from_guns, region) %>%
  mutate(!!metric_mutate := !!rlang::sym(metric) * 2)#apply arbitary multiplication by
```

Chapter 9

Server admin

9.1 RStudio (argonaut and argosafe)

9.1.1 Scheduled script (CRON)

Most scheduling is best done using RStudio Connect (argoshare), see below. For certain applications, it may be desirable to do this on argonaut or argosafe. For instance, if a scheduled job is required to be done in a trusted research environment.

This can be done by adding a cron job.

The following will execute the `R` script at 7AM everyday. A log file is written within the project.

```
crontab -e

# Add line (using e.g. nano editor)
00 07 * * * Rscript /home/user/project/script.R &>> /home/user/project/cron.log
```

The cronjob does not load the user environment. If you are using environment variables in R, e.g. API tokens, then these can be loaded within the R script. Include this at the top of your script.

```
# Log
print(paste("Start:", Sys.time()))

# Environment variables
readRenviron("/home/user/.Renviron")
```

9.2 RStudio Connect: argoshare

9.2.1 Delete user

```
# Delete user from argoshare
## Connect must be stopped first
sudo systemctl stop rstudio-connect

## List all users
sudo /opt/rstudio-connect/bin/usermanager list

## Find guid for the user you want to delete, looks like below.
## Delete user - need to say y/n
sudo /opt/rstudio-connect/bin/usermanager delete --users --user-guid 46cb5adb-4036-451

## Restart connect
sudo systemctl start rstudio-connect
```

9.3 Shiny server opensource setup - UoE Eleanor

This guide assumes you're already on the University of Edinburgh local network. If you're working from home, you'll first need to connect the UoE VPN. It also assumes you already have storage allocated to your project on Eleanor, and that your own computer is either Linux-based or a Mac.

9.3.1 Setting up Ubuntu Instance

- Log into the Openstack dashboard on Horizon - https://horizon.ecdf.ed.ac.uk/dashboard/
- Click on Instances and launch a new instance (Ubuntu 18.4, t1.small (free tier))
- Click on key pairs and create an SSH keypair, for securely logging on to the new instance.
- Save the private key to a secure place on your computer (~/.ssh is the usual spot for these things). Change the files permissions to make it secure. Eleanor will not let you SSH without this step. In a local terminal, type:

```
chmod 400 PATH-TO-PRIVATE-KEY
```

- Click on Network > Floating IPs and create a new floating IP. Associate it with your new instance. Make a note of your floating IP – this is the address you'll use to access your instance from now on.
- Click on Security Groups and create a new security group. Add new rules to this group, allowing TCP ingress and egress on the following ports: 80 (HTTP), 22 (SSH), 3838 (Shiny Server).
- Log into your instance using SSH. In a local terminal type:

```
ssh -i PATH-TO-PRIVATE-KEY ubuntu@YOUR-FLOATING-IP

#All being well, you'll now have a terminal for your lovely new Ububtu instance.
#Get your instance up-to-date. Run:

sudo apt-get update && sudo apt-get upgrade
```

9.3.2 Install Nginx

Nginx is the fast and light webserver that will host you Shiny code.

```
#Install nginx
sudo apt-get -y install nginx
```

- In a browser, enter your floating IP address. All being well, you should now get a "Welcome to Nginx" page.

9.3.3 Installing R

Installing the latest version of R isn't as easy as apt-get R. Sources and keys need to be added to the source.list file, then R can be installed (the latest version). The first command sets the sources to the RStudio mirror, then grabs keys to authenticate the installation. Enter the following commands in order:

```
sudo sh -c 'echo "deb http://cran.rstudio.com/bin/linux/ubuntu bionic-cran35/" >> /etc/apt/source
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E298A3A825C0D65DFD57CBB651716619E08

sudo apt-get update
sudo apt-get -y install r-base


#Now load R
R
#Enter the following in the R console
```

```
sessionInfo()
quit()
```

9.3.4 Installing Shiny Server

There are a few external libraries that are required to use Shiny Server. Let's install those, the R package shiny then move on to installing Shiny Server itself.

```
sudo apt-get -y install gdebi-core
sudo su - -c "R -e \"install.packages('shiny', repos='http://cran.rstudio.com/')\""
wget https://download3.rstudio.org/ubuntu-14.04/x86_64/shiny-server-1.5.13.944-amd64.d
sudo gdebi shiny-server-1.5.13.944-amd64.deb
```

Back in your browser, go to: YOUR-FLOATING-IP:3838 All being well, you'll get a Shiny Server test page.

9.3.5 Installing other Linux packages

You've got base-R installed and running now, but it's likely your Shiny app will have other unmet dependencies. Here are a few common packages you'll probably need, and the command needed to install them. Depending on your app, there may be others.

```
sudo apt-get install -y \
    pandoc \
    pandoc-citeproc \
    libssl-dev \
    libcurl4-gnutls-dev \
    libcairo2-dev \
    libgdal-dev \
    libgeos-dev \
    libproj-dev \
    libxml2-dev \
    libxt-dev \
    libv8-dev \
    git
```

9.3.6 Installing R packages

You'll probably also need a bunch of extra libraries in R (we've already installed Shiny, but that's it). You install these from within R, rather than via apt-get.

This is the syntax for installing R packages on the server – just replace the package names with the ones your app requires.

```
sudo su - -c "R -e \"install.packages(c('rmarkdown', 'ggplot2', 'dplyr', 'sp', 'rgdal', 'rgeos'),
```

NB: Installing R packages can take a long time. As we're only using the small, free instance in this guide, you may find it grinds to a halt completely when trying to compile large libraries like seurat. This probably means you've run out of system memory. If that happens - see Setting up a swap file at the end of this guide.

9.3.7 Loading your app

Assuming you've developed your app on a local R-Studio server, getting it onto its new home can be surprisingly tricky. I've found by far the easiest way is to use an SFTP client, such as Cyberduck, giving it the username and key pair you've already set up and just sending the files straight into your home directory.

Once you've transferred them, you'll need to move them, including subdirectories, into /srv/shiny-server/. Again, in your browser, go to YOUR-FLOATING-IP:3838. You'll either see your working app, or an error page listing missing dependencies. If it's the latter, follow the instructions above for installing R packages.

If you're not getting any useful error messages on the site, check out the app-specific log files in /var/log/shiny-server/

9.3.8 Make your app the default index page

You probably don't want visitors to your site to specify that they want port 3838, so now we're going to tell nginx to send visitors directly to your app. First, we're going to edit the nginx config file.

```
sudo nano /etc/nginx/sites-available/default

# Comment out (#) any other active server config lines, and add this...

server {
    listen 80;

    location / {
        proxy_pass http://127.0.0.1:3838/;
        proxy_redirect http://127.0.0.1:3838/ $scheme://$host/;
        # note, the following 3 lines added 2016-07-11, see updates section
```

```
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
     }
    }
```

Next, we're going to update the Shiny server config file

```
sudo nano /etc/shiny-server/shiny-server.conf

#Add the following (if it's not already there, which it may well be):

server{
    listen 3838 127.0.0.1;

    location / {
      site_dir /srv/shiny-server;
      log_dir /var/log/shiny-server;
      directory_index on;
    }
}
```

Go to your floating IP in the browser, without the 3838, and all should be well.

9.3.9 Getting a user-friendly domain name

If you want to share your app with the outside world, you'll need to open it up beyond the UoE network and, preferably, give it a catchy URL.

Making your site public: Back in OpenStack, under the Network menu, click on Floating IP and disassociate your floating IP. Now go up to Router and click clear gateway. Click Set Gateway and select Floating Network Public. Back in Floating IPs, click Associate and select the external IP address (probably starting 129.x.x.x). Under port, select your instance. Enter your external IP address into your browser, and your website should appear. It is now available outside the University network.

Setting up your domain name: Your first step here is obviously to buy a domain name from one of the many providers on the internet.Once you have this, use your domain provider's dashboard to edit DNS settings. You're looking to change the ANAME or CNAME settings – delete what's already there and set up a new ANAME record, with the public IP address of your site.

9.3.10 Appendix 1 – Password protecting your site

You may want to add a basic password to your site. This is easily accomplished (though there are more robust password protection mechanisms out there, if you're dealing with potentially sensitive data).

First, make sure you have Apache2-utils installed:

```
#First, make sure you have Apache2-utils installed:
sudo apt-get install apache2-utils

#Now, stop nginx and shiny:
sudo service nginx stop
sudo stop shiny-server

#Next, you'll need to go back into that Nginx config file, with
sudo nano /etc/nginx/sites-available/default

# Add the following two lines in the "location" section
location / {
      proxy_pass http://127.0.0.1:3838/;
      proxy_redirect http://127.0.0.1:3838/ $scheme://$host/;
      auth_basic "Username and Password are required";
      auth_basic_user_file /etc/nginx/.htpasswd;
    }
```

Once that's done, you'll need to edit Shiny Server's conf file so it only serves to loaclhost. Otherwise users would be able to creep around your authentication by going to port 3838.

```
sudo nano /etc/shiny-server/shiny-server.conf

#Copy and paste the below to your shiny-server.conf.
server{
    listen 3838 127.0.0.1;

    location / {
    site_dir /srv/shiny-server;
    log_dir /var/log/shiny-server;
    directory_index on;
    }
}
```

You're now ready to create the username and password visitors must enter to view your site.

```
cd /etc/nginx
sudo htpasswd -c /etc/nginx/.htpasswd USERNAME

#Restart Nginx and Shiny and you're good to go.
sudo service nginx start
sudo start shiny-server
```

9.3.11 APPENDIX 2 – Setting up a swap file

If your instance grinds to a halt while compiling R packages, you've most likely run out of memory and will need to set up a swap file. Just enter the following commands:

```
sudo fallocate -l 2G /swapfile
sudo dd if=/dev/zero of=/swapfile bs=1024 count=1048576
sudo chmod 600 /swapfile
sudo mkswap /swapfile
sudo swapon /swapfile

#Finally, make sure everything as it should be by running:
sudo free -h
```

Chapter 10

REDCap

10.1 API pull

1. REDCap's API can be used to automatically pull records from a REDCap project. Ask your project manager to give you API Export rights and then request an API token via the API page of the project (left hand menu when inside a project). The API option will only appear after the project manager can given you API rights.

2. Put the API token in your .Renviron file and Restart R. The easiest way to edit your .Renviron file is to run this in the Console: `usethis::edit_r_environ()`.

Example format within the .Renviron file:

```
redcap_token = ABC10000000000000000000000001DFG
redcap_uri = "https://redcap.cir.ed.ac.uk/usher/api/"
```

3. Go to the API playground in REDCap (left-hand menu inside a project) and from API Method select "Export Records".

4. Scroll down and look at the R code example. Now, we've not found that copying this exact example works very well for us. But looking at the format can be useful if you've made custom selections at the top, e.g., to pull specific records or variables. In most cases though, you can ignore the example and copy these lines:

```
library(tidyverse)
patient_data_orig <- list("token"=Sys.getenv("redcap_token"),
                          content='records',
                          action='export',
                          format='csv',
                          type='flat',
                          csvDelimiter='',
                          rawOrLabel='label',
                          rawOrLabelHeaders='raw',
                          exportCheckboxLabel='true',
                          exportSurveyFields='false',
                          exportDataAccessGroups='false',
                          returnFormat='json') %>%
  httr::POST(Sys.getenv("redcap_uri"), body = ., encode = "form") %>%
  httr::content()
```

`httr::content()` runs `readr::read_csv()` to process the result of the API call. So any arguments that you might need to pass to `read_csv()`, can be passed in there, e.g.:

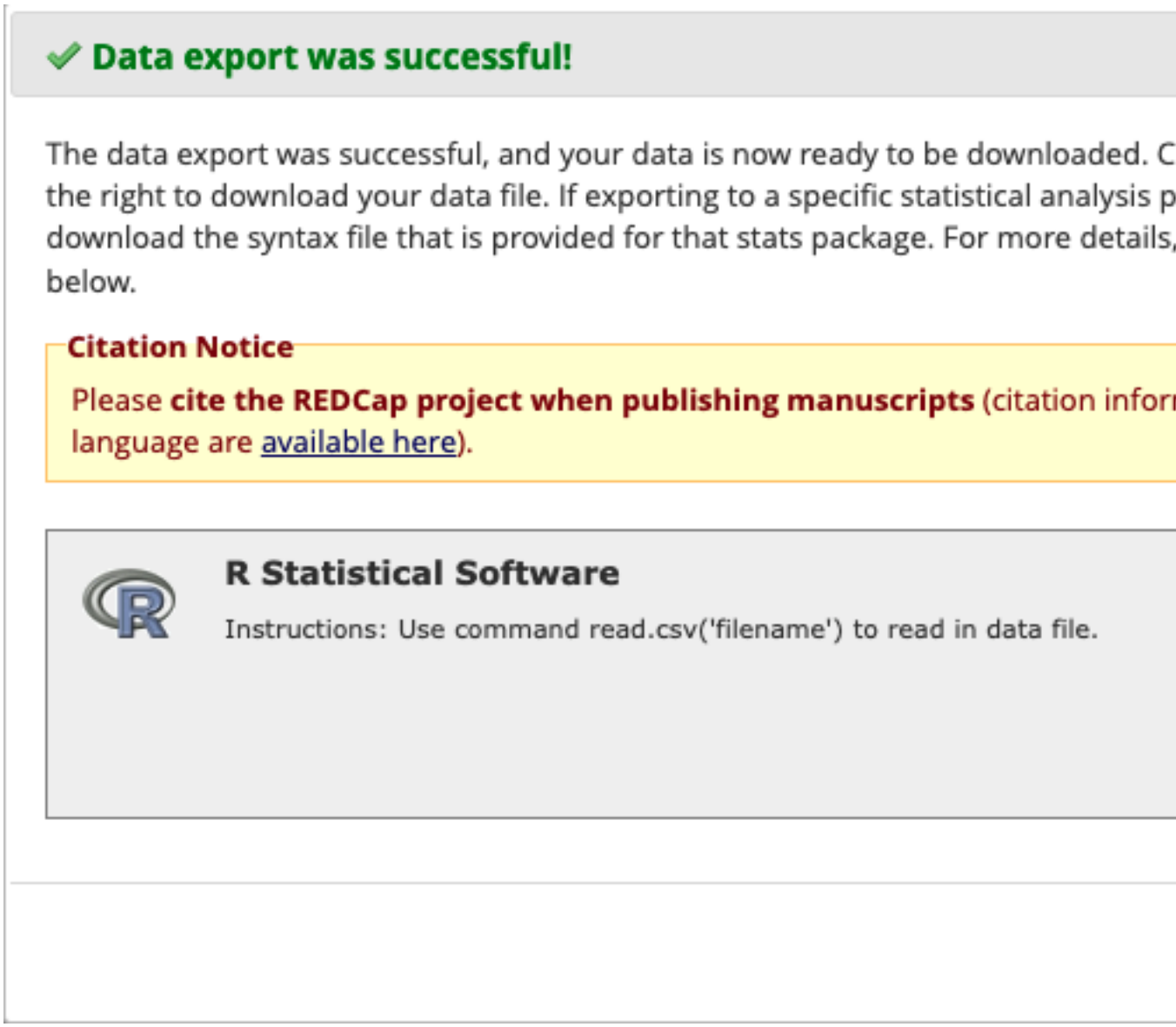
```
httr::content(guess_max = Inf, col_types = cols(.default = "c"))
```

The above code works well for the vast majority of projects. If, however, the project is huge then you'll need to pull data in batches using `redcap_read()` from `library(REDCapR)`: https://github.com/SurgicalInformatics/cocin_ccp/blob/master/01c_data_pull.R

5. `rawOrLabel='raw'` vs `rawOrLabel='label'`. Pulling raw data will give you categorical variables in their raw coding, e.g., `1, 2` instead of `"Male", "Female"`. Pulling 'labels' will give you the latter, but their factor levels will be ordered alphabetically (as the information on which one was 1 and which one 2 is lost/not pulled). If I'm working with a small dataset with categorical variables that only have a couple of levels each then I find it easier to pull 'labels' and use `fct_relevel()` to order them as necessary. In most cases, however, you'll want to use the exact same ordering that's been set-up on the database. To do this, you'll have to pull 'raw' and apply what's called a REDCap's factoring script.

10.2 Applying a REDCap factoring script

1. Getting the factoring script: leaving the API playground, click on "Data Exports, Reports, and Stats". On the line where it says "All data", click on Export Data, then select R Statistical Software, then click on Export Data. Then click on the R file:



Do not click on DATA CSV, because:

- this exact data is already coming via the API pull
- we don't download data onto our computers but pull it directly from REDCap to our secure analysis servers

2. Move the factoring script to your project directory. The script needs editing to work well with the packages we use. Edit the factoring script as such:

- Delete the first 7 lines.
- Instead, add in

```
library(finalfit)
library(magrittr)
```

3. Move the labelling section to the bottom of the script. This is because we are about to change it from Hmisc labels to finalfit labels, but the `factor()` function strips the latter. (Tidyverse functions, on the other hand, strip the former.)

4. Change the labels() code from, e.g.,

```
label(data$record_id)="Record ID"
label(data$redcap_data_access_group)="Data Access Group"
label(data$sex)="Sex"
label(data$age)="Age"
```

to

```
data$record_id %<>% ff_label("Record ID")
data$redcap_data_access_group %<>% ff_label("Data Access Group")
data$sex %<>% ff_label("Sex")
data$age %<>% ff_label("Age")
```

Steps:

- Remove `label(` from the beginning of each line
- Replace `)=` with `%<>% ff_label(`
- Complete each line with `)`

Hints and tips:
Consider using multiple cursors (click and drag your mouse while holding down Alt on Windows/Linux, or option on a Mac).
When doing find and replace, make sure to click "In selection" so don't accidentally remove from other sections.
For adding `)` to the end of line, use:

5. Remove `.factor` from column names as this creates a duplicate of each column.

6. Apply the factoring script on the pulled data: the factoring script assumes the tibble is called `data`. Instead of going through and changing each instance of this name, do this:

```
data = pulled_data
source("REDCap_factoring_script.r")
pulled_data = data
rm(data)
```

10.3 Alternative to the factoring script: `redcap_label()` from `library(collaboratorR)`

Optional: `redcap_label()` from `library(collaboratorR)` will do factoring for you by applying information from the project's metadata (which is also pulled via the API). Currently, it uses Hmisc labels so not fully compatible with the tidyverse.

10.4 Scannable barcodes in REDCap

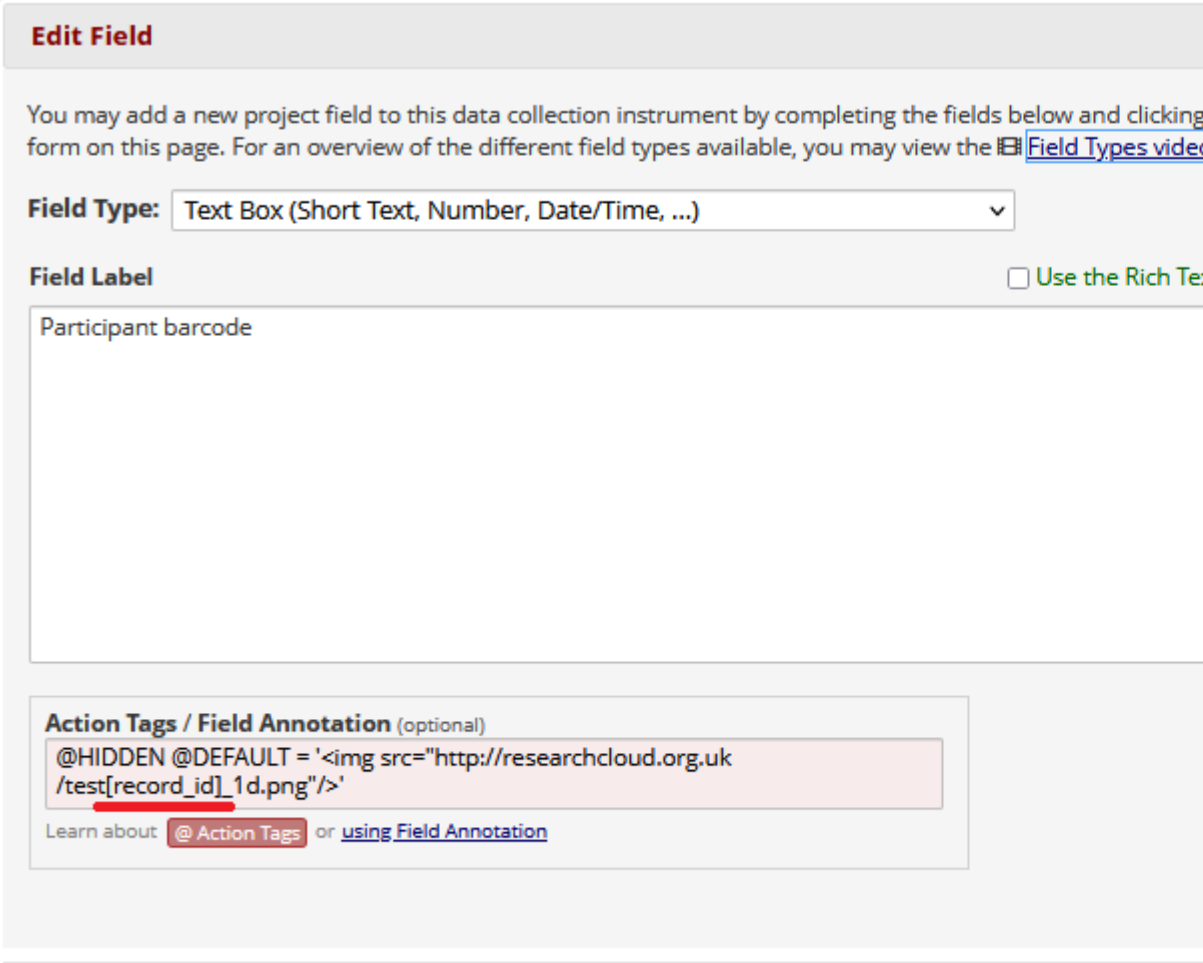
Barcodes (1D - like in shops, or 2D - like QR codes) can be useful to allow machines to read data and avoid human mistakes. It can also facilitate de-identification of samples (so they can be processed / handled without compromising any personal information). One use case for this might be if you are doing a study of blood biomarkers on serum, where a separate sample management system might be used to coordinate sample tracking within a laboratory (like a laboratory information management system - LIMS).

To do this, you will need to generate the barcodes, with the appropriate record number in the file name. In this example - we use the `record_id` field in REDCap as the research identifier (NOT a personal identifier) to embed scannable barcodes into REDCap. The advice is to generate many fold more barcodes than you will need. These barcodes should be served either from the web server OR from a public server. Barcodes should not include any personal information.

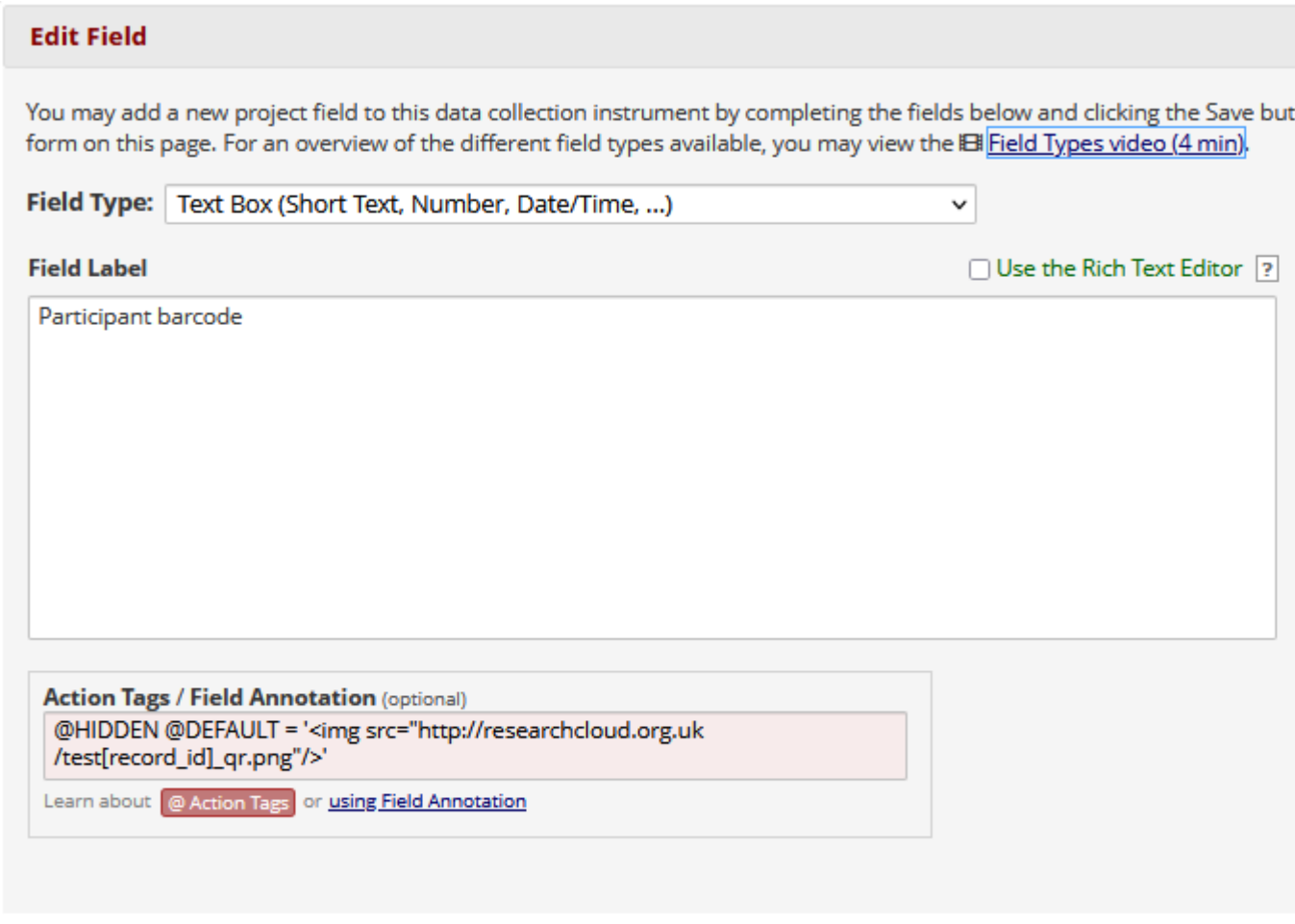
Once you have generated barcodes / QR codes with appropriate titles, then in REDCap set up three new fields - 1) an image / descriptive text field, then 2 x text fields.

In the text fields, used the `@HIDDEN` and `@DEFAULT` action tags to embed the images, and pipe in the `[record_id]` into the titles.

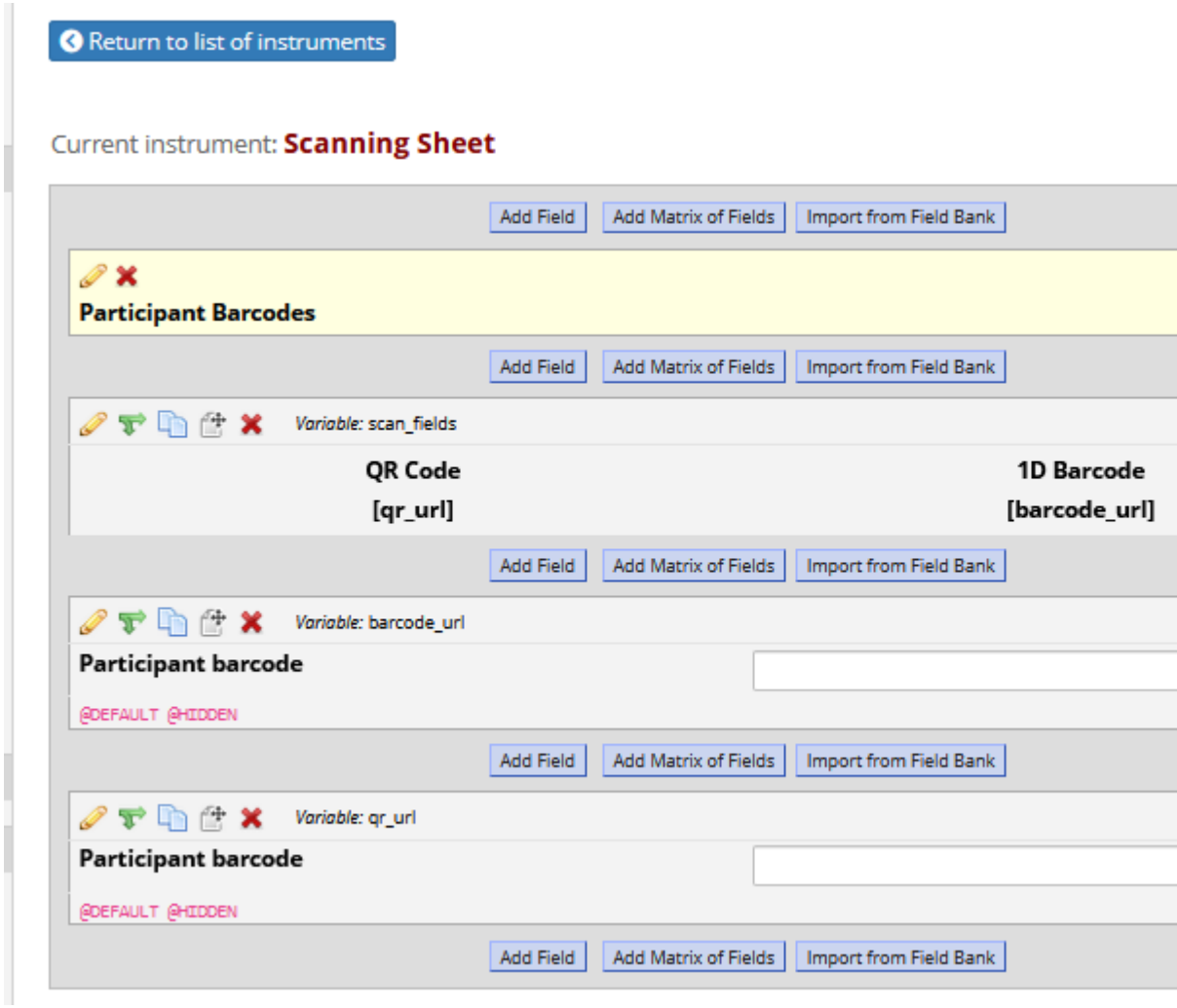
Piped code example for 1D barcode:



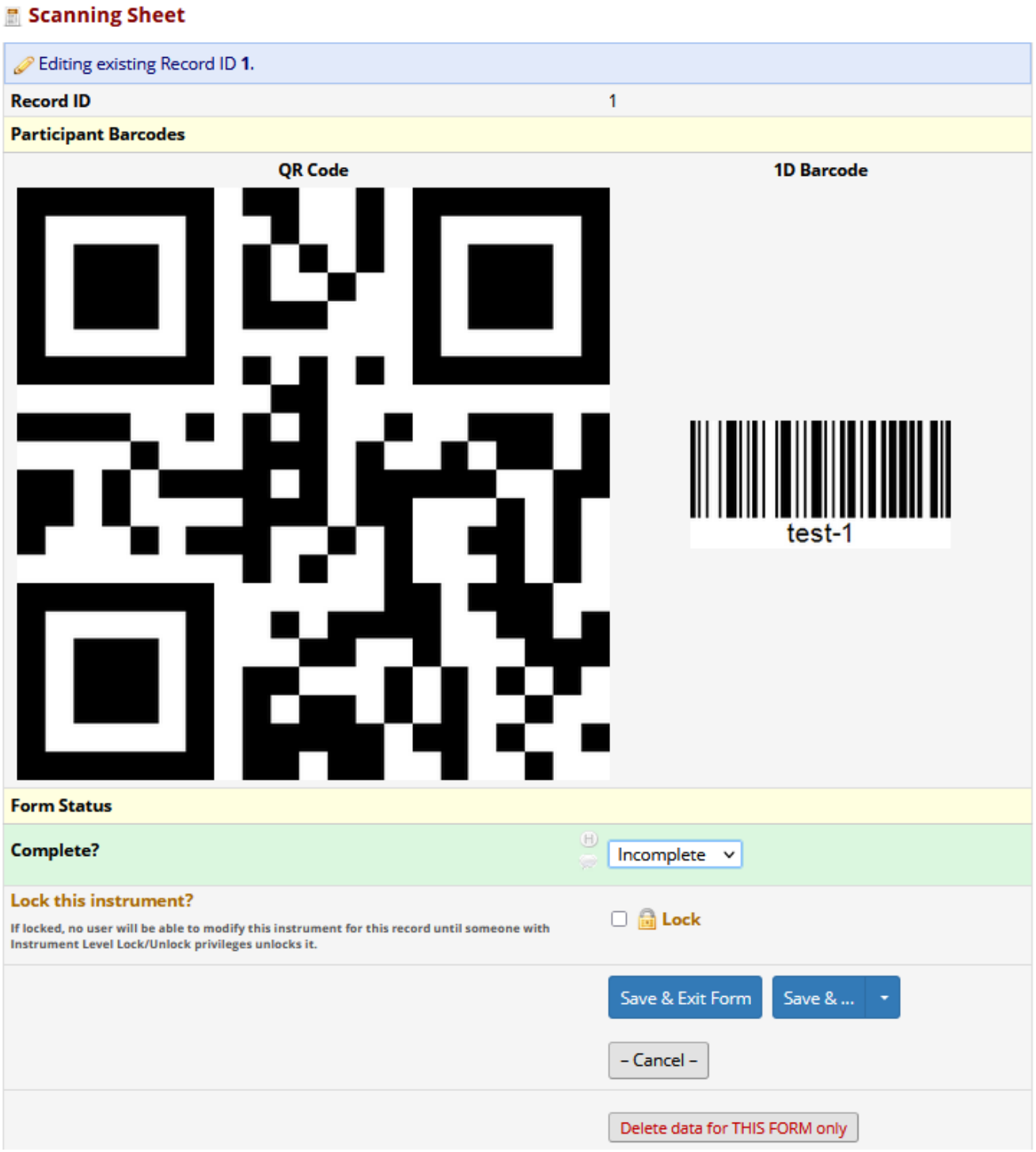
Piped code example for 2D barcode:



Field layout in REDCap:



Appearance after adding a record:


Scanning Sheet
Editing existing Record ID 1.
Record ID
1
Participant Barcodes
QR Code
1D Barcode
test-1
Form Status
Complete?
Incomplete
Lock this instrument?
If locked, no user will be able to modify this instrument for this record until someone with Instrument Level Lock/Unlock privileges unlocks it.
Lock
Save & Exit Form
Save & ...
– Cancel –
Delete data for THIS FORM only

Chapter 11

Working with digital pathology data and python

Histopathology is used to analyse tissues from the body under the microscope. When a sample of tissue is taken during and operation or biopsy, it is put into a preservative, dehydrated and then impregnated with paraffin wax. It may also be frozen. This allows the tissue to be very thinly sliced and mounted onto glass slides, where the tissue can then be stained and studied under the microscope. Different types of stains can be used to look at different things.

These stained slides can be scanned with a slide scanner to allow the image to be digitised using scanners. There are two main types of images. Brightfield - generated by using white light, which requires deconvolution to produce separate colour channels. Fluorescence - generated using a light source to excite fluorescent probes, this typically generates images with 2+ colour channels.

The file formats produced by these scanners varies by manufacturer and scanning technique. Essentially, data is stored as tiled TIFF images within what is possibly an XML structure. There are usually multiple images produced, not only of the tissue but also of the slide, including a macro image (no magnification, sometimes helps scanner locate tissue), and the slide label.

Unfortunately, it's quite a difficult format to interrogate easily. However, there is a nice program written that can solve this issue, known as OpenSlide.

OpenSlide has C, Python and Java versions. The python version is probably easiest to interact with, either directly in Python, or it can be combined with R using `reticulate`.

This chapter will not go into depth about issues around environments or how to install python properly. Please familiarise yourself with how to use `pip`, particularly `pip install` and set / activate python environments.

You can use RStudio for working in python too and this is what is suggested here. Other formats exist including jupyter notebooks.

For more information, Python for microscopists has excellent resources.

11.1 Installing OpenSlide

First install the linux version of openslide - the python files act as a wrapper to this system library.

```
apt-get install python3-openslide
```

To install openslide and the libraries we will need, first create a python environment. Do this inside the folder you are going to be working in.

```
virtualenv -p python3 openslide_env
```

You can then either activate the virtual environment to install python libraries there, or install them system/user wide.

```
python3 -m pip install openslide-python
```

We will also need some other libraries

```
python3 -m pip install numpy matplotlib tifffile
```

Installation is fairly straighforward -if there are issues, do it as a sudo user. Most of the problems will appear if you are trying to use different or conflicting versions of python or if there are incorrect permissions for installation.

11.2 Using OpenSlide

Once openslide is installed, we can now write python scripts. Do this in RStudio or other IDE in the same folder as your virtual environment. You can create a project in RStudio using this existing folder, then add new python scripts. Python scripts have the suffix `.py`.

First, start by checking you can successfully import the libraries.

```
from openslide import open_slide
import openslide
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt
import tifffile as tiff
```

If you get an error- try installing the libraries with `pip` as above.

11.2.1 Opening a slide and inspecting properties

```
file_path = '/home/test/slides/myslide.svs' # replace with where your slides are

slide_in = open_slide(file_path)

slide_in_props = slide_in.properties

slide_in_assc_images = slide_in.associated_images.items() # see if there are any associated image
```

11.3 Whole workflow

Here is the whole workflow for extracting magnified tiles of the image, taken from Python for microscopists.

```
file_path = '/home/test/slide_ins/myslide_in.svs' # replace with where your slide_ins are

slide_in = open_slide(file_path)

slide_in_props = slide_in.properties

print(slide_in_props)

print("Vendor is:", slide_in_props['openslide.vendor'])
print("Pixel size of X in um is:", slide_in_props['openslide.mpp-x'])
print("Pixel size of Y in um is:", slide_in_props['openslide.mpp-y'])

#Objective used to capture the image
objective = float(slide_in.properties[openslide.PROPERTY_NAME_OBJECTIVE_POWER])
print("The objective power is: ", objective)

# get slide_in dimensions for the level 0 - max resolution level
```

```
slide_in_dims = slide_in.dimensions
print(slide_in_dims)

#Get a thumbnail of the image and visualize
slide_in_thumb_600 = slide_in.get_thumbnail(size=(600, 600))
slide_in_thumb_600.show()

#Convert thumbnail to numpy array
slide_in_thumb_600_np = np.array(slide_in_thumb_600)
plt.figure(figsize=(8,8))
plt.imshow(slide_in_thumb_600_np)


#Get slide_in dims at each level. Remember that whole slide_in images store informatio
#as pyramid at various levels
dims = slide_in.level_dimensions

num_levels = len(dims)
print("Number of levels in this image are:", num_levels)

print("Dimensions of various levels in this image are:", dims)

#By how much are levels downsampled from the original image?
factors = slide_in.level_downsamples
print("Each level is downsampled by an amount of: ", factors)

#Copy an image from a level
level3_dim = dims[2]
#Give pixel coordinates (top left pixel in the original large image)
#Also give the level number (for level 3 we are providing a valueof 2)
#Size of your output image
#Remember that the output would be a RGBA image (Not, RGB)
level3_img = slide_in.read_region((0,0), 2, level3_dim) #Pillow object, mode=RGBA

#Convert the image to RGB
level3_img_RGB = level3_img.convert('RGB')
level3_img_RGB.show()

#Convert the image into numpy array for processing
level3_img_np = np.array(level3_img_RGB)
plt.imshow(level3_img_np)


#Return the best level for displaying the given downsample.
SCALE_FACTOR = 32
```

```
best_level = slide_in.get_best_level_for_downsample(SCALE_FACTOR)
#Here it returns the best level to be 2 (third level)
#If you change the scale factor to 2, it will suggest the best level to be 0 (our 1st level)
##################################

#Generating tiles for deep learning training or other processing purposes
#We can use read_region function and slide_in over the large image to extract tiles
#but an easier approach would be to use DeepZoom based generator.
# https://openslide.org/api/python/

from openslide.deepzoom import DeepZoomGenerator

#Generate object for tiles using the DeepZoomGenerator
tiles = DeepZoomGenerator(slide_in, tile_size=256, overlap=0, limit_bounds=False)
#Here, we have divided our svs into tiles of size 256 with no overlap.

#The tiles object also contains data at many levels.
#To check the number of levels
print("The number of levels in the tiles object are: ", tiles.level_count)

print("The dimensions of data in each level are: ", tiles.level_dimensions)

#Total number of tiles in the tiles object
print("Total number of tiles = : ", tiles.tile_count)

#How many tiles at a specific level?
level_num = 11
print("Tiles shape at level ", level_num, " is: ", tiles.level_tiles[level_num])
print("This means there are ", tiles.level_tiles[level_num][0]*tiles.level_tiles[level_num][1], '

#Dimensions of the tile (tile size) for a specific tile from a specific layer
tile_dims = tiles.get_tile_dimensions(11, (3,4)) #Provide deep zoom level and address (column, ro


#Tile count at the highest resolution level (level 16 in our tiles)
tile_count_in_large_image = tiles.level_tiles[16] #126 x 151 (32001/256 = 126 with no overlap pix
#Check tile size for some random tile
tile_dims = tiles.get_tile_dimensions(16, (120,140))
#Last tiles may not have full 256x256 dimensions as our large image is not exactly divisible by 2
tile_dims = tiles.get_tile_dimensions(16, (125,150))


single_tile = tiles.get_tile(16, (62, 70)) #Provide deep zoom level and address (column, row)
single_tile_RGB = single_tile.convert('RGB')
single_tile_RGB.show()
```

```
###### Saving each tile to local directory
cols, rows = tiles.level_tiles[16]

import os
tile_dir = "images/saved_tiles/original_tiles/"
for row in range(rows):
    for col in range(cols):
        tile_name = os.path.join(tile_dir, '%d_%d' % (col, row))
        print("Now saving tile with title: ", tile_name)
        temp_tile = tiles.get_tile(16, (col, row))
        temp_tile_RGB = temp_tile.convert('RGB')
        temp_tile_np = np.array(temp_tile_RGB)
        plt.imsave(tile_name + ".png", temp_tile_np)
```

11.4 Using python and R together

Using the `reticulate` package, R can communicate with python. This is slightly glitchy at the time of writing. So writing python objects from R seems to not work (using `py$` as mentioned in `reticulate` documentation), but python calling objects in the R environment seems reliable.

To do this we simply access the R environment in python using `r['myobject']`. Lets use the `file_path` example, where you might have used R to save the file path of specific slide file, then you want to use python to extract or manipulate the images.

In R:

```
file_path = '/home/test/slides/myslide.svs' # replace with where your slides are
```

In python:

```
file_path = r['file_path']
slide_in = open_slide(file_path)
```

11.5 Handling errors in python

Most slides will be scanned well, newer scanners use lasers and a variety of different tools to automatically detect where tissue is and the depth at which to scan. However, some files will not scan properly or be corrupted in transfer (files are at least 200M+ per scan, ususally around 500M).

If there are errors, it is quite annoying, particularly given the size of files being handled if a run fails due to errors in opening. To avoid this we can use the python `try` commands, where it will attempt the command but continue on fail.

```
#import pyvips
from openslide import open_slide
import openslide
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt
import tifffile as tiff

#import
file_path = r['file_path']

try:
  slide_in = open_slide(file_path)
  slide_in_props = slide_in.properties
  r['slide_properties'] = slide_in_props
  r['slide_associated_images'] = slide_in.associated_images.items()
except Exception as e:
  r['slide_properties'] = print("Error loading") # doesn't save error
  r['slide_associated_images'] = print("Error loading") # doesn't save error
```

11.6 Iterating over many many slide files

Combining all the above, we can massively extend functionality and the number of slides we can process, for example to extract hundreds of slides at once and apply the same analysis to them.

In R:

```
library(tidyverse)
library(reticulate)
Sys.setenv(RETICULATE_PYTHON = "/home/slides/openslide_env/bin/python")

#list files in a dir
files = data.frame(file_path = list.files('/home/slides',
                                          pattern = '.svs|.scn',
                                          full.names = T, recursive = T))#scn needs different

#make names
files = files %>%
  mutate(file_name =  basename(file_path),
```

```
        label_path = gsub('.svs|.scn', '', file_name),
        out_path = gsub('.svs|.scn', '_label.png', file_path),
        file_info = file.info(file_path))


#first get label properties
slide_properties_list = list()
associate_images_list = list()

#get properties

for(i in 1:length(files$label_path)){
  label_path = files$label_path[i]
  file_path = files$file_path[i]
  out_path = files$out_path[i]
  source_python('label_properties.py', envir = globalenv())
  slide_properties_list[[i]] = as.character(slide_properties)
  associate_images_list[[i]] = as.character(slide_associated_images)
}

openslide_data = data.frame(do.call(cbind, list(files$file_path, associate_images_list
```

In python `label_properties.py`:

```
#import pyvips
from openslide import open_slide
import openslide
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt
import tifffile as tiff

#import
file_path = r['file_path']

try:
  #slide_in = open_slide('/mnt/data/tdrake/light_microscopy/immune_staining_from_colin/
  slide_in = open_slide(file_path)
  slide_in_props = slide_in.properties
  r['slide_properties'] = slide_in_props
  r['slide_associated_images'] = slide_in.associated_images.items()
except Exception as e:
  r['slide_properties'] = print("Error loading")
  r['slide_associated_images'] = print("Error loading")
```

Chapter 12

Citations with R Markdown

Writing publications and other documents requires citations both of academic papers and of R packages.

Zotero provides efficient and free reference management which generates new citation objects with the click of a button in Chrome and easy generation of `.bib` files which can be used to create references in `rmarkdown` and `bookdown`.

Top tip: Do this on Day 1 of your PhD!

12.1 Zotero Set-up

Zotero should be installed and provides 300MB of free storage which covers a few thousand references as long as PDFs are stored elsewhere. The Chrome connector can be downloaded to allow one-click saving of citations. Just click the icon at the top right of the browser to store meta-data and files.

Set up an online account with Zotero so that your data is available remotely.

Zotfile should be installed to help with PDF storage. Download the `.xpi` file for Zotfile. To install **Zotfile** open Zotero and go to `Tools → Add-ons → cog in top right corner` and select downloaded `.xpi` file to install.

Better BibTeX is another plug-in which generates citation keys in a robust and reproducible way which is needed for larger projects (e.g. PhDs) to avoid duplicate citation handles. An `.xpi` file can be downloaded from GitHub and installed in the same way (check Zotero Plug-ins for more recent versions if not installing).

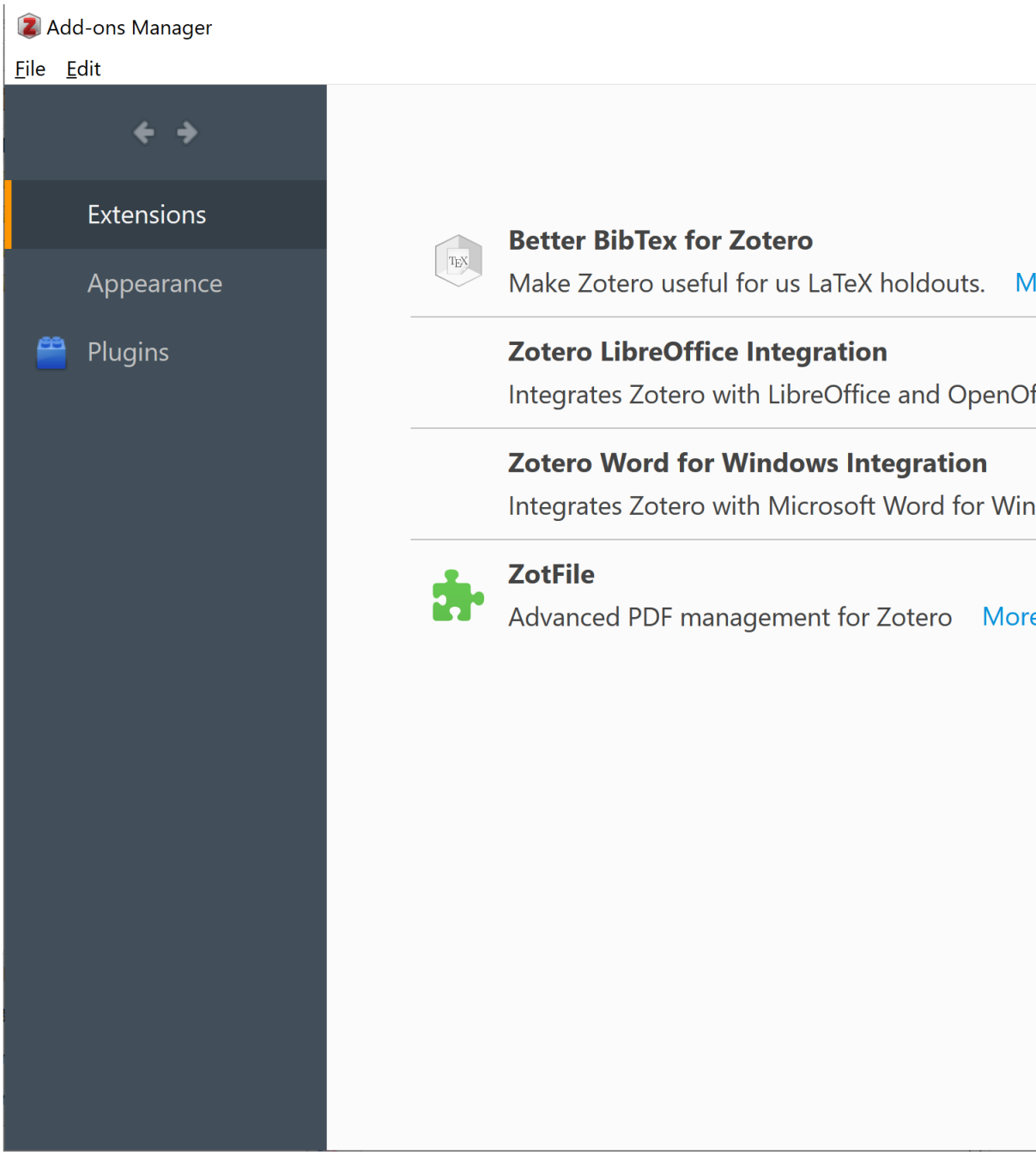


Figure 12.1: Installing a Zotero Plug-in from an .xpi File

12.2 File Storage Set-up

In addition to Zotero set-up it is a good idea to have a cloud storage service to maintain a library of PDFs without using up Zotero storage capacity. These will also need to be accessible to the `Windows Explorer` or the `Mac Finder` - there are several guides on how to do this for Google Drive and DropBox.

12.3 Folder Set-up

12.3.1 Storing PDFs

Create a folder in a cloud storage service to store PDFs which can be accessed from your own file explorer.

12.3.2 Storing Bibliographies

Create a folder which will store your `.bib` file. This can then be used to provide the references for your `.rmd` files.

12.3.3 Subfolder for References

It may be a good idea in Zotero to carefully categorise publications by their content. This makes it very easy to look up publications on a given topic (Zotero allows a citation to be shared across several folders such that a publication documenting an RCT with updated meta-analysis could be in folders for each type of study etc.).

Generally a parent folder will be used to generate a bibliography file and anything likely to be cited in a document / thesis should be in that folder or one of its subfolders.

12.4 Configuring Zotero

12.4.1 ZotFile PDF Preferences

To setup Zotero so that retrieved PDFs are automatically stored and renamed in the cloud storage without consuming the Zotero storage quota go to "Tools → ZotFile Preferences" and on the first tab: **General Settings** and set the folder and subfolder naming strategy for PDFs. Set the location of the files to a Custom location in cloud storage e.g.("~\Google Drive\Zotero PDF Library"). ZotFile will also store retrieved PDFs in subfolders to help with finding PDFs

at a later date. A reasonable setup is to create a subfolder with the first author surname so that all papers authored by one (or more) author with the same name are stored together using the \%a in the subfolder field. Other alternatives are to store PDFs in subfolders using year (\%y); journal or publisher (\%w); or item type (\%T).

Next the **Renaming Rules** tab can be configured to provide sensible names to each of the files (this is essential if PDFs are not to be stored as random strings of characters which provide no meaning). Setting the format to: {%a_}{%y_}{%t} provides names for the PDFs in the format of: `Fairfield_2019_Gallstone_Disease_and_the_Risk_of_Cardiovascular_Disease.pdf`. This shows author, year and first word of title without needing to expand the file name.

In the **Advanced Settings** tab it is strongly recommended to select removal of "Special characters", leaving these in creates problems when knitting to PDF as LaTeX may recognise the special characters in unexpected ways.

12.4.2 General Zotero Settings

Zotero has several configurable settings (accessed through: "Edit → Preferences"). The following settings are generally helpful (left as default if not mentioned):

General:

- Tick the following:
  - Automatically attach associated PDFs
  - Automatically retrieve metadata for PDFs
  - Automatically rename attachments using parent metadata
  - Automatically tag items with keywords and subject headings
  - All options in Group section
- Leave the following unticked:
  - Automatically take snapshots
  - Rename linked files

Sync:

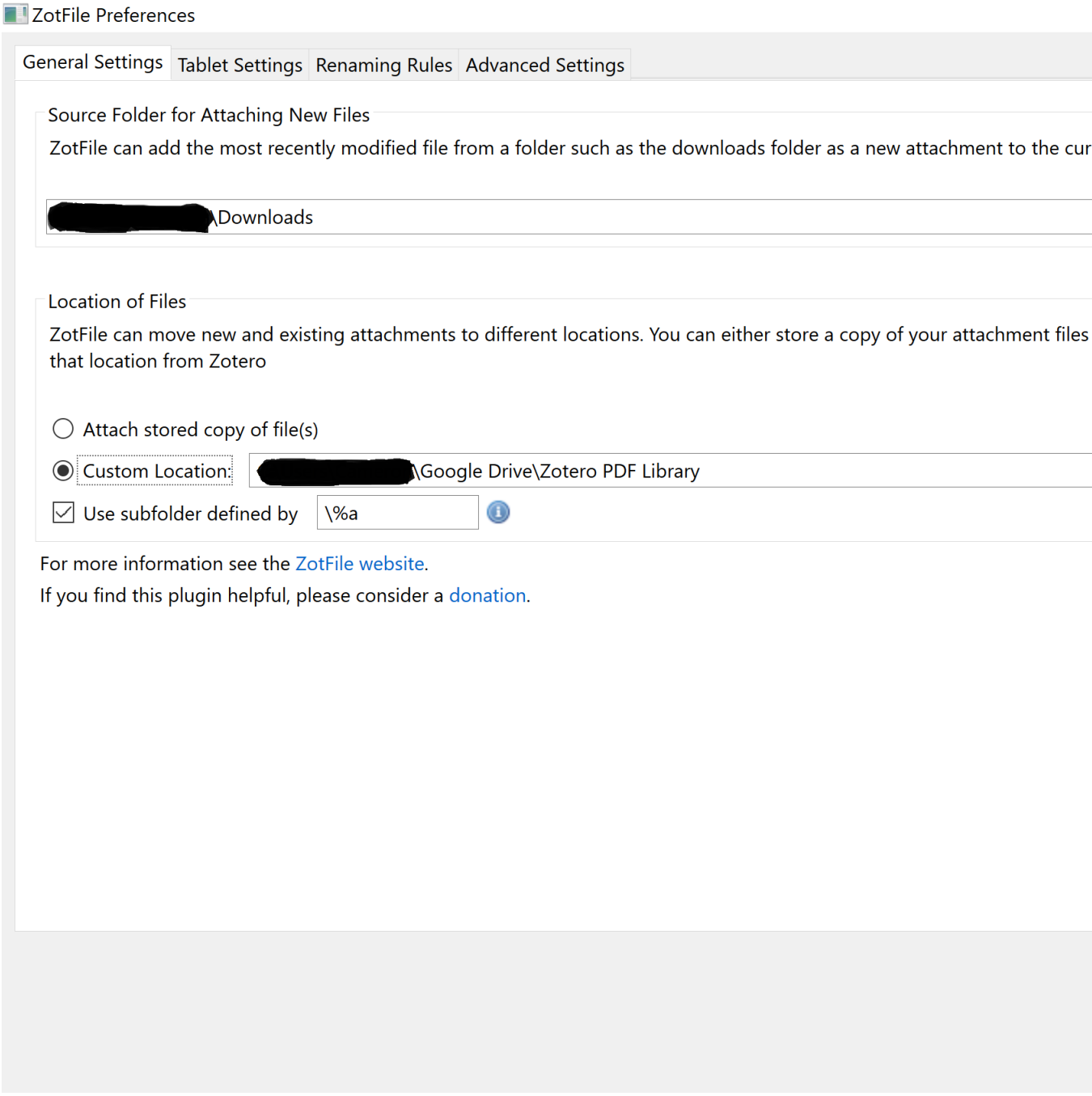


Figure 12.2: ZotFile PDF Storage Preferences

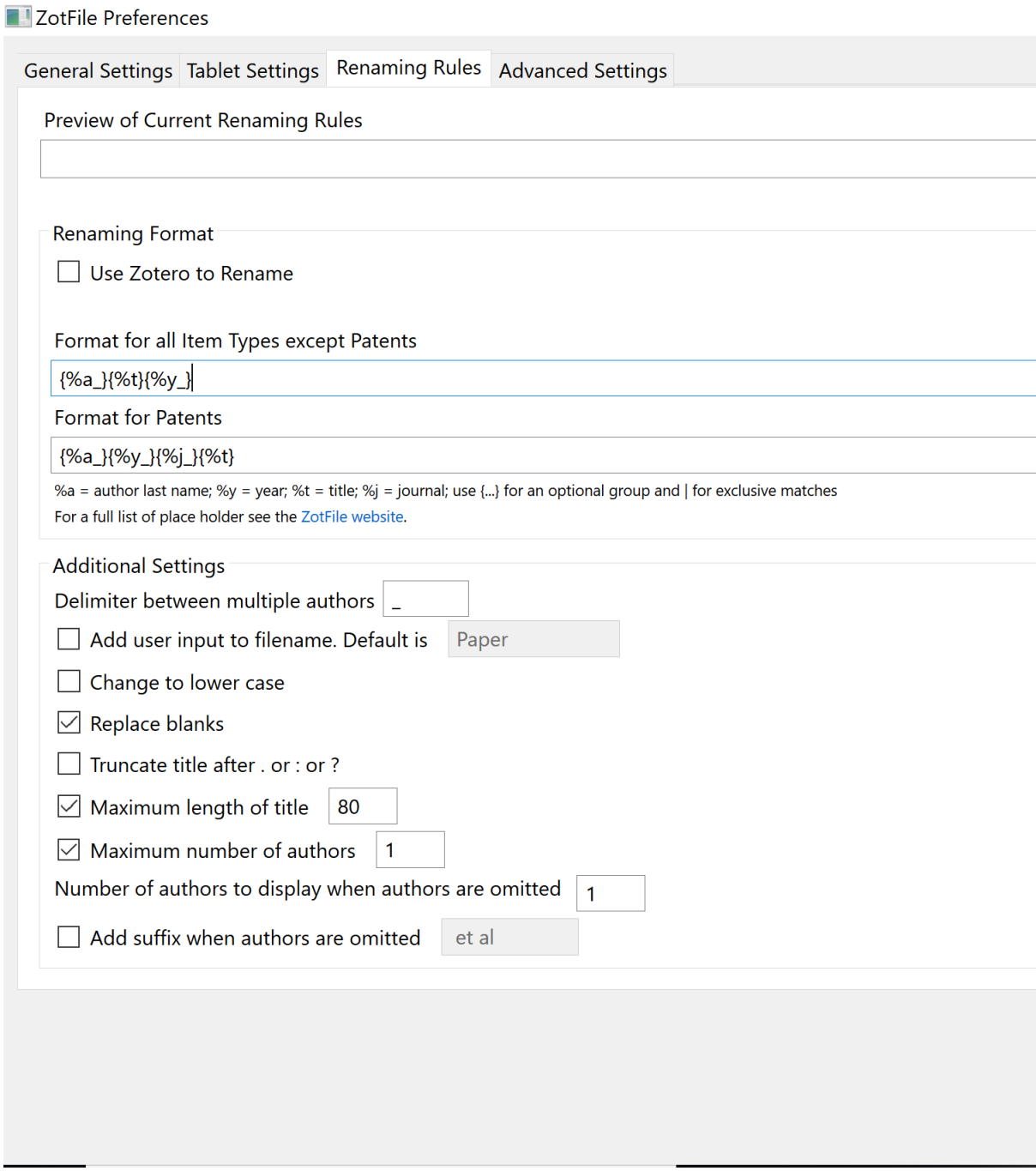


Figure 12.3: ZotFile PDF Storage Preferences

- Enter the account details
- Tick sync automatically
- Untick sync full text (if you choose to save PDFs then syncing full text will quickly consume the 300MB quota)

Search:

- Leave unchanged

Export:

- Leave unchanged

Cite:

- There are several sensible defaults but if there is a new citation style you wish to be able to use in Microsoft Word for example then click "Get additional styles" as there is probably a version that you need already created. You can click the "+" button to add a style from a `.csl` file if you have one already. Finally, if you are desperate for a style that doesn't already exist then you can select a citation style and click Style Editor and edit the raw `.csl` file. The `.csl` file for use in `.rmd` doesn't need configured here but instead within the YAML of your .rmd file
- In the Word Processors subtab (on the main Cite tab), you can install the Microsoft Word add-in to allow Zotero to work in Microsoft Word.

Advanced:

- Change nothing on the General subtab
- In the Files and Folders subtab select the path to directory for attachments
- Change nothing on the Shortcuts subtab
- Change nothing on the Feeds subtab

Better BibTex:

- In this section set the Citation Key format to `[auth:lower:alphanum]_[year:alphanum]_[veryshorttitle:l` (Figure 4). This generates a citation key for each reference in

the format of fairfield_2019_gallstones_scientificreports or harrison_2012_hospital_bmj. It always takes the first author's surname, the year, the first word of the title and the journal abbreviation if known. The clean and alphanum arguments to this field are used to remove unwanted punctuation which can cause citation to fail in LaTeX.

If an author has published an article with the same first word of the title in the same year then the second article appends an "a" to the handle and the third a "b" and so on.

12.4.3 Refresh BibTex Citation Key

If you already have used Zotero without this setup and want to refresh your citation keys to follow the standard pattern then select all references, right click and use "Better BibTex → Refresh BibTeX Key".

12.5 Generating a .bib File

For referencing in a new project, publication or submission it may be helpful to have a dynamic `.bib` file that updates with every new publication added to Zotero and can be accessed from any device through cloud storage.

To set up a `.bib` file, first find the folder that you wish to create the file from (this should be the folder which contains any citations you will use and ideally not the full library to cut down on unnecessary storage and syncing requirements).

Note that the `.bib` file will generate a bibliography from any citations stored directly in the folder when using default settings. This prevents use of subfolders which are particularly helpful for organising citations so it may be helpful to change the setting so that folders also show any citations stored in subfolders. To make this change go to "Edit Preferences" and select the "Advanced" tab and at the bottom of the "General" subtab select "Config Editor". This will bring up a searchable list of configurations (it may show a warning message before this) and search in the search box for "extensions.zotero.recursiveCollections". Set "Value" to TRUE and then when you click a folder you should see all of the citations also stored in subfolders.

Right click the folder which has all of the references (with or without subfolders) and select "Export Collection". A pop-up window will appear at which point select "Keep Updated" and if using RStudio desktop save the file in the directory where you have your `.rmd` project files. If you are working with RStudio server then save the file in a cloud storage location which will then be accessed from the server. A `.bib` file stored in Dropbox can be copied into RStudio server for a given project as per below.

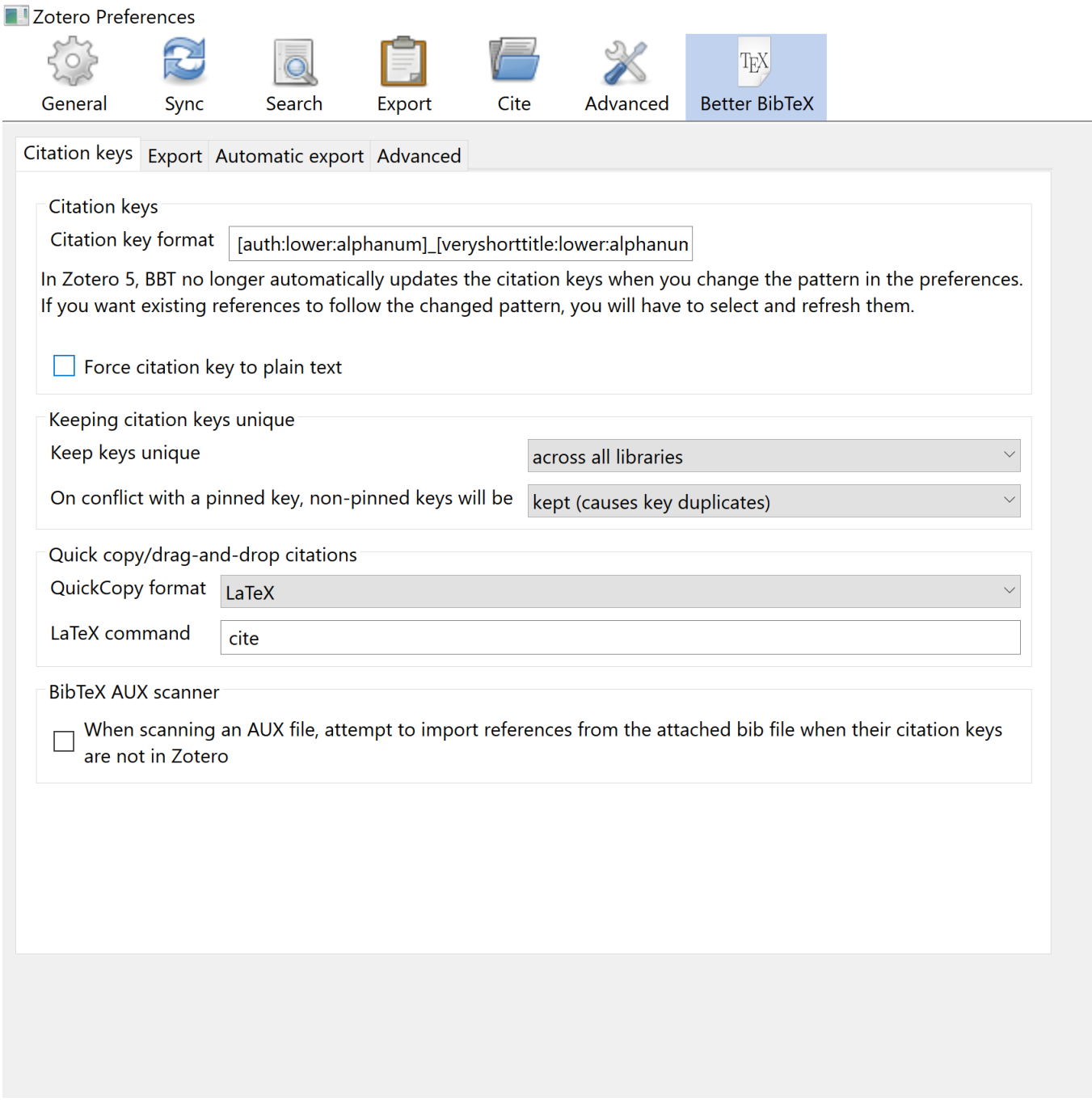


Figure 12.4: ZotFile PDF Storage Preferences

12.6 Linking DropBox and Rstudio Server

Dropbox provides a token to allow communication between different apps. The `rdrop2` package allows this. It may be necessary to create the token on RStudio desktop as creation on the server is buggy but this is perfectly ok.

Caution: The token generated by this process could be used to access your Dropbox from anywhere using RStudio if you do not keep it secure. If somebody were to access an unencrypted token then it would be equivalent to handing out your email and password. Use the `encryptr` package to allow safe storage of this token.

12.6.1 Token Creation

The code will create two files, a token and the `.httr-oauth` file from which a token can also be made. The encryptr package can then encrypt the files using a public / private key pair. It is essential that the password that is set when using genkeys() is remembered otherwise the token cannot then be used. In this case the original token can't be retrieved but could be created again from scratch.

```
library(rdrop2)
library(encryptr)

# Create token
token <- drop_auth()

# Save token
saveRDS(token, "droptoken.rds")

# Encrypt token
genkeys()               # Default file names are id_rsa and id_rsa.pub
encrypt_file("droptoken.rds", "droptoken.rds.encryptr.bin")
encrypt_file(".httr-oauth", ".httr-oauth.encryptr.bin")

# Same details should appear later
drop_acc()

# Remove token from local environment
rm(token)


# Delete the unencrypted files
system("rm droptoken.rds")
system("rm .httr-oauth")
```

The following files will then be needed to upload to the RStudio server:

- `droptoken.rds.encryptr.bin` – or the name provided for the encrypted DropBox token
- `id_rsa` – or the name provided for the private key from the private / public key pair

12.6.2 DropBox Linkage

Now that the encrypted token and necessary (password-protected) private key are available in RStudio server, the following can be saved as a separate script. The script is designed to read in and decrypt the encrypted token (this will require a password and should be done if the .bib file needs updated). Only the drop_download() needs repeated if using the token again during the same session. The token should be cleared at the end of every session for additional security.

```
library(rdrop2)
library(encryptr)

# ******** WARNING ********
# Losing the unencrypted token will give anyone
# complete control of your Dropbox account
# If you are concerned this has happened,
# you can then revoke the rdrop2 app from your
# dropbox account and start over.
# ******** WARNING ********


safely_extract_dropbox_token <- function(encrypted_db_token = NULL,
                                         private_key_file = NULL){
  decrypt_file(encrypted_db_token,
               file_name = "temporary_dropbox_token.rds",
               private_key_path = private_key_file)

  token <<- readRDS("temporary_dropbox_token.rds")

  system("rm temporary_dropbox_token.rds")
}

safely_extract_dropbox_token(encrypted_db_token = "droptoken.rds.encryptr.bin",
                             private_key_file = "id_rsa")

# Then pass the token to each drop_ function
```

```
drop_acc(dtoken = token)

# The path is the Dropbox file location
drop_download(path = "My_Dropbox_Home_Directory/Zotero Library/my.bib",
              local_path = "my.bib",
              dtoken = token,
              overwrite = TRUE)
```

Now that the `.bib` file has been created and is stored as "my.bib" in the local directory, it should update whenever the token is loaded and drop_download() is run. The `.bib` file should be listed in the `.rmd` YAML.

12.7 Citing R Packages

When citing R packages a separate `.bib` file can be created. The following code could be added to an `index.rmd` file in `bookdown` or in a setup chunk in a stand-alone `.rmd`.

```
# automatically create a bib database for R packages
knitr::write_bib(c(
  .packages(), 'encryptr', 'knitr', 'rmarkdown'
), 'packages.bib')
```

Bibliography

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

Xie, Y. (2023). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.35.