

The Surgical Informatics Cookbook

Surgical Informatics, University of Edinburgh

2019-10-01

Contents

1	Introduction	5
1.1	How to contribute	5
1.2	Rules of posting	6
1.3	Indexing	6
2	Snippets	9
2.1	Working with CHIs	9
2.2	Working with dates	12
3	Methods	17
4	Machine learning	19
4.1	Deep learning	19
5	Final Words	21
6	Plotting	23
6.1	GGHighlight Example	23

Chapter 1

Introduction

1.1 How to contribute

(Steps 1. to 5. 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. Open the terminal on argonaut, go to your home folder (just `cd` will default to taking you home). If the prompt says `username@argonaut:~$` that means you’re home - `~` means user’s home.
4. In the terminal, do `git clone git@github.com:SurgicalInformatics/cookbook.git`.
5. Look at the Files tab - a new folder called `cookbook` should have appeared at the bottom. Click on it and open the `cookbook.Rproj` file.
6. Add your thing by editing the appropriate `.Rmd` file - there’s one for each chapter. Use the Build tab to Build your changes into a book.
7. If anyone has pushed since you cloned/last pulled (hopefully they’ve not been working on the exact same chapter): click on More - Clean All. 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.
8. Then Build Book again - this will include the new changes you pulled as well as your changes.
9. Git tab - commit everything, Push quickly before anyone else does or you’ll have to go back to step 7. 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 -A` which is the same thing. Still need to Commit though!

10. Have fun!

1.2 Rules of posting

Rules of how to post here.

1.3 Indexing

1.3.1 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}
```

1.3.2 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)`.

1.3.3 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)
```

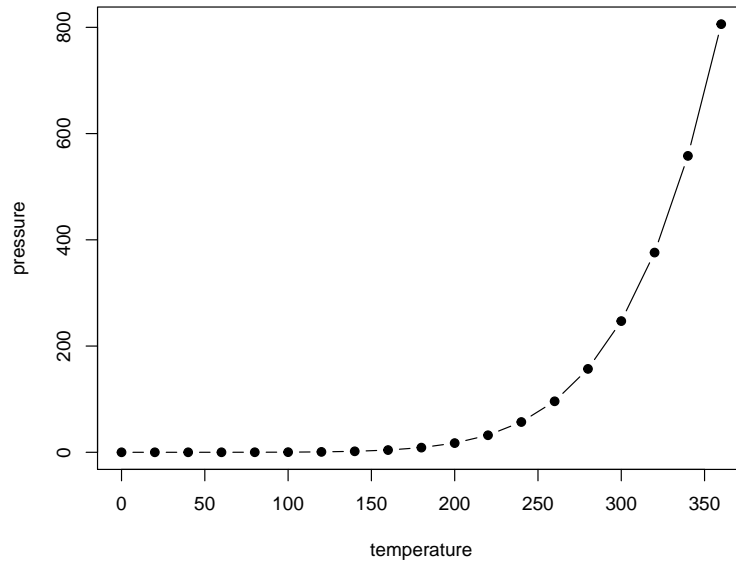


Figure 1.1: Here is a nice figure!

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  
)
```

1.3.4 Citations

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2019) 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 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.

2.1.1 `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
```

2.1.2 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
```

2.1.3 `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] 49 75 70
```

```
# 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    49
## 2 1811431232 1943-11-18 Male    75
## 3 1304496368 1949-04-13 Female  70
```

2.1.4 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 - .} %>% # Subtract 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     49 FALSE
## 2 1811431232 1943-11-18 Male     75  TRUE
## 3 1304496368 1949-04-13 Female   70  TRUE
```

2.2 Working with dates

2.2.1 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
```

2.2.2 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>      <drtm>          <drtm>
```

## 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

2.2.3 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.

2.2.3.1 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(0,0,0,1,0,0,1,1,0,0,0,0,0,0,0,1,1,1,0,0,1,1),
  group = c(rep(0, 8), rep(1, 12)) %>% factor(),
  odate = 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 odate      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
```

2.2.3.2 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
```

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

## Dependent: Surv(time, status)      all      HR (univariable)
## 1                group 0 2 (40.0)      -
## 2                1 3 (60.0) 0.76 (0.10-5.51, p=0.786)
##      HR (multivariable)
## 1                -
## 2 0.76 (0.10-5.51, p=0.786)
```


Chapter 3

Methods

We describe our methods in this chapter.

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

Final Words

We have finished a nice book.

Chapter 6

Plotting

6.1 GGHhighlight Example

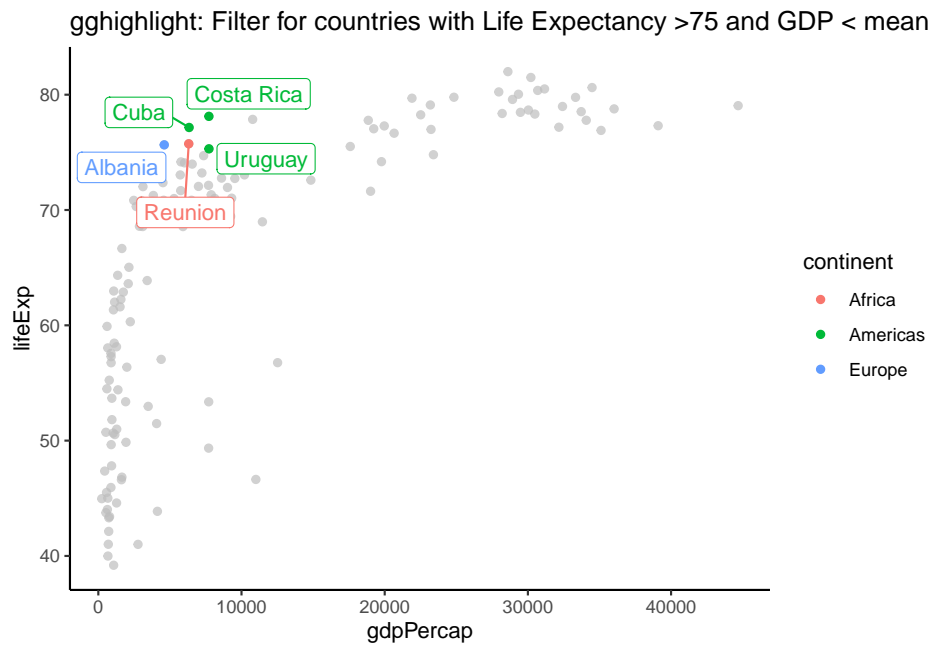
Plotting with gghighlight is pretty awesome allowing you to filter on any variable. It seems that gghighlight overwrites any 'colour' variable you put in the main aes. To get round this and have labels, save as a plot and add geom_label_repel separately.

```
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_d
```



Bibliography

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

Xie, Y. (2019). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.12.