Tutorial 3: Week 6 PDF

Brittany Blankinship

17-05-2022

Contents

Tables **2**

Figures **3**

Images **4**

Inline Code **5**

Mathematical notation 5

Functionalities **5**

HTML 5

PDF 7

Word 7

Themes 7

Reproducibility! **8**

```
library(tidyverse)
library(prettydoc) #for other theme options
library(rmdformats) #for other theme options
library(gapminder) #for our data
library(knitr) #for report generation
library(gt) #for pretty tables

#save the gapminder data into the working environment & create some variables for analysis
mydata <- gapminder %>%
  # round gdpPercap to 0 decimals:
  mutate(gdpPercap = round(gdpPercap)) %>%
  # divide pop by million, round to 1 decimal:
  mutate(pop_millions = (pop/1e6) %>% round(1))

# correlation for inline code example below
LePcor <- cor.test(mydata$lifeExp, mydata$pop)
```

Tables

When making tables, it is a good idea to use the `gt()` function from the `gt` package to produce nice looking tables rather than the usual r output.

Compare this table output:

```
mydata %>%
  select(-pop) %>%
  sample_n(10)
```

```
## # A tibble: 10 x 6
##    country       continent  year lifeExp gdpPercap pop_millions
##    <fct>         <fct>     <int>   <dbl>     <dbl>        <dbl>
##  1 Switzerland   Europe     1957    70.6     17909          5.1
##  2 Ecuador       Americas   1977    61.3      6680          7.3
##  3 Cote d'Ivoire Africa     1962    44.9      1729          3.8
##  4 Ecuador       Americas   2002    74.2      5773         12.9
##  5 Sri Lanka     Asia       1952    57.6      1084          8
##  6 Korea, Rep.   Asia       1972    62.6      3031         33.5
##  7 Pakistan      Asia       1952    43.4       685         41.3
##  8 Bulgaria      Europe     1962    69.5      4254          8
##  9 Ecuador       Americas   2007    75.0      6873         13.8
## 10 Mauritania    Africa     1992    58.3      1361          2.1
```

To this one:

```
mydata %>%
  select(-pop) %>%
  sample_n(10) %>%
  gt()
```

country	continent	year	lifeExp	gdpPercap	pop_millions
Central African Republic	Africa	1972	43.457	1070	1.9
Korea, Dem. Rep.	Asia	1952	50.056	1088	8.9
Singapore	Asia	1992	75.788	24770	3.2
Pakistan	Asia	1992	60.838	1972	120.1
Australia	Oceania	1977	73.490	18334	14.1
Congo, Dem. Rep.	Africa	1952	39.143	781	14.1
Eritrea	Africa	2007	58.040	641	4.9
Hong Kong, China	Asia	1962	67.650	4693	3.3
Madagascar	Africa	1992	52.214	1041	12.2
Bangladesh	Asia	1997	59.412	973	123.3

For further styling options, the `gt` introduction and `gt` cookbook can help!

Below I have grouped the data by `continent`, renamed the variables/columns from what they are called in the dataset, and added stripes to alternating rows.

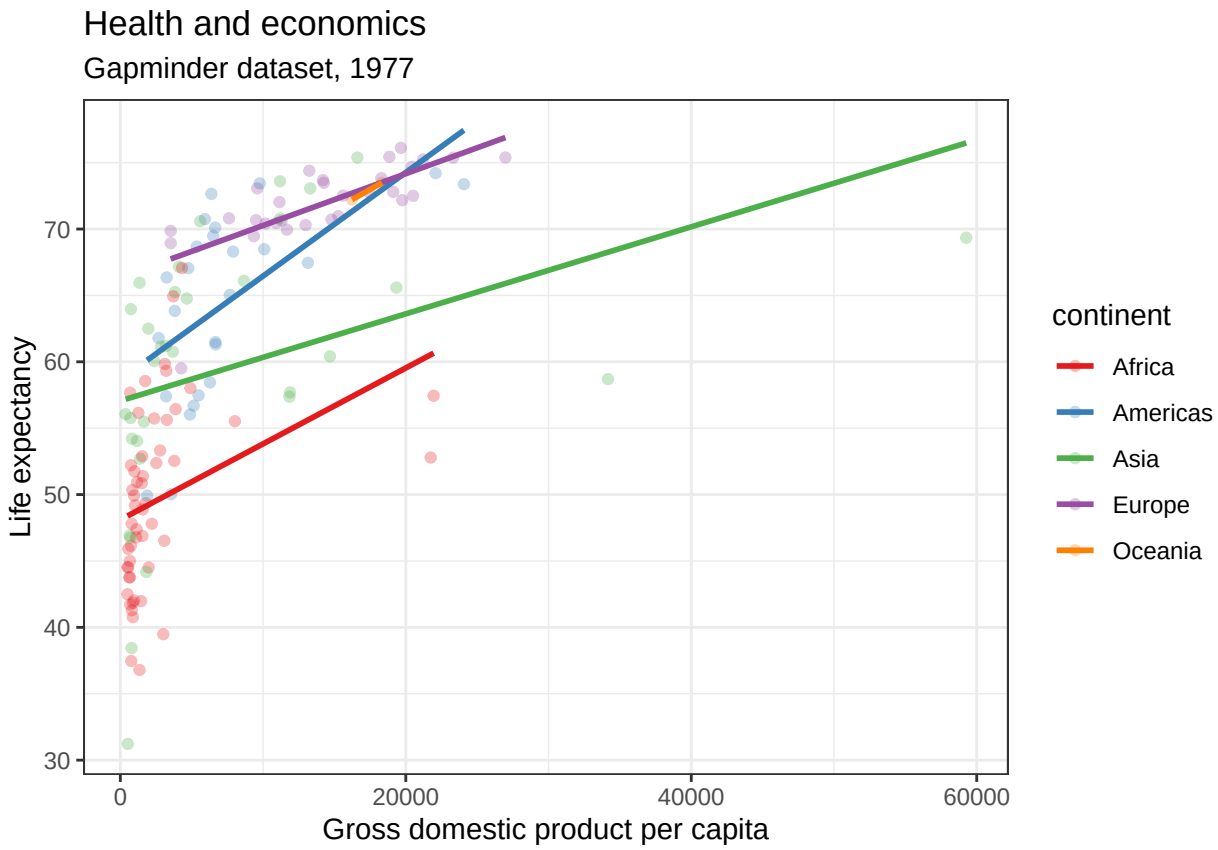
```
mydata %>%
  select(-pop) %>%
  sample_n(10) %>%
  gt(rowname_col = "continent") %>%
  cols_label(country = md("**Country**"),
             continent = md("**Continent**"), year = md("**Year**"),
             lifeExp = md("**Life Expectancy**"),
             gdpPercap = md("**GDP per Capita**"),
             pop_millions = md("**Population (millions)**")) %>%
  opt_row_striping()
```

	Country	Year	Life Expectancy	GDP per Capita	Population (millions)
Americas	Haiti	1977	49.923	1874	4.9
Africa	Angola	1982	39.942	2757	7.0
Europe	Iceland	1987	77.230	26923	0.2
Americas	Mexico	1987	69.498	8688	80.1
Africa	Ghana	2002	58.453	1112	20.6
Africa	Togo	1962	43.922	1068	1.5
Africa	Congo, Rep.	1997	52.962	3484	2.8
Asia	Korea, Dem. Rep.	1977	67.159	4106	16.3
Asia	Korea, Dem. Rep.	1982	69.100	4107	17.6
Asia	Singapore	2002	78.770	36023	4.2

Figures

You can easily integrate figures into your report as well using a code chunk dedicated to the figure of interest.

```
mydata %>%
  filter(year == 1977) %>%
  ggplot(aes(y = lifeExp, x = gdpPercap, colour = continent)) +
  geom_point(alpha = 0.3) +
  theme_bw() +
  geom_smooth(method = "lm", se = FALSE) +
  scale_colour_brewer(palette = "Set1") +
  labs(x = "Gross domestic product per capita",
       y = "Life expectancy",
       title = "Health and economics",
       subtitle = "Gapminder dataset, 1977")
```



Images

You can easily insert images into your R Markdown file either from a webpage or from a file in the same directory:

1. From a webpage

To get the link to a picture from a webpage go to your internet browser of choice, Images, and right click on the image and select "Copy Image Link".

Note: When knitting to pdf, you need to add another line of code in the YAML for LaTeX `graphicx` to pull an image from the internet and then save them locally in the same directory as the RMD file.

```
`output:
  pdf_document:
    pandoc_args: ["--extract-media", "."]`
```

2. From a file

Note: to knit this document on your own device, you will need to change the file path above in number 2. "figures/Map of Health Boards.png" means go to the "figures" folder and find the file called "Map of Health Boards.png". If you do not have a figures folder within your working environment, you will get an error when trying to knit the document.

Figure 1: Beach Chairs

Inline Code

You can also use r code directly in text. For example:

> There are 1704 observations in the data set.

This can be an incredible time saver if you are creating a report where the data itself might still be coming in. Rather than needing to manually update descriptive information such as how many observations there are in the data set, let R do it for you!

I also will often use this when reporting in text results of analyses. For example:

> The correlation between life expectancy and population is $r(1702) = 0.065$, $p = 0.007$

Mathematical notation

You will notice above that I used something else in combination with the inline r code - mathematical notation! This is based on LaTeX. In side a text chunk, you can use mathematical notation if you surround it by dollar signs `$` for "inline mathematics" and `$$` for "displayed equations". This post by R Pruim is my go-to resource for mathematical notation in R Markdown.

Functionalities

HTML

In general, I find that when knitting to an `html_document` there are more interesting functionalities. Indeed, Markdown was originally designed for HTML output so the HTML format has the richest features.

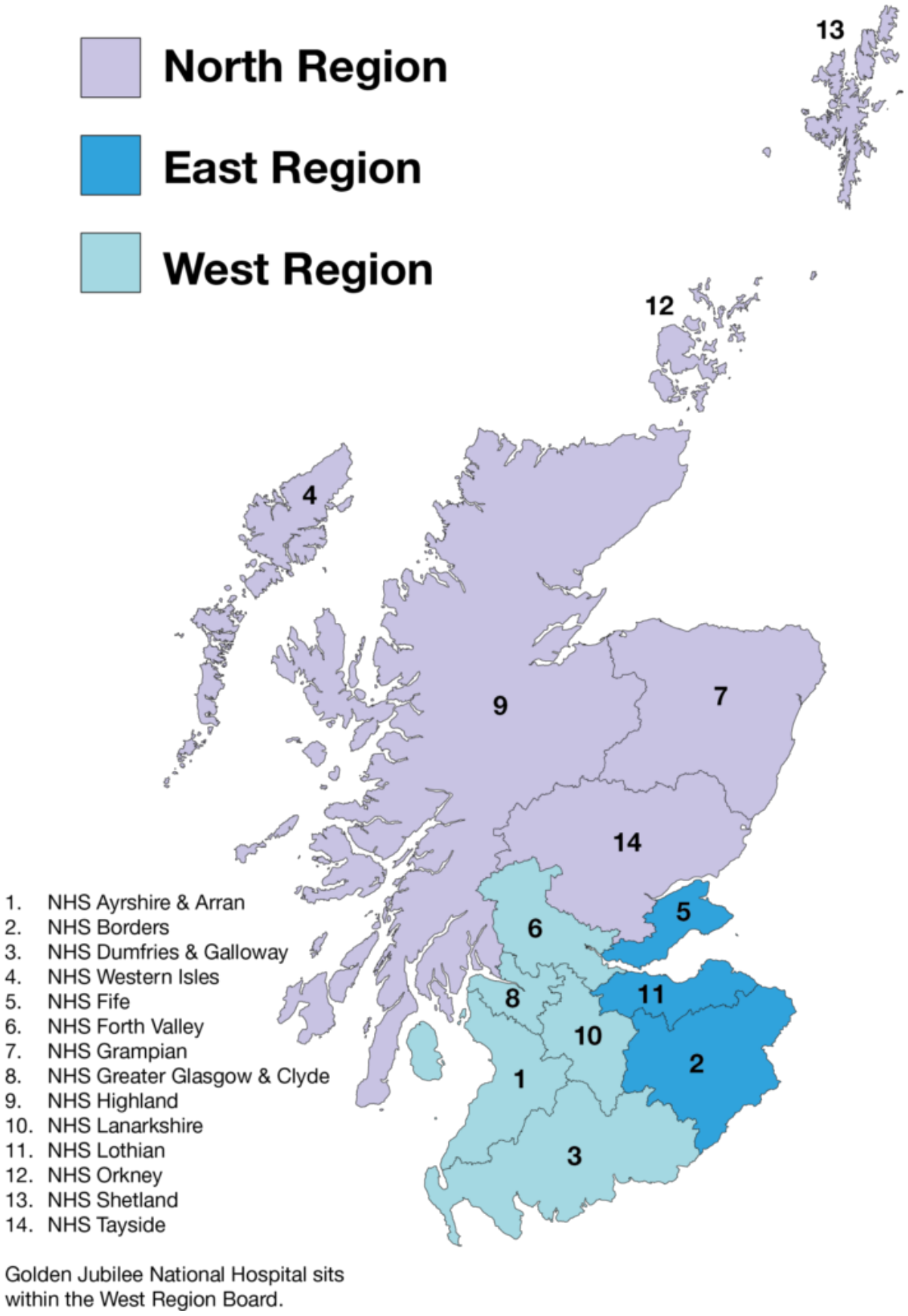


Figure 2: NHS Scotland HBs

- floating table of contents
- code download output option
- code folding (hide or show)
- tabbed sections

You can have a look at some of the other options when knitting to HTML in the HTML document chapter in R Markdown: The Definitive Guide

Chunk names are not necessarily required to knit, but are good practice and help to support more advanced knitted approaches. Chunk labels should be

- unique
- do not use spaces, rather use dashes (-)
- use alphanumeric characters (a-z, A-Z and 0-9), because they are not special characters and will surely work for all output formats
- spaces and underscores in particular, may cause trouble in certain packages, such as `bookdown`.

We do not cover the `bookdown` package as part of this course, but it extends the functionality of rmarkdown to allow for figures and tables to be easily cross-referenced within your text (among other things).

PDF

PDFs in R Markdown also have a variety of features, but not nearly as many as HTML documents. To find out more, check out the PDF document chapter in R Markdown: The Definitive Guide

Word

The most notable feature of Word documents in R Markdown is the ability to create a "style reference document". To find out more, including a short video on how to create and use a reference document, check out the Word document chapter in R Markdown: The Defintive Guide

Themes

In general, there are a variety of theme options available to use with R Markdown HTML documents

`prettydoc`

The `prettydoc` package includes a variety of other theme options when knitting to HTML: https://prettydoc.statr.me/index.html

Once you have installed the package, you can open a prettydoc formatted document from "From Template" tab when choosing to create a new R Markdown file.

To use a `prettydoc` theme not from a template, you need to edit the YAML accordingly:

```
`output:
    prettydoc::html_pretty:
    theme: cayman`
```

Note: When using `html_pretty` engine for themes, `code_folding`, `code_download`, and `toc_float` are not applicable.

`rmdformats`

The `rmdformats` package includes a variety of other theme options when knitting to HTML: https://github.com/juba/rmdformats. Some themes allow for things like a dynamic table of contents, but not all of them. See the "Features matrix" table on the above webpage for more info.

Similar to above, once you have installed the package, you can open a rmdformts formatted document from "From Template" tab when choosing to create a new R Markdown file.

To use a `rmdformats` theme not from template, you need to edit the YAML accordingly:

```
`output:
    rmdformats::robobook`
```

Reproducibility!

In practice, depending on your audience, you will need to decide to show your code or not. It is unlikely that you will want to show the code used to produce your analysis, tables, or figures to an audience unfamiliar with R and would therefore set `echo = FALSE` in the set-up chunk. For this course, though, and in particular for the programming assignment, you will need to set `echo=TRUE` so that we can see your code and the product of that code.

While it can take up space, it is good practice to finish a document calling the `sessionInfo` function, which lists all of the packages you used, their versions, and more.

```
## R version 4.1.3 (2022-03-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] gt_0.4.0         knitr_1.38       gapminder_0.3.0  rmdformats_1.0.3
##  [5] prettydoc_0.4.1  forcats_0.5.1    stringr_1.4.0    dplyr_1.0.8
##  [9] purrr_0.3.4      readr_2.1.2      tidyr_1.2.0      tibble_3.1.6
## [13] ggplot2_3.3.5    tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] lattice_0.20-45    lubridate_1.8.0    assertthat_0.2.1   digest_0.6.29
##  [5] utf8_1.2.2         R6_2.5.1           cellranger_1.1.0   backports_1.4.1
##  [9] reprex_2.0.1       evaluate_0.15      highr_0.9          httr_1.4.2
## [13] pillar_1.7.0       rlang_1.0.2        readxl_1.4.0       rstudioapi_0.13
## [17] Matrix_1.4-1       checkmate_2.0.0    rmarkdown_2.13     labeling_0.4.2
## [21] splines_4.1.3      munsell_0.5.0      broom_0.7.12       compiler_4.1.3
## [25] modelr_0.1.8       xfun_0.30          pkgconfig_2.0.3    mgcv_1.8-40
## [29] htmltools_0.5.2    tidyselect_1.1.2   bookdown_0.26      fansi_1.0.3
```

```
## [33] crayon_1.5.1       tzdb_0.3.0         dbplyr_2.1.1       withr_2.5.0
## [37] commonmark_1.8.0   grid_4.1.3         nlme_3.1-157       jsonlite_1.8.0
## [41] gtable_0.3.0       lifecycle_1.0.1    DBI_1.1.2          magrittr_2.0.3
## [45] scales_1.1.1       cli_3.2.0          stringi_1.7.6      farver_2.1.0
## [49] fs_1.5.2           xml2_1.3.3         ellipsis_0.3.2     generics_0.1.2
## [53] vctrs_0.4.0        RColorBrewer_1.1-3 tools_4.1.3        glue_1.6.2
## [57] hms_1.1.1          fastmap_1.1.0      yaml_2.3.5         colorspace_2.0-3
## [61] rvest_1.0.2        haven_2.4.3
```