

R for Health Data Science

Week 05: Functional Reporting in R

Sam Stewart

2021-03-19

```
dat = read.csv("data/framinghamFirst.csv",header=TRUE,
              na.strings=".",stringsAsFactors=FALSE)
dat$BMIGroups = cut(dat$BMI,breaks=c(0,18.5,25,30,Inf),
                   labels=c("Underweight","Normal","Overweight","Obese"))
dat$SEX = factor(dat$SEX,levels=1:2,labels=c("Male","Female"))
dat$DIABETES = factor(dat$DIABETES,levels=0:1,labels=c("No Diabetes","Diabetes"))
dat$HYPERTEN = factor(dat$HYPERTEN,levels=0:1,labels=c("Normotensive","Hypertensive"))
```

1 Introduction

This week we'll be focusing on reporting in R. I find one of the major advantages of R is that it separates the analysis from the reporting - when you run a test (t-test, regression model, ...) it doesn't naturally produce anything. This allows the analyst to produce their own, functional report, rather than using the output from a log file.

The challenge with this is that you *have* to do all the reporting yourself - if you have a STATA program with your analysis you can run it and the output file will have all the info you need to produce a report, while in R you need an extra step to produce that material.

The major advantage that R has now is **RMarkdown** - a system to integrate reporting with analysis in a single file. This will allow you to integrate reporting with analysis seamlessly, so that when data is updated the reporting can be updated automatically. I would direct you to a couple of additional resources for RMarkdown:

- Official Documentation This is the official documentation - it's very helpful, but a bit dry and dense. You'll end up there a lot once you get the hang of the basics.
- RMarkdown Guide: Is an online book (available for print order as well) that will guide you through the R Markdown process
- HBC Training: Harvard Biostats has a good intro-to-RMarkdown series here - there's no biostats in it, but it's a great start (I'll borrow from it today)
- RMarkdown cheatsheet: There are so many copies of this same PDF on the web - if you like printing things out I would suggest printing this and keeping it on hand, otherwise you'll just Google "RMarkdown Cheat Sheet" a lot (that's my solution)

Finally, if you want to see some examples, the Github page has markdown files for the week 2-4 lectures, and will continue to have them for the future lectures.

2 RMarkdown Basics

RMarkdown is an integration of R+Markdown, a simple typesetting program popular on the web (Github, Reddit and Stackexchange all support markdown in their responses). The goal of Markdown was to create

an “easy to write, easy to read” typesetting language that could easily be converted to HTML (or other languages) for typesetting.

I can’t typeset the examples as well as the cheatsheet, so let’s quickly go through that to see the basics of formatting.

2.1 YAML Header

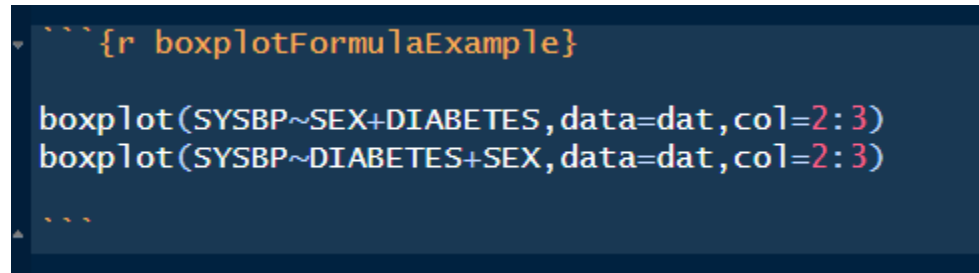
The header of the RMarkdown document is key to getting it to produce the right material. You can specify the title, subtitle, author and date, and then control the output document with the `output:` command. The output types you’ll use most often are `html_document` and `pdf_document`. As you’ll see in the sample slides, the output types themselves can have arguments: here are the two headers I use for PDF and HTML output:

```
title: "R for Health Data Science"
subtitle: "Week 05: Functional Reporting in R"
author: "Sam Stewart"
date: "2021-03-19"
output:
  html_document:
    number_sections: true
    toc: true
    toc_depth: 3
    toc_float:
      collapsed: false
```

```
title: "R for Health Data Science"
subtitle: "Week 05: Functional Reporting in R"
author: "Sam Stewart"
date: "2021-03-19"
output:
  pdf_document:
    number_sections: true
```

3 Code Chunks

Using RMarkdown allows us to embed chunks of R code within our Markdown report: starting a line with three back-ticks (top-left button on the keyboard, “```”) tells R that we’re entering a code chunk. Adding a `{r}` tells it that it’s an R code chunk, and will evaluate using R.



In the example we named the chunk `boxplotFormulaExample` - this isn’t a requirement, but it’ll make debugging and navigating easier if you name all your code chunks.

3.1 Options

Code chunks can accept a number of options that control various types of output - see the complete documentation for the full list, but these ones from the cheat sheet are the most commonly used:

Option	Default	Description
eval	TRUE	Whether to evaluate the code and include its results
echo	TRUE	Whether to display code along with its results
warning	TRUE	Whether to display warnings
error	FALSE	Whether to display errors
message	TRUE	Whether to display messages
tidy	FALSE	Whether to reformat code in a tidy way when displaying it
results	markup	markup, asis, hold, or hide
cache	FALSE	Whether to cache results for future renders
comment	##	Comment character to preface results with
fig.width	7	Width in inches for plots created in chunk
fig.height	7	Height in inches for plots created in chunk

You can also set the default block options at the start of the report - here's the default I have setup for our lectures, and a more complex one from the HBC series

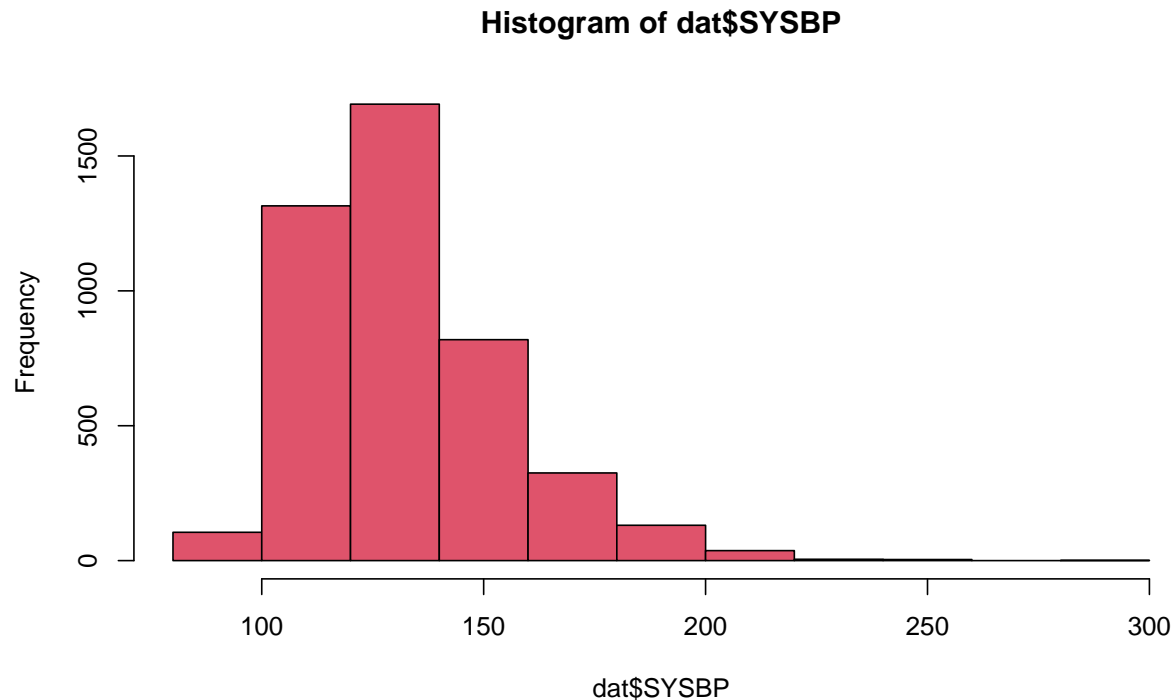
```
knitr::opts_chunk$set(tidy=TRUE, echo = TRUE, message=FALSE, warning=FALSE, fig.width=8, fig.height=5)
```

```
opts_chunk$set(
  autodep = TRUE,
  cache = TRUE,
  cache.lazy = TRUE,
  dev = c("png", "pdf", "svg"),
  error = TRUE,
  fig.height = 6,
  fig.retina = 2,
  fig.width = 6,
  highlight = TRUE,
  message = FALSE,
  prompt = TRUE,
  tidy = TRUE,
  warning = FALSE)
```

3.2 Figures and Tables

Figures produced by R are embedded directly within the HTML document - there are no external files created - if you open the source code on the HTML versions of our lectures you can see that the figure is represented in HEX directly in the file.

```
hist(dat$SYSBP,col=2)
```



```
tab = table(dat$SEX, dat$BMIGroup)
tab
```

```
##
##      Underweight Normal Overweight Obese
## Male           12    703        992   232
## Female          45   1233        856   342
```

```
kable(tab)
```

	Underweight	Normal	Overweight	Obese
Male	12	703	992	232
Female	45	1233	856	342

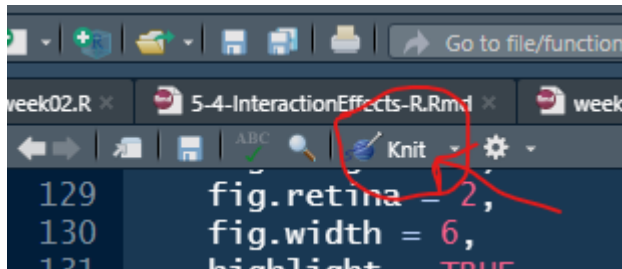
```
kable_styling(kable(tab))
```

	Underweight	Normal	Overweight	Obese
Male	12	703	992	232
Female	45	1233	856	342

Printing a table just produces the R-system output. the command `kable()` converts the table to markdown, and `kable_styling()` adds some production value. We'll come back to the `kable_styling()` and the `kableExtra` library next week when we talk about `dplyr` and pipes - you can make some fairly impressive tables directly from R code, see the `kableExtra` vignette for some examples.

4 Running RMarkdown Files

To run the markdown files you need to 'knit' them - this is done with either the 'knit' button at the top of the screen, or by pressing Ctrl+Shift+k.



Note that knitting a document runs all the code - this can be a bit time consuming, particularly if you have some slow data-processing code. To address this use the `cache=TRUE` option in slow blocks - if the code in the block didn't change from the last execution then it won't re-run it.

5 Using RMarkdown for Reporting

At the end of last class we asked 5 biostats questions - we'll now explore how to report those. I've created a second document called `week05-breakout.Rmd` that has the reporting all setup, we'll start it together, then split out to groups to complete the following tasks:

1. Improve the reporting of the t-test comparing glucose levels to sex
2. Improve the reporting of the regression models predicting glucose with sex and smoking
3. Improve the reporting of the Effect Measures (RR, OR, RD) for the effect of smoking on hypertension