KIND Knowledge Information Data Learning Network

NHS Education for Scotland

Dynamic report demo

2022-06-09 14:42:30

Contents

Introduction 1

Who is this demo for? 2

How does it work 2

What you need 2

How does all this work? 2

Is it worth it for me? 3

Starting an Rmarkdown document 3

Getting data into R 4

Getting text to update to reflect data 5

Doing data processing 6

Graphs 7

Introduction

This is an example of a dynamic report, written in R and Rmarkdown. It gives some example ways of automating common data-driven tasks. The idea is that this report can be a starting point for automating report writing processes. The idea is to replace (or speed up) manual report writing using Word and Excel). The demo gives some basic introduction to the workflow. It then walks through a series of example tasks based on a toy data set. The demo also covers ways of producing the report in a variety of formats including pdf, Word document, and html webpage.

Who is this demo for?

Spend lots of time updating routine reports, such as:

- monthly figures (admissions, cases, . . .)
- continuing data monitoring (users, learners. . .)

It isn't intended to be a full introduction to working in R/Rmarkdown from scratch.

Instead as a guide and support for those who are considering learning a little bit of R as a way out of writing manual reports.

How does it work

Explain here a bit about R and Rmarkdown

You might like to have a quick look at some of the excellent introductory R resources **add some resources**

Working through the report will take a couple of hours.

What you need

- working R and Rstudio installation **or**
- access to a RStudio Cloud account

Easy to set-up and free for small-scale work. Likely not suitable for production work in health and care owing to information governance concerns, but by far the easiest way to get going from scratch if you've never worked with R before.

- the source code for this report - GitHub link
- the demo data file demo.csv
- some experience of building reports manually
- a good idea of what you'd like your report to do
- time and space to complete the demo, and think about how it might be useful in your own practice

My suggestion would be not to try and use this demo to change your way of writing reports under pressure. There's quite a lot to think about here, and you might need to spend a good bit of time working out how to adapt this demonstration to fit your report. Think of this as the start of a journey, rather than a destination.

How does all this work?

This demonstration will take you, step by step, through building a simple report in R and Rmarkdown. The big difference here from the usual way of building reports by hand is that you will combined the analysis, visualisation, and production stages of your report into a single step. As this demo will show, you can do all of this using R and Rmarkdown. You'll load the data, clean and analyse the data, and produce outputs (text, tables, graphs) straight into your report.

Is it worth it for me?

- Much harder the first time, and slower
- So if you have regular reports, especially complicated ones with lots of graphs, deffo
- Local IG concerns

Starting an Rmarkdown document

There's a bit of preamble at the head of Rmarkdown documents that controls how they are built. For now, I'll suggest that you skip trying to figure it all out, and just use someone else's. The easiest option is to start a new Rmarkdown document direct from the RStudio menu `File >> New file >> R Markdown...`. Alternatively, you can copy the header from this document, and paste it at the top of your document. Everything between the pair of three dashes: `---` is concerned with setting up this Rmarkdown document:

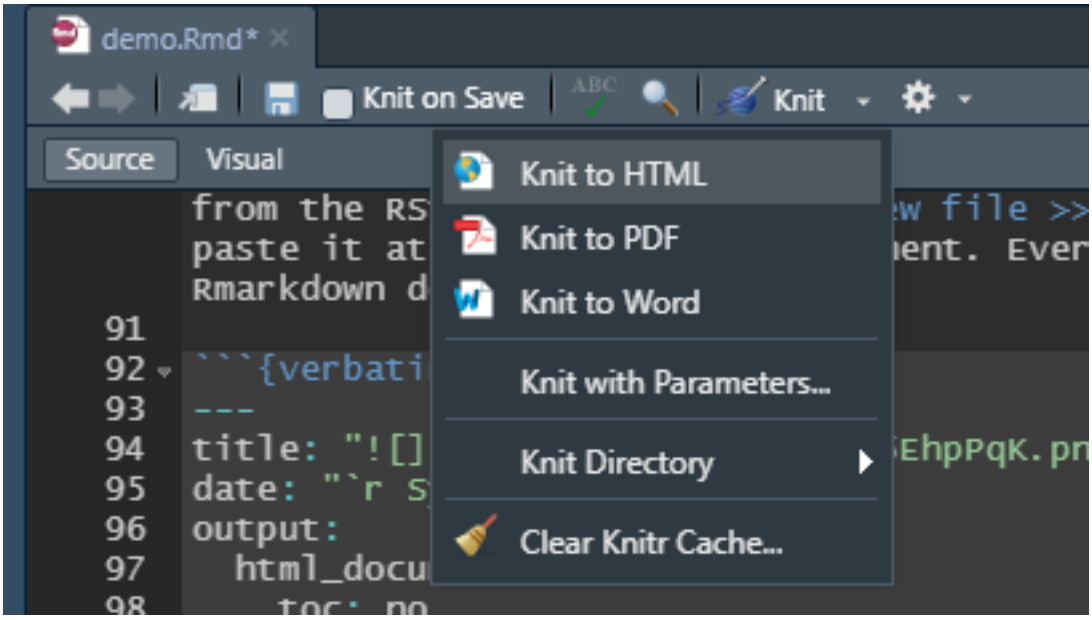
```
---
title: "![](https://i.imgur.com/5EhpPqK.png) Dynamic report demo"
date: "`r Sys.time()`"
output:
  html_document:
    toc: no
    toc_depth: 2
    number_sections: no
    toc_float:
      collapsed: no
  word_document:
    toc: yes
    toc_depth: '2'
  pdf_document:
    toc: yes
    toc_depth: '2'
---
```

This gives a mini-header to your document, which is tweakable. If you look at the very top of this page, you can see there's a KIND Learning Network header image, followed by the document title. You can easily change both of those by editing the URL and the title text in the following line:

```
title: "![](https://i.imgur.com/5EhpPqK.png) Dynamic report demo"
```

More info on YAML editing in chapter 2 of the excellent R Markdown Cookbook.

There are also options for several different output formats (html, word, and pdf). In RStudio, you can select which output format you'd like your report to be rendered as:



Try these out now - this demo report should knit to each of the three formats without any problems. I'd recommend sticking with html while you're working on a report, though, just because it tends to render most quickly, and plays nicely with other aspects of the workflow while you're writing.

Once you've got the header in place and tweaked to your liking, we're ready to move on to something more useful: getting data into your report.

Getting data into R

This is easy to do in practice, but is a bit of a conceptual leap if you're used to collecting and receiving data in Excel. Instead of opening a data file directly with Excel so that you can inspect and analyse it, we instead load all the data in that file into R before we can use it.

Because there are lots of different types of data out there, there are lots of ways of loading data into R.

Let's look at the first R chunk of our doc:

```
library(tidyverse)
library(NHSRplotthedots)
library(lubridate)
demo <- read_csv("demo.csv") %>% mutate(date = dmy(date))
```

Code chunks in Rmarkdown start and end with a triple backtick:

```
```{r setup, echo=TRUE, eval=TRUE,warning = FALSE, message=FALSE}
library(tidyverse)
library(NHSRplotthedots)
library(lubridate)
demo <- read_csv("demo.csv")
```
```

There are then a few options that you can set for the code chunk. Here:

- r: telling Rmarkdown to interpret this code as R. There are other languages that you can use in Rmarkdown, which is one of the strengths of building reports in this way.
- setup: the chunk label. This can be anything you like, but no duplicates are allowed. Very useful for navigation in more complicated reports
- echo=TRUE: whether to show this code in the report. If this is set to FALSE, you'll just see the output of the code (more on this below)
- eval=TRUE: whether or not to run the code. If eval=FALSE, the code won't do anything other than appear in the report (useful if you're explaining how something works - like this report!)
- warning = FALSE: whether to show warnings in the report if something goes wrong with your code
- message=FALSE: whether to show information messages about how your code is running

Lots more info in the Rmarkdown Cookbook

That chunk of R code is doing two different things: + the three lines starting `library(...)` load three packages that help you do your analysis. More info

you can also use `knitr::opts_chunk$set`, which is a really powerful way of managing chunk options - but that's a bit beyond the scope of this demo.

Getting text to update to reflect data

Let's have a look at the what's in demo.csv. We load this into the report in the setup chunk, using

```
demo <- read_csv("demo.csv") %>% mutate(date = dmy(date))
```

That gives us a tibble as follows:

date	count1	count2	count3
2022-05-01	11	93	82
2022-05-02	15	114	99
2022-05-03	18	28	10
2022-05-04	19	107	88
2022-05-05	21	140	119
2022-05-06	170	217	47

(there are another 60 or so rows here that I've cut out to save space).

This data will change as the data in the csv file changes. I'll encourage you to fiddle about with this, but if you're used to copy and pasting data tables about to update, this will be a great help.

We can also produce summary text inline. For example, we can count up all of the count1 column (a total of 6643). Or find out that the maximum value in count3 is 141. Or we can say that on the first day of this range (19113), the value of count2 was 93. Or that the mean (average) of count3 was 70.5.

I'll say it again: **change the data, and all this stuff will update automatically**. No need to copy and paste at many locations. For example, you might like to include a date in your report describing which month the data is about. You can do this easily: the latest date in this data comes from July. Change the data, and this report will update - go on, try it!

Doing data processing

We can do useful stuff like add the columns together:

date	count1	count2	count3	total
2022-05-01	11	93	82	186
2022-05-02	15	114	99	228
2022-05-03	18	28	10	56
2022-05-04	19	107	88	214
2022-05-05	21	140	119	280
2022-05-06	170	217	47	434

(again, just showing the first few rows of data).

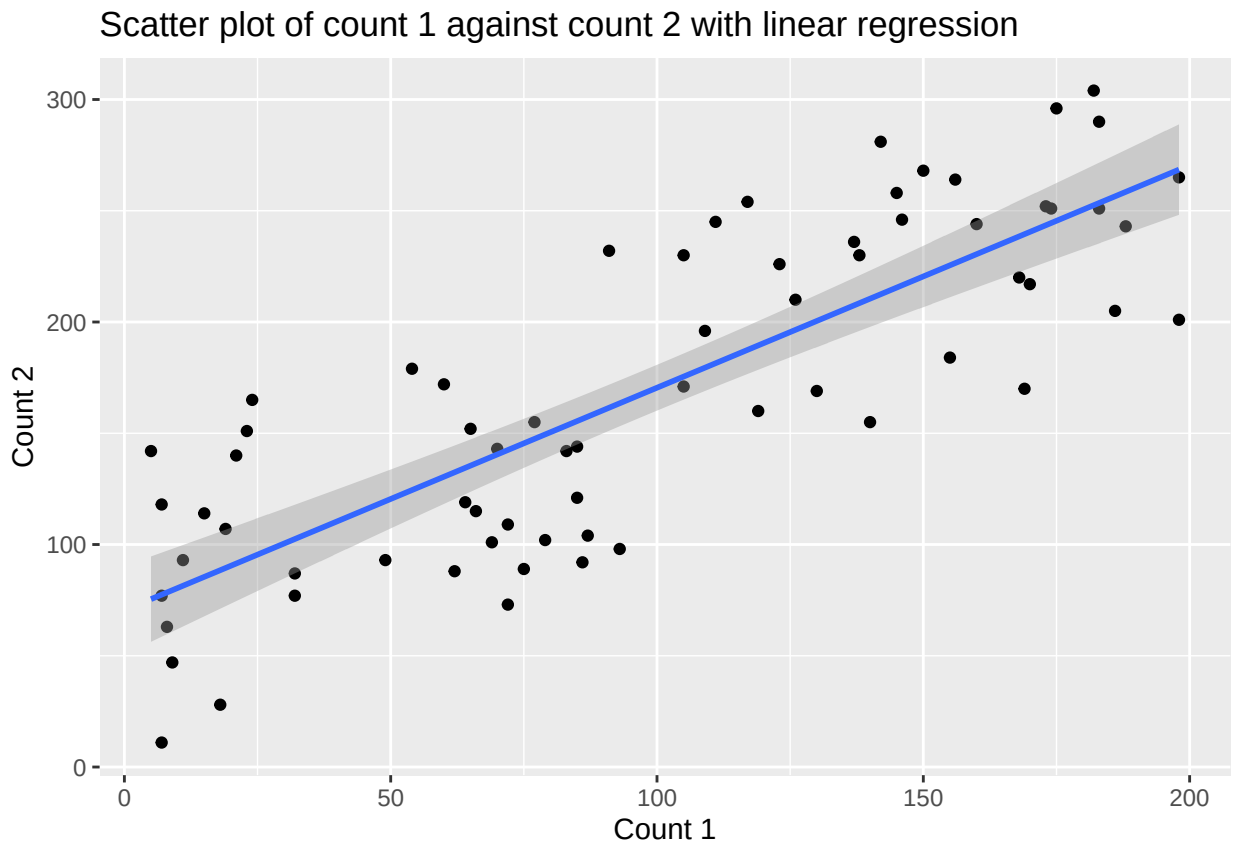
We can also summarise tables of data:

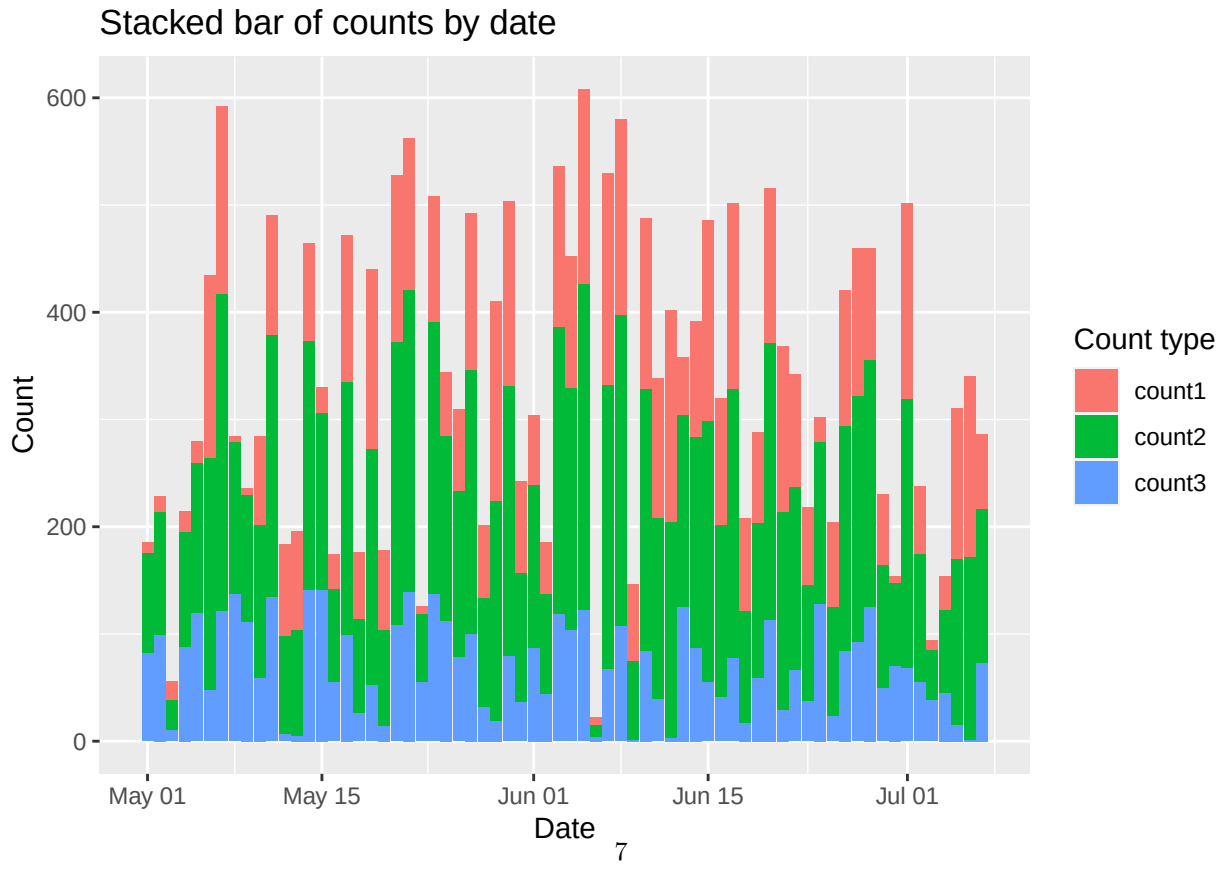
count1	count2	count3	total
98	168	70	336

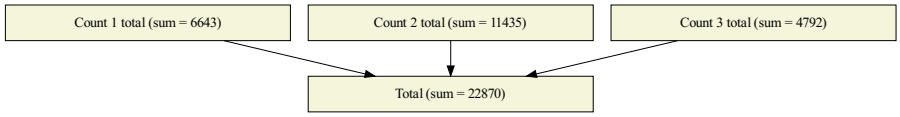
or re-name columns in our tables:

Mean of count 1	Mean of count 2	Mean of count 3	Mean of total
98	168	70	336

Graphs


Scatter plot of count 1 against count 2 with linear regression
Count 2
300
200
100
0
0
50
100
150
200
Count 1


Stacked bar of counts by date
Count
600
400
200
0
May 01
May 15
Jun 01
Jun 15
Jul 01
Date
Count type
count1
count2
count3


Count 1 total (sum = 6643)
Count 2 total (sum = 11435)
Count 3 total (sum = 4792)
Total (sum = 22870)