PDF QC Reports

AACC University 2022: Doing More with R

Dustin R. Bunch

29 Jun 2022

The Form of a Report

In my mind a report has four functional areas from top to bottom they are.

1. A header with information, logo, or both.
2. A body containing the data or analysis that is being conveyed.
3. A signature/date line for the appropriate sign-off.
4. A footer that is able to hold institutional information.

What are the steps to create a report with rstudio and rmarkdown?

The first two areas to tackle are the header (1) and footer (4).

To do this we look toward the YAML.

The Header and Footer

What is the YAML?

From https://yaml.org/, YAML stand for "YAML Ain't Markup Language" and "YAML is a human friendly data serialization standard for all programming languages".

Well, based on this I still didn't really know what YAML was.

So I looked toward https://bookdown.org/yihui/rmarkdown/, where YAML is described as the metadata for controlling certain aspects of your document.

You may recognize YAML as the extra stuff at the top of a rmarkdown file as an example.

```
---
title: "Basic_QC_Report"
author: "Dustin R. Bunch"
date: "12/9/2020"
output: pdf_document
---
```

The YAML metadata can consist of LaTeX and this is where we get the greater control over how our report will look.

What is LaTeX?

From https://www.latex-project.org/, LaTeX is a document preparation system for high-quality typesetting of technical and scientific documentation. The most important information for our use is there are packages

available for LaTeX similar to R.R has CRAN for packages https://cran.r-project.org/ while LaTeX has CTAN https://ctan.org/.

The information for the package we will use for our report is called *fancyhdr* more information can be found at https://ctan.org/pkg/fancyhdr?lang=en

Let us get started on building our report.

Header

First we add two new sections to the YAML.

```
geometry: margin=0.75in
header-includes:
```

The YAML metadata area has one major quirk which is spacing matters. My suggestion is to let rstudio set the spacing and do not change it. It will cause errors when attempting to knit which can take a while to pin down if you are not thinking about it.

geometry: will set the margins for your pdf. header-includes: will have our header and footer information

The next thing is to add the LaTeX package to the header-includes:. We perform this by adding `- \usepackage{fancyhdr}`

```
header-includes:
- \usepackage{fancyhdr}
```

Next we add the header information using `- \fancyhead[]{}` which has options availabe in the [] that include placement information L=Left, R=Right, and C=Center and O=Odd and E=Even pages. Inside the {} is the information we are presenting on the report. In this case we want to put our branding logo and department information.

```
- \pagestyle{fancy}
- \fancyhead[LO,LE]{\includegraphics[width=4in,height=1.5in]{Example_header.jpg}\\}
- \fancyhead[RO,RE]{Department of Pathology \\ and Laboratory Medicine,\\ Biochemical
  Genectis Section}
```

Footer

The footer is similar to the header the only thing that is changed is head to foot `- \fancyfoot[]{}`.

Add some data for the footer and we end up with YAML metadata that looks like the following example.

```
author: "Dustin R. Bunch"
date: "November 23rd, 2020"
geometry: margin=0.75in
header-includes:
- \usepackage{fancyhdr}
- \pagestyle{fancy}
- \fancyhead[LO,LE]{\includegraphics[width=4in,height=1.5in]{Example_header.jpg}\\}
- \fancyhead[RO,RE]{Department of Pathology \\ and Laboratory Medicine,\\ Biochemical
  Genectis Section}
- \fancyfoot[LO,LE]{Institutional Disclaimer:}
```

The Signature and Date

The signature and date were add through a png image.

How to add an image in rmarkdown?

Insert the following into a non-code section of the rmd document `![]()\`. The ! indicates it is an image. The [] contain the metadata for the image. The () is the location of the image on your system. Finally, the \ prevents the image from creating a "Fig #" caption on every image.

```
![Signature Line](2020_AACC_2.JPG)\
```

Additionally, we add some break using the following code to create space between the rest of the report and the signature line.

The Body

There are 2 parts to a rmarkdwon report body. 1. Text 2. Code

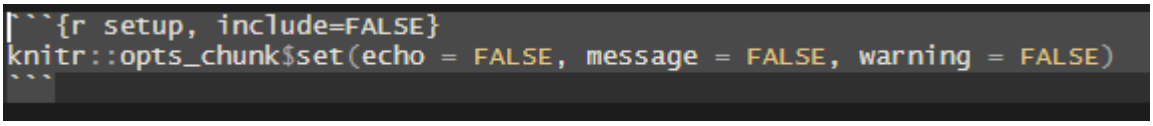
The code can be split into 3 additional parts. 1. Structure Code 2. Tables 3. Figures

The Text

The text can be placed anywhere outside of code blocks.

Remember code blocks look like 3 backticks with {} followed by 3 backticks

An Example



For this example I used a short preamble about the purpose of the report. There is punctuation available for the text sections for more information please visit https://bookdown.org/yihui/rmarkdown/.

The Code

Structure Code

Structure code includes setting knitr options, importing the packages that will be used, and any other structure you may want to impose. I create a file structure for all my code project. Be sure to include the *include=FALSE* option in the code block {} for any structure code. See above for an example of this.

First I set up my knitr chunck options for a report.

```
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE)
```

You want to make sure that messages and warnings generated during the processing of the code do not write to the pdf report and *echo=FALSE* prevents the chucks from printing.

The structure code used for the Basic QC Report is the following

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE)
```

```{r folder_structure, include=FALSE}
dir.create("src")
dir.create("cache")
dir.create("data")
dir.create("results")
dir.create("plot")
```

```{r, pkgs_libraries, include=FALSE}
# This code will install required packages if they are not already installed
# This is useful for teaching but not requied

### Option 1
if (!require("knitr")) {
   install.packages("knitr")
   library(knitr)
}
if (!require("janitor")) {
   install.packages("janitor")
   library(janitor)
}
if (!require("tidyverse")) {
   install.packages("tidyverse")
   library(tidyverse)
}
if (!require("kableExtra")) {
   install.packages("kableExtra")
   library(kableExtra)
}

### Option 2
   library(here)
```

Tables and Figures

Before we can get to the tables and figures, the data used for the report needs to be imported and cleaned up. For this report there are two data sources a primary and secondary. The primary has the data that needs to be summarized and the secondary data contains the historically expected values for the QC.

Import Data

To import the data into the workspace, we use read.csv or readxl::read_xlsx depending on the file type. I also use the *here* and *janitor* packages. *here* allows one to traverse the file structure easier. *janitor* has multiple functions, but the one I use the most is *clean_names()* which standardizes the column names by making them all lower-case and removing special characters. There is one cavet if the a special character is important to identify the data then these tags need to be modified manually.

```r
```{r import_data, include =FALSE}

### Import the major dataset
dataset_df <- read.csv(here::here("data",
                                  "2020_AACC_DATASET.csv")) %>%
                                  janitor::clean_names()
### Import the supporting dataset
expected_df <- readxl::read_xlsx(here::here("data",
                                  "2020_AACC_expected_values.xlsx")) %>%
                                  janitor::clean_names()

```
```

Data Wrangling

I break data wrangling into sections so I don't forget to add something. First is data type transformations then tidying of the data so it is easier to use tidyverse functions. Then create any variables needed throughout the report (I update as the need arises!). Finally, I select the data I plan to use. I have found having this in one spot makes writing my R code more efficient. As part of data wrangling I am constantly checking the data types with *dplyr::glimpse()* and then I remove it in the final production code.

```r
```{r data_wrangling}
### Adjust any data type issues
dataset_df$date <- as.Date(dataset_df$date, "%m/%d/%Y")
dataset_df$x11desc <- as.numeric(dataset_df$x11desc)
dataset_df$test_code <- as.factor(dataset_df$test_code )
dataset_df$qc_code <- as.factor(dataset_df$qc_code )

### Make the data set more tidy for tidyverse funtions
dataset_df <- drop_na(pivot_longer(dataset_df, cols = c("tyr", "phe", "x17op", "prog", "cortms", "doc", "testms", "andro", "x11desc", "x17pre"),
names_to = "analyte"))

### Create start and end variable to be used throughout the report
start_date <- as.Date("2020-09-01")
end_date <- as.Date("2020-10-01")

### Select a portion of the data for whcih to create the report
one_month <- dataset_df %>%
   dplyr::filter(between(date, as.Date(start_date),
                         as.Date(end_date)) & test_code == "PATY" & analyte == "phe" )
```
```

Tables

The first part of the QC table code block is spent developing the data frame for reporting. We group by the qc_code and analyte then summarizing the data from theprimary data set. Next we combine the primary and secondary data set using the *left_join()* function and the *rename()* function to rename columns to better present in a report. At this point three calculations are performed. The first adds to the data frame the percent Coefficient of variation for the calculated data (Cal. %CV). The second is the standard deviation index (SDI) which is calculated by subtracting the expected and calculated means then dividing by the expected standard deviation (SD) with an ideal value of 0. In my mass spectrometry lab, SDI values between -0.8 to 0.8 are acceptable. The third calculation is the standard deviation ratio (SDR) which is calculated by dividing the expected SD by the calculated SD with an ideal value of 1. In my mass spectrometry lab, SDR values between 0.3 to 1.1 are acceptable.

```
**QC Data Table (`r start_date` to `r end_date`)**
```{r qc_table, error=FALSE}
###
### Build the dataframe that will be on the report
###

### Group the data
active_df <- active_data  %>%
   group_by(qc_code, analyte) %>%
   summarise(
   "Cal. Mean" = round(mean(value), 1),
   "Cal. SD" = round(sd(value), 1),
   "n" = length(value)
)
### Combine the primary and secondary data
active_df <- active_df %>%
   left_join(expected_df, by = c("qc_code", "analyte")) %>%
   rename(
   "Exp. Mean" = expected_mean,
   "Exp. SD" = expected_sd,
   "QC Code" = qc_code,
   "Analyte" = analyte
)

### Perform some calculations
active_df[['Cal. CV%']] <- round((active_df[['Cal. SD']]/active_df[['Cal. Mean']])*100, 1)
active_df[["SDI"]] <- round((active_df[['Cal. Mean']]- active_df[['Exp. Mean']])/active_df[["Exp. SD"]] , 1)
active_df[["SDR"]] <- round(active_df[["Exp. SD"]]/active_df[["Cal. SD"]] , 1)
```

Once that is accomplished, we move to the display and formatting of the column. The *knitr::kable* function and the package *kableExtra* will allow us to format the table easily 90% of the time per the kableExtra website. https://cran.r-project.org/web/packages/kableExtra/vignettes/awesome_table_in_html.html

```
dt %>%
  kbl() %>%
  kable_styling()
```

This is the most basic kableExtra output with dt = datatabe, kbl() calling *knitr::kable*, and kable_styling() applying format. With the data we have processed trying some different stylings by replacing kable_styling() with one of the following functions. 1. *kable_paper()* 2. *kable_classic()* 3. *kable_classic2()* 4. *kable_minimal()* 5. *kable_materials()* 6. *kable_material_dark()* –my favorite.

The *kable_styling()* function also allows for options by adding the option **bootstrap_options** but many of these are more relevant to html output rather than pdf. *kableExtra* also allow for the addition of a table footnote which I have used here to explain SDI and SDR because it is so long the option **threeparttable = TRUE** was used to prevent the table from rendering too small. Try it without and look at the resultant pdf.

```
###
### Use Kable and Kableextra to format the table for pdf output
###
active_df %>%
   kbl() %>%
   kable_styling(font_size= 10, latex_options=c("scale_down", "HOLD_position"), bootstrap_options = c("striped", "hover", "condensed")) %>%
   footnote(general = "SDI stands for standard deviation index and is calculated by substracting the expected and calculated means then dividing by the
expected standard deviation (SD) with an ideal value of 0. SDI values between -0.8 to 0.8 are acceptable in my mass spectrometry lab. SDR is the
standard deviation ratio and is calculated by dividing the expected SD by the calculated SD with an ideal value of 1. SDR values between 0.3 to 1.1 are
acceptable in my mass spectrometry lab.",
           threeparttable = T)
```

Figures

A basic Levey-Jennings plot was created with *dplyr::group_by*, *dplyr::mutate*, and *ggplot*.

Please create a new code chunk for the plot and add the following code.

```
### Group, scale, and plot the data
active_data %>%

### Group
group_by(qc_code) %>%

### Scale so all the data fits on one plot
```
```

```
mutate(value_scaled = (value-mean(value))/sd(value)) %>%

### Plot the data
ggplot(aes(x = date, y = value_scaled, color = qc_code, ymin = -1, ymax = 1)) +
   ### 1 SD range
   geom_ribbon( alpha = 0.3) +
   ### 2 SD range
   geom_ribbon(aes(ymin =-2, ymax = 2), alpha =0.2) +
   ### 3 SD range
   geom_ribbon(aes(ymin =-3, ymax = 3), alpha =0.1) +
   geom_line() +
   geom_point() +
   ggtitle("Levey-Jennings Chart") +
   xlab("Date") +
   ylab("SD Scaled Values")
```

At this time a basic pdf QC Report can be produced by using Knit. Give it a try.

Parameterized Reports

Have you ever wanted to modify some R code that you have but didn't want to copy and paste then find and replace all the variables?

I have and one way to accomplish this is through parameterization. So next we are going to parameterize our report we just created.

Three areas will have to be modified. 1. The YAML 2. The data seletion portion of the report 3. We need to knit with parameters

Updating the YAML

Our previous YAML should look something like this.

```
author: "Dustin R. Bunch"
date: "November 23rd, 2020"
geometry: margin=0.75in
header-includes:
- \usepackage{fancyhdr}
- \pagestyle{fancy}
- \fancyhead[LO,LE]{\includegraphics[width=4in,height=1.5in]{Example_header.jpg}\\}
- \fancyhead[RO,RE]{Department of Pathology \\ and Laboratory Medicine,\\ Biochemical
  Genectis Section}
- \fancyfoot[LO,LE]{Institutional Disclaimer:}
output:
  pdf_document:
    df_print: kable
  extra_dependencies: threeparttable
  html_document:
    df_print: paged
```

We are going to add params: to lowest layer of the YAML. This allows us to supply parameters to the report at the beginning of excution without having to do a find and replace. These parameters are stored in a varialble call "params" which can be access within the program using a standard R call of params$var1. These variable will have names that are supplied by the user when creating the params: section in the YAML. I created 5 variables that may be useful and set their default values in the params: section of the YAML.

```
params:
   start: !r lubridate::today()-30
   end: !r lubridate::today()
   testcode: "PATY"
   analyte: "phe"
   data:
      label: "Input Dataset:"
      value: data/2020_AACC_DATASET.csv
      input: file
```

As you can see R code can also be used in the YAML with **"!r package::function"**. These parameters can be access within code with calls of *params$start*.

Updating the data selection portion of the code

Within the data_wrangling code chuck is the code of data select.

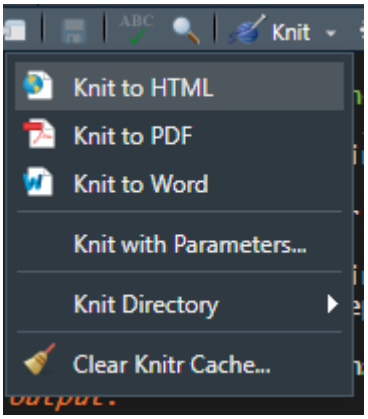
Please try to update this section using the data stored in the params.

If everything worked out it should look something like this

```
### Select a portion of the data for whcih to create the report
active_data <- dataset_df %>%
   dplyr::filter(between(date, as.Date(params$start),
                         as.Date(params$end)) & test_code == params$testcode & analyte == params$analyte )
```

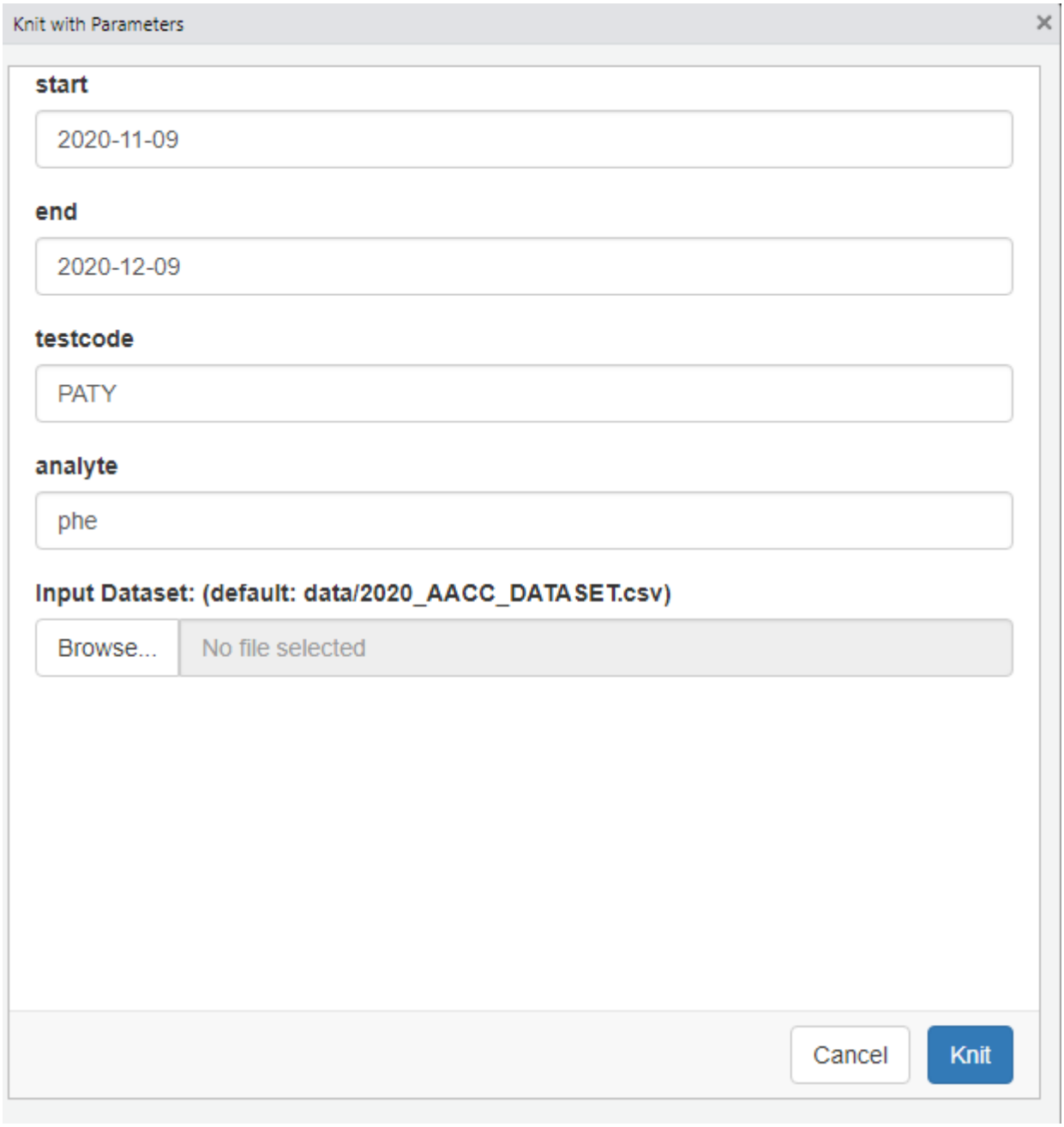
Knitting with parameters

To knit a parameterized document properly, you need to select the down arrow next to knit and select "knit with parameters" which will be the fourth option.



If you don't knit the document with parameters after making the parameters changes the R Markdown will create an error depending on what is missing. For this example the error is concerned with the plot.

Once "knitting with parameters" is selected, a pop-up with the parameters listed in the parameters list. The defaults are listed there but can easily be changed.



The data set has testcodes CAH6 and PATY and analytes tyr = tyrosine, phe = phenylalanine, x17op = 17OH-progesterone, prog = progesterone, cortms = cortisol, doc = deoxycorticosterone, testms = testosterone, andro = androstenedione, x11desc = 11-desoxycortisol, x17pre = 17OH-pregnenolone.

Give them a try and see how the data changes.