Simplified Technical Report Tables

A Healthy Waters Partnership Analysis

Adam Shand

October 21, 2024

Table of Contents

# Introduction

This script is an experimental script written to test a variety of table output options from an R script. The R packages we are looking at for this analysis are:

* openxlsx2
* kable (and kableExtra)
* huxtable

However it is important to note that before this selection of packages was reached, several additional packages were piloted such as “gt”, “reactable”, “openxlsx”, “DT”, etc. Upon reviewing each of these packages, core flaws were identified that limited their feasibility.

The general objectives that we are trying to achieve are:

1. Manipulate tables as “normal” using tidyverse syntax
2. Save tables to a csv file with no formatting
3. Save tables to a xslx file **with** conditional formatting
4. Present tables in Quarto outputs with conditional formatting, specifically:
   * HTML
   * docx (word)

## Script Set Up

This script requires multiple core spatial packages, each of which is loaded below.

# Load Example Data

First lets bring in some example tables to conduct our comparison on.

These tables, each demonstrate different challenges we will face. Example Table 1 contains natural numbers from 0 to 100, which represent the range of values we would expect to see in a standard reporting table creating within our workflow. Example Table 1 also contains several “trip ups” - such as negative numbers, NA values, character values, and values greater than 100. These are designed to test the unknown/error handling capacity of each of the methods that we will explore.

Example Table 2 expands on this, with additional columns and rows that add context to the table.

Example Table 3 explores a niche subset of tables that are sometimes used that report values between a range of -1 to +1, all other challenges remain the same.

# Openxlsx2

The openxlsx2 package has been specifically designed to save tables from R to the .xlsx excel format. However it does not offer any capacity to print these tables inline, to html, or to word. Lets review each of the processes we wanted to complete again:

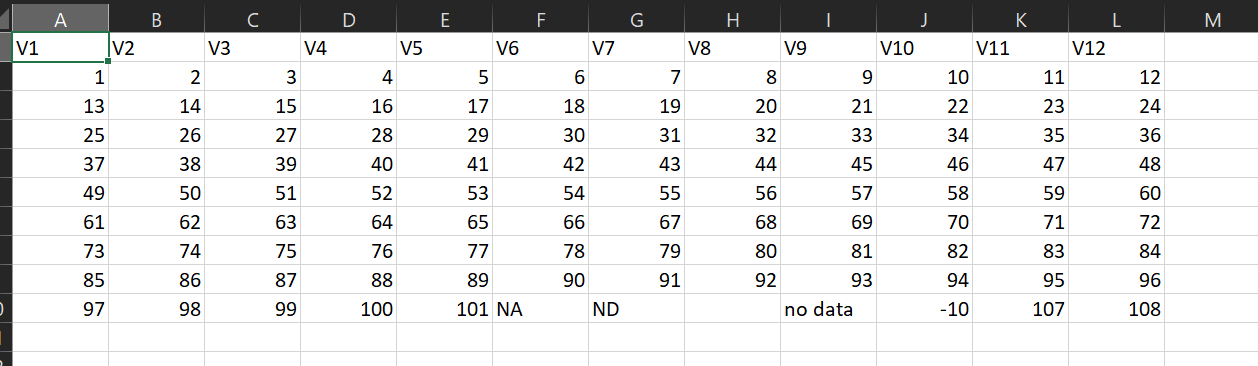
1. Manipulate tables as “normal” using tidyverse syntax (yes)
2. Save tables to a csv file with no formatting (using standard tidyverse functions yes)
3. Save tables to a xslx file **with** conditional formatting (yes)
4. Present tables in Quarto outputs with conditional formatting, specifically:
   * HTML (no)
   * docx (word) (no)

Although this package does not cover all of our needs, it does cover the niche requirement of saving to xlsx. Thus I have written functions that utilise openxlsx2 for saving tables to xlsx, these are demonstrated below:

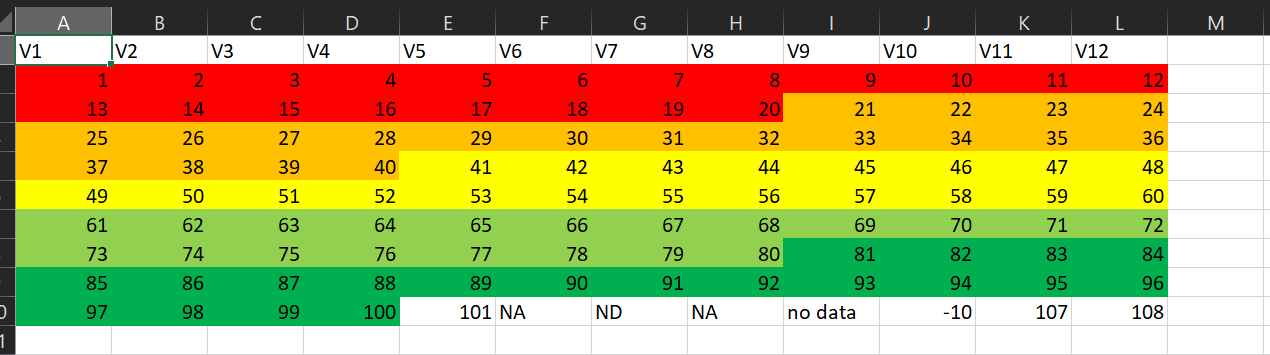
## Export to Excel With Colour

Saving the table with colour can be achieved using the .xlsx workbook file format (which retains formatting/styling options) and a custom function as such:

Here is a screenshot of before and after, note the variety of NA type values that are ignored:



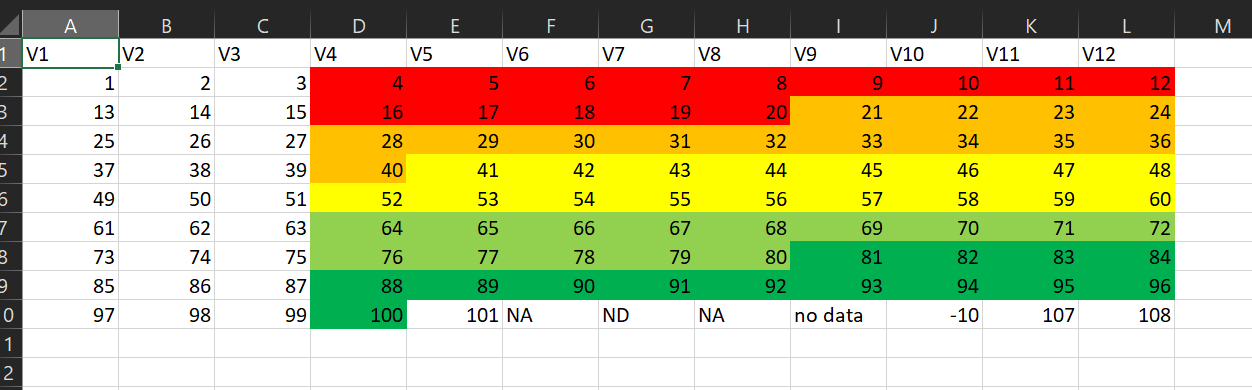
coloured Table Original



Coloured Table

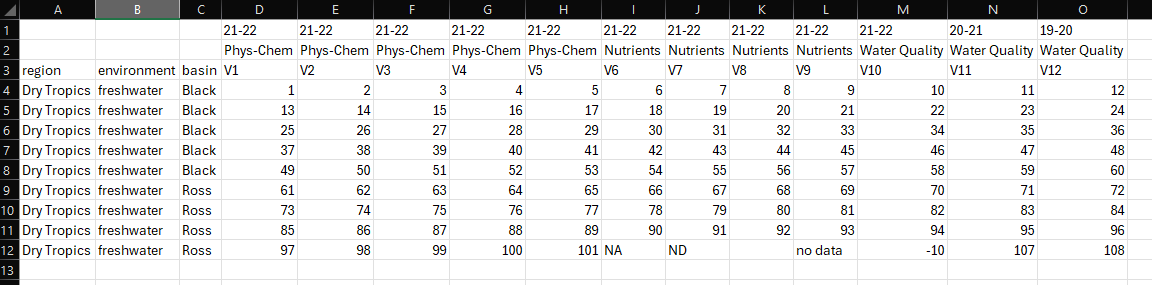
It is also very simple to change where the colouring is applied. For example, lets pretend the first three columns are variables such as site, basin, and zone. These columns shouldn’t be colored, and this can be achieved by changing the starting point for the function:

which looks like this:



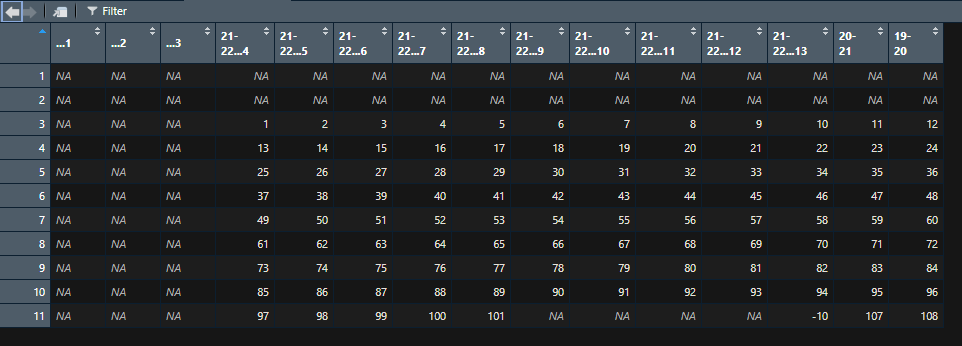
Coloured Table 2

It is also equally possible to handle multiple header rows containing non numeric values. Consider the following table:



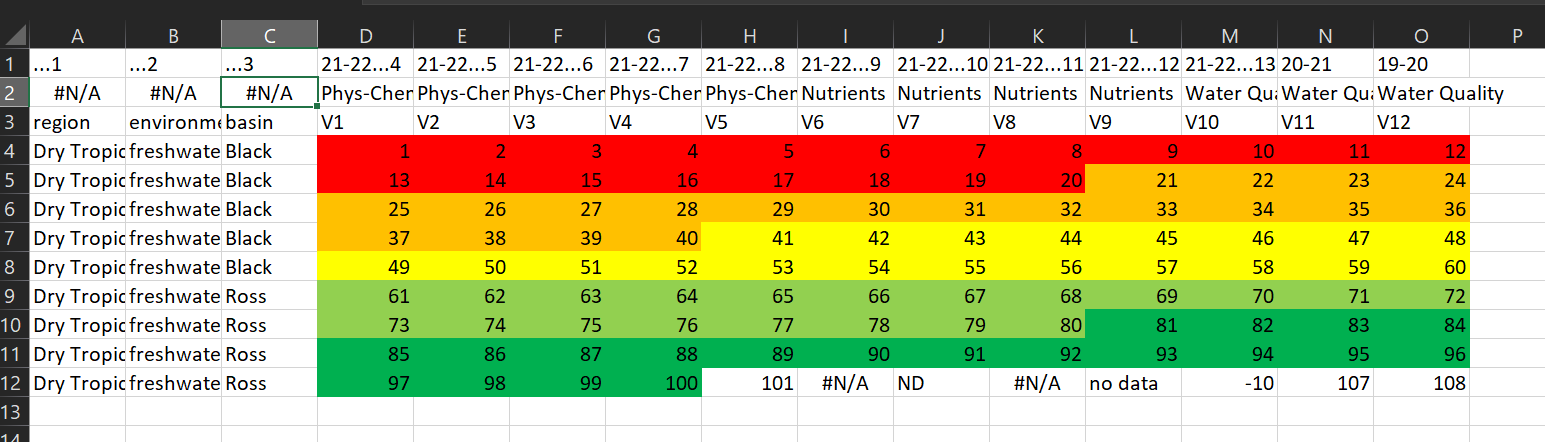
Coloured Table 3

which, thanks to the first two rows, forces all indicator columns to be character. Although, given that we are colouring by comparing the score value against a range of values (i.e. is x < or > y), we actually need columns to be numeric. However, when converting to numeric, the character values in rows one and two are replaced with NA:



Coloured Table 4

However the colour function accounts for this and replace NA values in the output, with its original value that was read from the input. True NAs remain NAs, and character values that get wiped will be restored. There isnt even any need to specify where the colour rules should begin using this method.



Coloured Table 5

# Kable

The kable and kableExtra packages have been written with the main purpose of being simple and easy to use for HTML outputs. However with the draw back of very limited support for any other situation. If we were to once again review our processes that we wanted to complete:

1. Manipulate tables as “normal” using tidyverse syntax (yes)
2. Save tables to a csv file with no formatting (using standard tidyverse functions yes)
3. Save tables to a xslx file **with** conditional formatting (no)
4. Present tables in Quarto outputs with conditional formatting, specifically:
   * HTML (yes)
   * docx (word) (very very basic)

Given that several R packages provide the ability to save table to a html format, the viability of this table is limited. However we will demonstrate a use case below to explore how user friendly the package is.

## Print to HTML

Below we demonstrate how kable can print a table to a html formatted document, with conditional formatting.

However this method does not retain character values. This may become more apparent with the more complex example table 2:

The reason for this is that each column in an R data frame must be of a single type (e.g. character, numeric, boolean), and the actual formatting of the table is taking place within the R engine. It may be possible to make each column a list, which can contain multiple types, but this package is supposed to be reducing complexity not increasing it.

## Print to Word

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 1: This is a table caption.   | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | | 97 | 98 | 99 | 100 | 101 | NA | ND | NA | no data | -10 | 107 | 108 | |

# Huxtable

The final option of the lot is the huxtable package. This package boasts the most support, and when we refer to our requirements:

1. Manipulate tables as “normal” using tidyverse syntax (yes)
2. Save tables to a csv file with no formatting (using standard tidyverse functions yes)
3. Save tables to a xslx file **with** conditional formatting (no)
4. Present tables in Quarto outputs with conditional formatting, specifically:
   * HTML (yes)
   * docx (word) (yes)

We can see that in combination with the openxlsx2 package it covers every requirement that we have. To demonstrate its flexibility, several examples are presented below.

## Inline HTML

first up is a demonstration of inline HTML. A simple, unformatted table can be created as follows:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |
| 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 |
| 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 |
| 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 |
| 97 | 98 | 99 | 100 | 101 |  | ND |  | no data | -10 | 107 | 108 |

Simple styling can be set as follows:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **V1** | **V2** | **V3** | **V4** | **V5** | **V6** | **V7** | **V8** | **V9** | **V10** | **V11** | **V12** |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |
| 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 |
| 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 |
| 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 |
| 97 | 98 | 99 | 100 | 101 |  | ND |  | no data | -10 | 107 | 108 |

Which works just as well on the more complicated table:

And by the way, here is an example of landscape layout and of a custom function to style the table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Region** | **Environment** | **Basin** | **2021-2022** | | | | | | | | | | **2020-2021** | **2019-2020** |
| **Phys-Chem** | | | | | **Nutrients** | | | | **Water Quality** | | |
| V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 |
| Dry Tropics | freshwater | Black | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
| 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 |
| 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 |
| 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| Ross | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 |
| 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 |
| 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 |
| 97 | 98 | 99 | 100 | 101 |  | ND |  | no data | -10 | 107 | 108 |

However by far the more interesting component is the conditional formatting that we can apply, which is actually ridiculously easy ([Table 2](#tbl-exampledata))

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Table 2: This is a table caption.   |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **V1** | **V2** | **V3** | **V4** | **V5** | **V6** | **V7** | **V8** | **V9** | **V10** | **V11** | **V12** | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | | 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 | 71 | 72 | | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 | | 85 | 86 | 87 | 88 | 89 | 90 | 91 | 92 | 93 | 94 | 95 | 96 | | 97 | 98 | 99 | 100 | 101 |  | ND |  | no data | -10 | 107 | 108 | |

# Final Decision

Broadly speaking, no one package can do all that is required., thus two packages can be selected. Of the three packages assessed, only the openxlsx2 package can handle conditional outputs to the .xlsx file type. From the remaining two options, huxtable is by far the better option as it can print to both html and docx formats (although does require minor adjustments between each).

Therefore, the final selection of packages to use for handling all table outputs from R scripts is (as of 27/09/2024):

* openxlsx2
* huxtable