Create a simple Word Report

Step-by-step correction

Hélène Langet Zhihan Zhu

2025-02-03

Table of contents

This page provides a step-by-step correction for building a simple MS Word report using the Quarto notebook `exercise3.qmd` as a starting point.

> **!** Important
>
> To get the most out of your learning experience, attempt to solve the exercise on your own before looking at this correction. Remember, trials and errors are an essential part of the learning process, strengthening your understanding and helping you build confidence. Take your time, experiment, and learn actively!

1 Setup basic elements of the Quarto notebook

All the basic elements you need to setup are defined in the YAML header at the beginning of your Quarto notebook.

```yaml
---
title: My outbreak report                    ①
author: Helene Langet                        ②
```

```yaml
date: 2023-12-31                                                    ③

format: docx                                                        ④

execute:                                                            ⑤
  echo: false                                                       ⑥
  warning: false                                                    ⑦
---
```

① Update the `title` option in the Quarto notebook. Quotation marks are optional.
② Set the `author` option to your name. Quotation marks are optional.
③ Add the `date` option and set it to **2023-12-31**. Quotation marks are optional.
④ Set the `format` option to **docx** to generate a MS Word document.
⑤ Add an `execute` block to customize the execution behaviour for your Quarto notebook. This block allows you to control how code, warnings, and other outputs are displayed in the rendered document.
⑥ Add the `echo` option and set it to **false** to hide code in the rendered document.
⑦ Add the `warning` option and set it to **false** to hide warnings in the rendered document.

> **!** YAML indentation
>
> Remember that YAML is a whitespace-sensitive language where indentation determines the structure; tabs are not recognised for indentation. The recommended practice is therefore to use **two spaces** per indentation level to ensure consistency and avoid errors.

2 Create publication-ready summary statistics tables

Creating publication-ready tables in R often involves leveraging specialized packages. Popular options include `gtsummary`, `gt` and `flextable`. In what follows, we will use `gtsummary` which is an excellent package for creating summary statistics tables. To install it, run the command `install.packages("gtsummary")` in your R console.

Table 1

````
```{r}
#| tbl-cap: Population characteristics # <10>

#Table 1
#Generate a summary table displaying population characteristics
```
````

Table 1: Population characteristics

Characteristic	**N = 65,669**[1]
age	50 (35, 65)
sex	
1	33,114 (50%)
2	32,555 (50%)
bmi	29 (21, 38)
confirmed	
0	13,235 (20%)
1	52,434 (80%)
death	
0	64,455 (98%)
1	1,214 (1.8%)

[1]Median (Q1, Q3); n (%)

```
subdf |>
  dplyr::select(age, # <8>
                sex,
                bmi,
                confirmed,
                death) |>
  gtsummary::tbl_summary() # <9>
```

⑧ Use the `dplyr::select()` function to select specific columns (**age**, **sex**, **bmi**, **confirmed**, and **death**) from the **subdf** dataset.

⑨ Use the `gtsummary::tbl_summary()` function to create a table that summarizes the descriptive statistics of the selected variables in the whole population.

⑩ Add a caption to the table by using the `tbl-cap` chunk option.

Table 2

```
```{r}
#| tbl-cap: Demographic characteristics of deceased vs. alive # <12>

#Table 2
#Generate a summary table comparing the demographic characteristics
```
```

Table 2: Demographic characteristics of deceased vs. alive

Characteristic	**Overall** N = 65,669[1]	**0** N = 64,455[1]	**1** N = 1,214[1]
sex			
1	33,114 (50%)	32,504 (50%)	610 (50%)
2	32,555 (50%)	31,951 (50%)	604 (50%)
age	50 (35, 65)	50 (35, 65)	52 (37, 67)
bmi	29 (21, 38)	29 (21, 38)	34 (28, 41)

[1]n (%); Median (Q1, Q3)

```
#of individuals who died versus those who are still alive
subdf |>
  dplyr::select(sex,
                age,
                bmi,
                death) |>
  gtsummary::tbl_summary(by = death) |> # <10>
  gtsummary::add_overall() # <11>
```

(10) Use the `gtsummary::tbl_summary()` function to create a summary table grouped by the **death** column, showing separate statistics for individuals who have died vs. those who are still alive.

(11) Use the `gtsummary::add_overall()` function to add a row to the table that shows overall summary statistics for the whole population, regardless of their death status.

(12) Add a caption to the table by using the `tbl-cap` chunk option.

3 Create publication-ready figures

Figure 1

```
```{r}
#| fig-cap: Weekly count of all cases, confirmed cases and deaths # <13>

#Figure 1
# Aggregate the data to get the weekly count of all cases, confirmed
 ↪  cases and deaths
```
```

```
weekly_data <- subdf |>
  dplyr::group_by(week) |>
  dplyr::summarise(count = dplyr::n(),
                   confirmed_count = sum(confirmed == "1"),
                   death_count = sum(death == "1"))

# Plot the weekly cases, confirmed cases and deaths
ggplot2::ggplot(weekly_data, ggplot2::aes(x = week)) +
  ggplot2::geom_line(ggplot2::aes(y = count,
                                  color = "All cases"),
                     size = 1) +
  ggplot2::geom_line(ggplot2::aes(y = confirmed_count,
                                  color = "Confirmed cases"),
                     size = 1) +
  ggplot2::geom_line(ggplot2::aes(y = death_count,
                                  color = "Confirmed deaths"),
                     size = 1) +
  ggplot2::labs(x = "Week",
                y = "Count",
                color = "Legend") +
  ggplot2::scale_color_manual(values = c("All cases" = "#440e54", # <14>
                                         "Confirmed cases" = "#f8766d",
                                         "Confirmed deaths" =
  "#128984")) +
  ggplot2::theme_minimal() + # <15>
  ggplot2::theme(panel.grid.major.y = ggplot2::element_line(linewidth =
  0.5, # <16>
                                                            linetype =
  "dashed",
                                                            color =
  "grey"),
                 panel.grid.minor.y = ggplot2::element_line(linewidth =
  0.5,
                                                            linetype =
  "dashed",
                                                            color =
  "grey"),
                 panel.grid.major.x = ggplot2::element_blank(), # <17>
                 panel.grid.minor.x = ggplot2::element_blank())
```

(13) Add a caption to the table by using the `fig-cap` chunk option. Remove the title parameter from `ggplot2::labs` to avoid the repetition.
(14) Assigns specific colors to each line for better distinction.
(15) Apply the `ggplot2::theme_minimal()` function to set a clean and simple aesthetic for the plot. This theme removes unnecessary elements such as background shading and focuses attention on the data by displaying only grid lines and axis labels.
(16) Add dashed lines for horizontal (major and minor) grid lines.
(17) Omits vertical (major and minor) grid lines for a cleaner look.

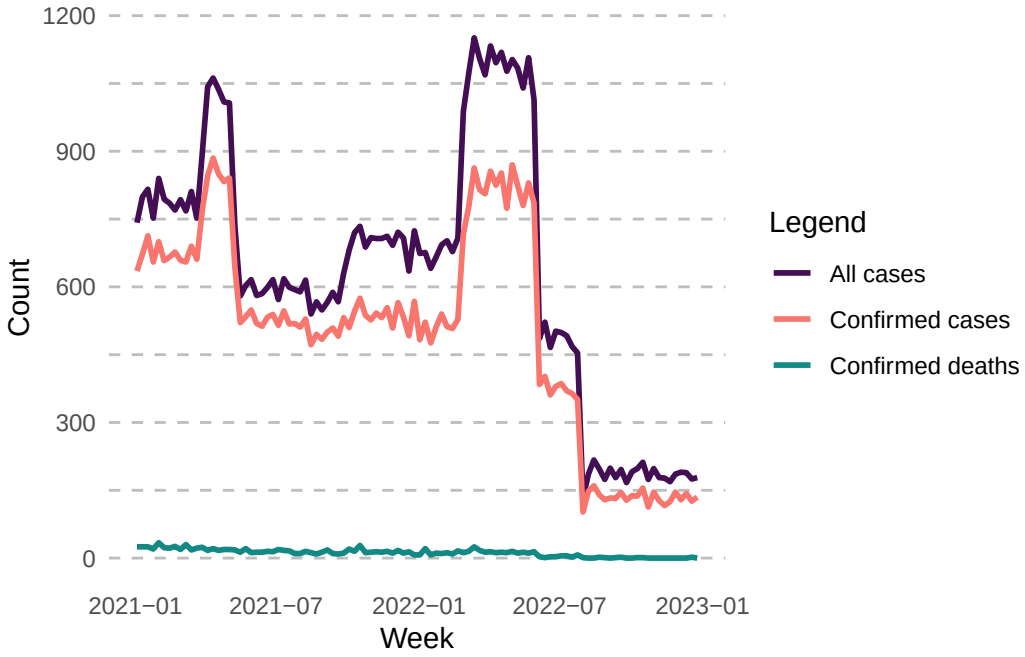

Figure 1: Weekly count of all cases, confirmed cases and deaths

4 Present statistical models and results

```r
```{r}
#| echo: true # <18>

#Logistic regression model
model <- glm(death ~ bmi + age, # <19>
             subdf |> dplyr::filter(confirmed == "1"), # <21>
             family = binomial) # <20>
```
```

```
```

(18) The `echo: true` option ensures that the code chunk used to generate the logistic regression model is displayed in the rendered output, which is helpful for transparency and reproducibility.

(19) The `glm()` function fits a generalised linear model to predict **death** (binary outcome) from two predictors (**bmi** and **age**). The function returns a model object containing various components, such as coefficients (effects of predictors), residuals (errors), and fitted values (predicted probabilities).

(20) Set the `family` argument to **binomial** to fit a logistic regression model. This is appropriate for binary outcomes like death, as it models the probability of the outcome occurring using a logit (*log-odds*) function (i.e. coefficients represent the effect size of each predictor on the log-odds of the binary outcome).

(21) Filter data using the `dplyr::filter()` function to subset the dataset so that the model is estimated from confirmed cases only.

Table 3

```
```{r}
#| tbl-cap: Formatted regression table # <24>

#Table 3
gtsummary::tbl_regression(model, # <22>
                          exponentiate = TRUE) # <23>
```
```

(22) Use `gtsummary::tbl_regression()` to create a table summary of the logistic regression results.

(23) Set the `exponentiate` argument to **TRUE** to convert the coefficients (*log-odds*) into odds ratios, which are more interpretable and useful for understanding the strength of associations.

(24) Add a caption to the table by using the `tbl-cap` chunk option.

Table 3: Formatted regression table

Characteristic	**OR**[1]	**95% CI**[1]	**p-value**
bmi	1.04	1.03, 1.04	<0.001
age	1.00	1.00, 1.01	0.003

[1]OR = Odds Ratio, CI = Confidence Interval