Document-based Report Generation using officer Package of R

Ojaswini Mandar Oak[1], Pranavkumar A Bhadane[2]
[1]*Student, Artificial Intelligence and Machine Learning Department, Agnel Polytechnic, Vashi*
[2]*Professor, Artificial Intelligence and Machine Learning Department, Agnel Polytechnic, Vashi*

***Abstract*—In many government and corporate settings, routine report generation is common. These reports are typically reviewed and edited in Microsoft Word format due to its widespread accessibility and familiarity. Often, these reports utilize statistical outputs, like tables and graphs, to communicate the results. As underlying data changes, multiple iterations of report drafting and submission are required. In such situations, manual transfer of analytical outputs to word becomes time consuming, error prone and tedious. We intend to publish a series of technical papers which explore the potential of open-source technologies to automate and reproduce such reports efficiently. We commence this series by demonstrating the capabilities of officer package of R in seamlessly exporting tabular and graphical content into Word documents.**

***Index Terms*—R, R Programming, officer package, Report Generation, Reproducible Reports, Word Document, Automated Reports, Programmatically Generated Reports**

I. INTRODUCTION

Many industries require repetitive report generation. A few examples include Finance, Market Research, Telecommunication, Retail, E-commerce, Clinical, Logistics, Government, etc. While final submission may happen in PDF format, draft reports are frequently created, reviewed, commented, edited, and finalized as Microsoft Word documents. Many analysts and researchers are familiar with statistical capabilities of R and Python, but struggle to export results to Word documents, and hence land up manually transferring results to Word or RTF files. R provides many tools to produce automated and reproducible reports [1][2], which ensure that the entire workflow – from the data analysis stage to the final document formatting – can be automatically regenerated without errors. In this paper, we focus on the officer package [3] of R [4], for exporting such reports. To demonstrate capabilities of functions from this package, we have generated example tables and visualizations using the flextable package [5] and ggplot2 package [6]. By combining paragraphs, tables and graphs with customized layouts, formats, and styles, we can produce professional, error-free, and reproducible reports.

II. A BRIEF OVERVIEW OF R

R is an open-source software system and environment for implementing statistical techniques, data manipulation and generating customized, professional graphs and tables. R provides several packages and tools for report generation. A few of these are given in Table I

Table I: Report Generation Tools in R

Package / Tool	Description
officer	Used for generating outputs in word, with extensive control over the formatting, styling, layouts etc
r2rtf [7]	Used for exporting rich text format files, mainly for clinical reports
rtf [8]	Used for exporting RTF files with base R plots and tables using basic formatting
R markdown	Literate programming tool used with knitr[9] and rmarkdown[10] package for rendering documents in multiple formats. The knitr package provides an interface for dynamic and reproducible report in R Markdown and Quarto document, which is implemented by rmarkdown package [11].
Quarto [12]	Next generation of R Markdown with support for multiple languages and rendering into several formats

To generate examples of graphical and tabular outputs, we have used the ggplot2 package and flextable package of R (see Table II). However, these packages have not been explored in great depth in this paper, as the focus of this paper is on document-based report generation using officer. Readers interested in a more detailed exploration of these packages are encouraged to study the following references: [5], [6] and [13].

Table II: Tabulation and Visualization Tools in R

Package / Tool	Description
ggplot2	ggplot2 package in R provides extensive functionality to create various types of charts
flextable	Flextable package in R offers many functions and inbuilt themes which can be applied to a data-frame to style and customize it according to the user's needs

An example of a flextable would-be Table I and Table II presented above describing the functionality of various packages. All the subsequent visualizations contained in the examples shown in the next sections of this paper have been created using ggplot2 and flextable package. The R code for each table and graph can be accessed via a GitHub repository created for this paper, the link for which has been provided in the References section [14].

III. EXPORTING A ONE-PAGE DOCUMENT

The process of exporting plots or tables into a word document is made fairly simple using the officer package. Below are two examples demonstrating the creation of two simple word documents, one which includes a single flextable object, and another which contains a single ggplot2 object. To begin exporting a document using the officer package, the first step that needs to be taken is initializing a new Word Document object using the read_docx() function. read_docx() creates an empty .docx file. The result is stored in the variable assigned to the function. Subsequent content insertion and formatting are performed on this blank .docx file. To save this file in the current working directory, print() function is used, with the value of the 'target' argument specifying the name of the file.

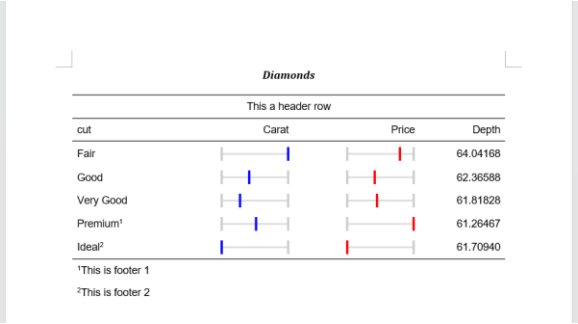

Fig. 1: A single table on a Page

The flextable in Fig. 1 has been inserted using body_add_flextable () function of officer. For the plot in Fig. 2, body_add_gg () function of officer has been used.

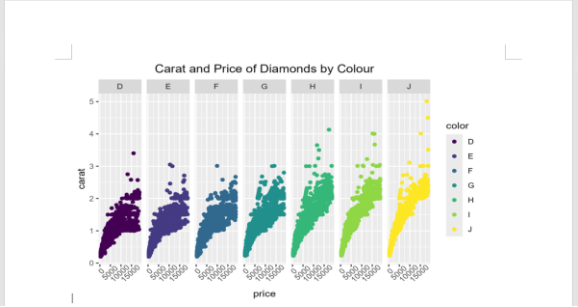

Fig. 2: A single plot on a Page

The link for the R code for exporting the table and the plot can be viewed by visiting [15] and [16] in the References section.

IV. DOCUMENT WITH MULTIPLE TABLES ON A SINGLE PAGE WITH LANDSCAPE LAYOUT

Fig. 3 shows two flextables exported to a word document using the officer package. A series of steps need to be taken to achieve this output. First, create both tables using flex table (). To orient the page in landscape, assign an object the necessary properties using page_size = page size (orient = 'landscape') option of the prop_section() function. To export the tables, a word document needs to be created using read_docx () function. Blank lines can be added using body_add_par (). Use body_add_flextable () function to insert the flextables. Lastly, apply landscape properties created previously to the document using body_set_default_section () function from officer. The GitHub link to access the R code for exporting this document can be found in the References Section [17].

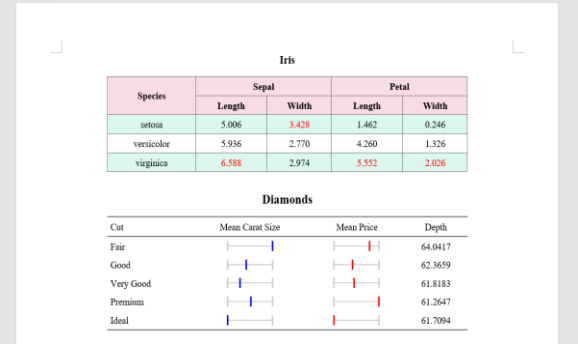

Fig. 3: Multiple tables in a single document

V. DOCUMENT WITH MULTIPLE PLOTS CONTAINED IN MULTIPLE SECTIONS ON A SINGLE PAGE

Officer package includes functions to generate a document with multiple sections on a single page. This has been demonstrated in the next example (Fig. 4). The page contains primarily two sections, which divide the page into two parts (top and bottom). The bottom section has been further customized to show content in two columns with user defined width. The top section contains one plot, the left-hand side of the bottom section contains another plot, and the right-hand side contains some text. This paper is another example of a document with multiple sections. All the columns and images have been divided into separate sections. Each section has been assigned a two-column or one-column layout using block_section() and body_end_block_section () of the officer package. The code for the output shown in Fig. 4 can be found in the References Section [18].

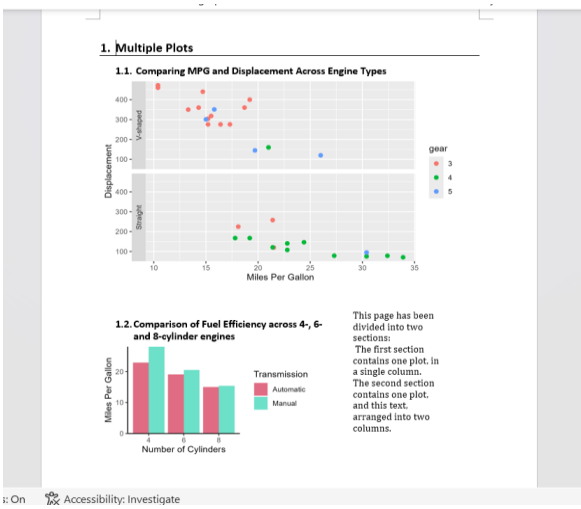

Fig. 4: Multiple sections in a single document

VI. MULTIPLE PAGES FROM A SINGLE R SCRIPT

If a report requires similar contents for every page, it is possible to create loops and functions to generate each page and add it to the document. Here is an example showcasing both a table and a plot panel added to every page in the document. Since each page follows the same formatting, with only the underlying values changing, it is convenient to just create a function to generate each page and loop through the values. This document basically shows the statistics of answers given to each question of a feedback form. There are 13 questions in total- hence there are 13 pairs of tables and pie charts in this document, each describing the scores on each question. The data for this document is obtained from an excel file with two sheets- one containing questions, and one containing score. Fig. 5 shows 3 of the 13 programmatically generated pages using loops. The R code for this example can be accessed using the link provided in the References Section [19].

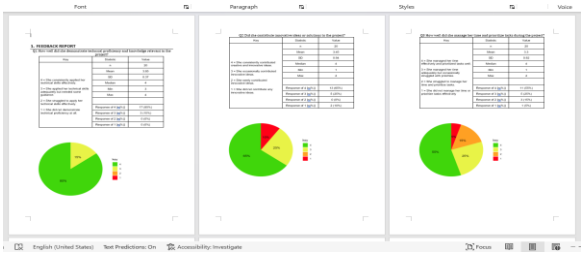

Fig. 5: Multiple Pages of the same format with one table and one plot

VII. MULTIPLE DOCUMENTS FROM A SINGLE R SCRIPT

In the previous case, all the pages were added to a single document. It is also possible to generate separate documents for each question in the feedback question list. The formatting stays the same, but instead of adding the pages to a single document, we modify our function to print every page as a separate word document, keeping the formatting of each document the same. So, we export multiple reports (13 in this case as there are 13 questions) from a single R script. The names of the files shown in Fig. 6 will be Question1.docx, Question2.docx and so on. The R code for this example can be accessed via the link provided in the References Section [20].

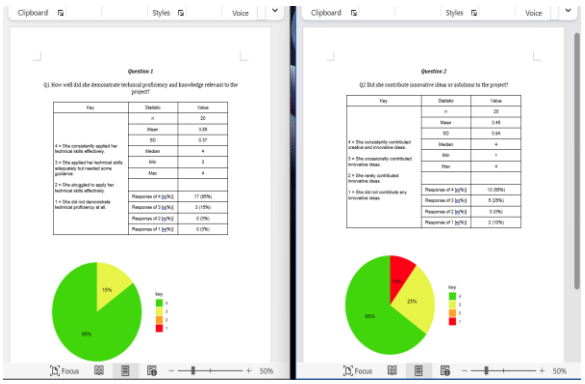

Fig. 6: Multiple documents from a single R script

VIII. CREATING A DOCUMENT WITH CUSTOM PAGE ORIENTATION

The officer package provides functions to orient specific pages to a different layout. This can be observed in Fig. 7. The first and last pages are portrait, and the middle page is landscape. All this has been done using the prop_section() function. To center the plot in each layout, it is first saved as an image. Then it is added as an external_image() in fpar(). Lastly, in the fp_p option of fpar(), fp_par(text.align = 'center') has been applied to center align the image. The GitHub Link to access this code can be found in the References Section [21].

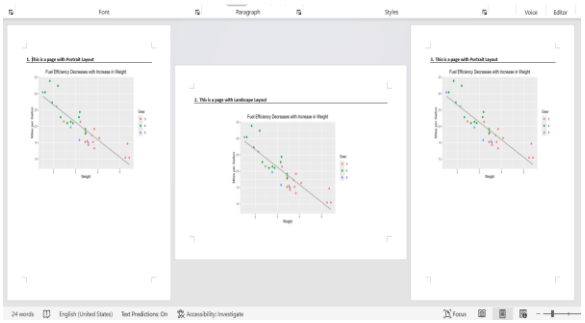

Fig. 7: Custom Layouts using Officer

IX. COMBINING MULTIPLE TABLES INTO A SINGLE TABLE

The officer package offers great flexibility and versatility in report generation. It can be used for general reports as well as specific ones. Here is an example of a clinical study report. Fig. 8 shows the first page of a three-page document providing the summary of demographic and baseline characteristics of subjects enrolled in a clinical trial. It consists of several tables, each one being a flextable object. The header of all the flextables except the first are removed using delete_part(part = 'header'). A vector for column width is created, and this vector is passed to the width () function for every table. All the flextables are added to the document using the body_add_flextable () function. A template is used to achieve the landscape layout, but this can also be done using officer's body_set_default_section() function. Font family and font size have been set using the set_flextable_defaults() and fpar() functions. The R code for this example can be accessed via the link provided in the References Section [22].

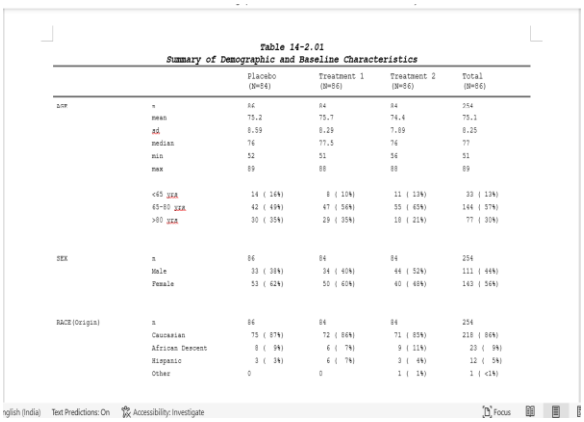

Table 14-2.01
Summary of Demographic and Baseline Characteristics

		Placebo (N=84)	Treatment 1 (N=86)	Treatment 2 (N=86)	Total (N=86)
AGE	n	84	84	84	254
	mean	75.2	75.7	74.4	75.1
	sd	8.59	8.29	7.89	8.25
	median	76	77.5	74	77
	min	52	51	56	51
	max	89	88	88	89
	<65 yrs	14 (16%)	8 (10%)	11 (13%)	33 (13%)
	65-80 yrs	42 (49%)	47 (56%)	55 (65%)	144 (57%)
	>80 yrs	30 (35%)	29 (35%)	18 (21%)	77 (30%)
SEX	n	86	84	84	254
	Male	33 (38%)	34 (40%)	44 (52%)	111 (44%)
	Female	53 (62%)	50 (60%)	40 (48%)	143 (56%)
RACE(Origin)	n	86	84	84	254
	Caucasian	75 (87%)	72 (86%)	71 (85%)	218 (86%)
	African Descent	8 (9%)	6 (7%)	9 (11%)	23 (9%)
	Hispanic	3 (3%)	6 (7%)	3 (4%)	12 (5%)
	Other	0	0	1 (1%)	1 (<1%)

Fig. 8: Document with multiple sections in a table

X. CREATING A DOCUMENT BASED ON AN EXISTING TEMPLATE

Generating documents based on a template can be very useful while creating reports for an organisation, or ones that need to adhere to a certain format or layout. Document creation based on a template provides many additional facilities such as headers, footers, watermarks, page numbers, custom styles, and layouts. To create a template, first create a blank word document. Add your custom styles, headers, footers, and watermarks, and save this word file with any name, say 'MyTemplate.docx'. In your R script, pass the file name of your template as an argument in read_docx (). The document will then be created based on that template. For example, the document shown in Fig. 9 is created with a template. It has a header, footer, watermark, and page numbers on each page. The R code for this example can be accessed via the link given in the References Section [23].

Fig. 9: Document based on an existing template

XI. CONCLUSION

All the examples shown previously showcase various functionalities officer provides for creating document-based reports. This entire paper itself is an example of officer package's versatility and flexibility. All the custom styles, page numbers, layouts and images have been added using various officer functions into an R script. The officer package has many more functionalities, all of which are beyond the scope of this paper. Interested users can further explore the capabilities of officer package by visiting [24] and [25].

REFERENCES

[1] Peikert, Aaron & Van Lissa, Caspar & Brandmaier, Andreas. (2021). Reproducible Research in R: A Tutorial on How to Do the Same Thing More Than Once. Psych. 3. 836-867. 10.3390/psych3040053.

[2] Janet E Simons, Daniel T Holmes, Reproducible Research and Reports with R, The Journal of Applied Laboratory Medicine, Volume 4, Issue 3, 1 November 2019, Pages 471–473, https://doi.org/10.1373/jalm.2018.028373

[3] Gohel D, Moog S (2024). _officer: Manipulation of Microsoft Word and PowerPoint Documents_. R package version 0.6.6, <https://CRAN.R-project.org/package=officer>.

[4] R Core Team (2024). _R: A Language and Environment for Statistical Computing_. R Foundation for Statistical Computing, Vienna, Austria. <https://www.R-project.org/>.

[5] Gohel D, Skintzos P (2024). _flextable: Functions for Tabular Reporting_. R package version 0.9.6, <https://CRAN.R-project.org/package=flextable>.

[6] H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

[7] Wang S, Ye S, Anderson KM, Zhang Y (2020). "r2rtf-an R Package to Produce Rich Text Format (RTF) Tables and Figures." In _PharmaSUG_. <https://www.pharmasug.org/proceedings/2020/DV/PharmaSUG-2020-DV-198.pdf>.

[8] Schaffer ME (2020). _rtf: Rich Text Format (RTF) Output_. R package version 0.4-14.1, <https://CRAN.R-project.org/package=rtf>.

[9] Xie Y (2024). _knitr: A General-Purpose Package for Dynamic Report Generation in R_. R package version 1.46, <https://yihui.org/knitr/>.

[10] Allaire J, Xie Y, Dervieux C, McPherson J, Luraschi J, Ushey K, Atkins A, Wickham H, Cheng J, Chang W, Iannone R (2024). _rmarkdown: Dynamic Documents for R_. R package version 2.29, <https://github.com/rstudio/rmarkdown>.

[11] S. Mati et al., "Demystifying Knitr Package: Essential Recipes and Easy Steps for Adding Knit-Engines in R," 2024 12th International Symposium on Digital Forensics and Security (ISDFS), San Antonio, TX, USA, 2024, pp. 01-06, doi: 10.1109/ISDFS60797.2024.10527232.

[12] Allaire, J., Teague, C., Scheidegger, C., Xie, Y., Dervieux, C., & Woodhull, G. (2025). Quarto (Version 1.8) [Computer software]. https://doi.org/10.5281/zenodo.5960048

[13] Wickham, H., Mine Cetinkaya-Rundel, & Grolemund, G. (2023). R for data science: Import, tidy, transform, visualize, and model data (2nd ed.). O'Reilly Media.

[14] Github Repository for accessing all R scripts. <https://github.com/ojaswini-oak/Rcode>.

[15] <https://github.com/ojaswini-oak/Rcode/blob/main/single_table.R>.

[16] <https://github.com/ojaswini-oak/Rcode/blob/main/single_plot.R>.

[17] <https://github.com/ojaswini-oak/Rcode/blob/main/multiple_tables.R>.

[18] <https://github.com/ojaswini-oak/Rcode/blob/main/multiple_sections.R>.

[19] <https://github.com/ojaswini-oak/Rcode/blob/main/multiple_pages.R>
[20] <https://github.com/ojaswini-oak/Rcode/blob/main/multiple_documents.R>
[21] <https://github.com/ojaswini-oak/Rcode/blob/main/custom_layout.R>.
[22] <https://github.com/ojaswini-oak/Rcode/blob/main/demography.R>.
[23] <https://github.com/ojaswini-oak/Rcode/blob/main/template_based.R>
[24] <https://ardata-fr.github.io/officeverse/>
[25] <https://davidgohel.github.io/officer/>