Automation of Publications in Official Statistics using R

Guido Schulz (guido.schulz@destatis.de)
Federal Statistical Office of Germany (Destatis)

ABSTRACT

A key task of official statistical authorities is to collect and disseminate indicators periodically. Automation using a wide range of R packages bears massive potential to cut down the resources necessary for the creation of publications. Furthermore, automation in R has the potential to improve transparency, punctuality and coherence of statistical products. The dynamic reporting engine knitr in particular allows for an efficient combination of R's functionalities of data retrieval, data manipulation and customizable plotting on the one hand, and the layout and typesetting flexibility of LaTex or other markup languages on the other. This allows official statistical authorities to produce either ready-to-print PDFs or interactive websites while adhering to their corporate design requirements. Furthermore, dynamic reporting makes it possible to update periodic publications automatically. A work in progress example of automated statistical country profiles - a product the German Federal Statistical Office regularly publishes based on a wide range of official international sources - will be presented to illustrate both advantages and challenges in the practical use of dynamic reporting using R and knitr in particular.

Keywords: *R, Automation, Dynamic Reporting, Official Statistics, knitr*
JEL Classification: *Y10, Y50*

1. INTRODUCTION

A key task of official statistical authorities is to collect and disseminate indicators periodically. Besides the publication of statistics in public databases, one of the most common channels for dissemination is the publication of printed or online thematic reports. Notwithstanding all the differences between official statistical authorities and their respective publication requirements, it is fair to claim that all face the common challenge of ensuring the workflow of publication production functions in a reliable and resource efficient way. Traditionally, publication production workflows depend on specific colleagues scattered over a range of departments, different software (licenses) need to be employed at different stages of production and the appearance of publications has to be manually adjusted to specific corporate design requirements. Such complex workflows demand not only a high level of coordination and planning, but they are highly labour intensive. Furthermore, bottlenecks emerge during the process quite often, putting timeliness or even the quality of the product at

stake. Many publication production workflows include individual automated operations, but the complexity of workflows that mix different software - and GUI-only software in particular - quickly limits the degree of automation potentially achievable. Thus, if the goal is to make periodical production of statistical reports less labour intensive, less error-prone, more reliable and more transparent, a workflow must be established that allows for the highest possible degree of automation. While streamlining tasks and workflows should naturally be the goal of any institution, EU national statistical institutes are obliged to "systematically and regularly identify strengths and weaknesses to continuously improve process and product quality" (Eurostat, 2011), as it is set out in the principles of the European Statistics Code of Practice.

With this paper, I seek to contribute to this objective by proposing a highly customizable workflow for report production in R that addresses all of the aforementioned problems. The dynamic reporting engine `knitr` (Xie, 2014; 2015; 2017c) will be at the core of the proposed workflow. The paper is divided into four parts. First, I am going to argue for the introduction of `knitr`, explain its general purpose and main functionalities that make it fundamental for automation in R. With that knowledge, I can then propose a generic, and therefore highly flexible architecture of a `knitr` based workflow that incorporates a wide range of other R packages. The generic workflow will consequently be translated into a concrete real-life example, where practical strengths and difficulties become apparent. Finally, I will conclude with a short summary and name remaining challenges for a full implementation of the proposed workflow.

2. KNITR AND ITS POTENTIAL FOR AUTOMATING PUBLICATIONS

Before embarking on an appraisal of `knitr`'s potential for automating the production of publications, I want to briefly explain what `knitr` is. `Knitr` is an R package written and maintained by Yihui Xie, that provides a "general-purpose tool for dynamic report generation in R using literate programming techniques" (Xie 2017c). The paradigm of literate programming as formulated by (Knuth, 1984) asks programmers to "mix the source code and documentation [or narrative] together, [so that] we can either extract the source code out […] or execute the code to get the compiled results" (Xie, 2015, p.1). In our case of interest, `knitr` serves as a software package that compiles softcoded or dynamic source code into numeric, literal or graphical output, creating a publication ready report. `Knitr`'s default engine for the input source code is R, but amazingly it also accepts other programming

languages including Python, Awk, Ruby, Bash, Perl, SAS, Scala, CoffeeScript and others. In this paper, I want to stick to the simple case of using `knitr`'s default engine R. Outputted reports can take on a variety of formats - DOCX, HTML and PDF being the most important ones. Thus, `knitr` is not just a convenient dynamic reporting engine, but given the wide range of accepted input programming languages and output formats, it is also highly flexible.

Introducing `knitr` to the workflow of statistical report production and combining it with other R packages bears massive potential on various levels. Potential improvements in both production process and product quality include:

- Automated data retrieval, data processing and data presentation/ visualisation together significantly reduce the labour input required.
- Replacing arduous manual hardcoding with softcoding minimizes errors.
- Report production scripts that are run automatically on specific time points foster adherence to dissemination time schedules.
- Automation enhances coherence and comparability in both design and content.
- Providing reproducible code (cf. Sandve et al., 2013) that covers the entire workflow vastly improves transparency – both inhouse and beyond.
- Simplifying report production workflows to just one or two programming languages boils down the necessary skills for practitioners.
- `Knitr` based workflows can reduce or even eliminate software license expenditures all together.

This broad list of advantages should provide enough motivation to consider incorporating `knitr` into statistical report production workflows. Now, the task is to show what exactly a technical implementation should entail for these advantages to unfold.

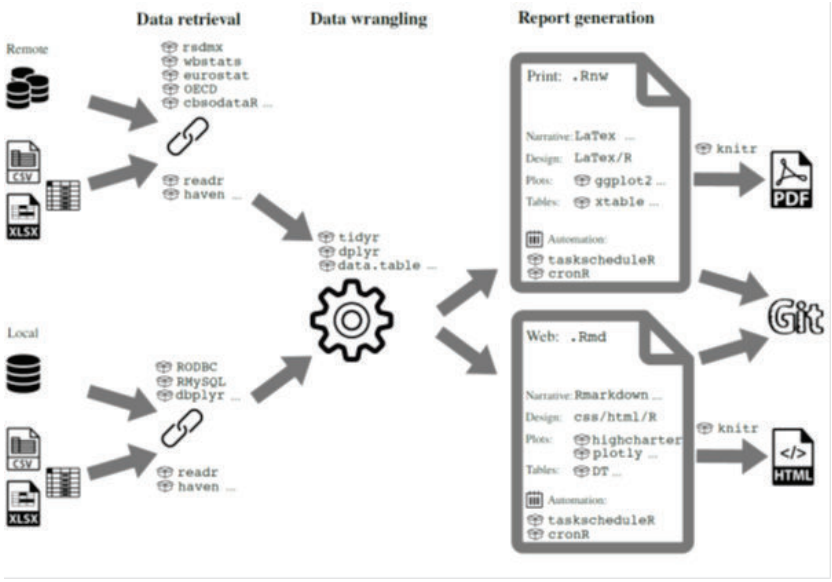
3. GENERIC ARCHITECTURE OF A KNITR BASED WORKFLOW

In this section I want to propose a generic architecture of a `knitr` based workflow to automate statistical report production. Alongside `knitr`, a number of other R packages will be recommended for integration into the workflow. Naturally, they may be replaced or supplemented according to the reader's specific needs and publication requirements.

Figure 1 depicts the scheme of a generic workflow architecture. The workflow consists of a sequence of three main operations: data retrieval, data wrangling and report generation. Each step of the flow will now be explained consecutively.

Scheme of a generic knitr based workflow for automated publications

Figure 1



3.1. Data retrieval

The raw tabular data you want to process for your report may be stored in remote or local databases or individual files. In case you want to connect to remote databases run by institutions offering official statistics, the chances are the R community will provide a handy package to do the job for you. Packages such as `wbstats` (Piburn, 2016), `eurostat` (Lahti et al., 2017), `OECD` (Persson, 2016) allow for - as the names imply - easy access to the vast databases of these intergovernmental organizations. A few national statistical institutes, such as the Dutch CBS, offer their own packages (e.g. `cbsodataR` (De Jonge, 2016)) to allow end-users to connect to their public database. For users wanting to connect to multiple data providers using the same syntax, the `rsdmx` (Blondel, 2017) package is an attractive alternative. It enables a connection to data providers that offer an embedded web-service interface to SDMX, including many international and national institutions. Connecting to local databases by contrast requires procedures from another

set of packages (RODBC (Ripley and Lapsley, 2017), RMySQL (Ooms et al., 2017), dbplyr (Wickham, 2017), etc.) from which you should pick according to the nature of your local database-management system. Importing remotely or locally stored individual rectangular data files (.csv, .tsv, .xlsx, etc.) can be conveniently done with readr. For reading remote or local data from foreign statistical packages (SPSS, Stata, SAS) the haven package provides the best functionalities.

3.2. Data wrangling

Once the data has been successfully retrieved, the next big task is to tidy, process and manipulate the data to make it fit for literal or visual representation within the report. Currently, the most powerful yet user friendly packages for data wrangling arguably are tidyr (Wickham and Henry, 2017), dplyr (Wickham, 2017) and data.table (Dowle and Srinivasan, 2017).

3.3. Report generation

Finally, in this last step knitr enters the workflow and the above promised advantages can take effect. In case you wish to produce a specifically designed report in PDF format, the source document containing a mixture of code and narrative should be an R Sweave (Rnw) file. Rnw files only accept LaTex as an input language for the narrative.

The flexibility of LaTex and its packages should allow you to tune the page layout and general typography according to your specific needs. In order to ensure coherence with the design - i.e. colours, spacings, fonts - between plots and the rest of your report, you should take advantage of the flexibility of ggplot2 (Wickham and Chang 2016, 2) theming, its extensions (ggforce (Pedersen, 2016), ggtheme (Arnold, 2017), etc.) and the packages that allow for general plot customization (gridExtra (Auguie, 2017), gtable (Wickham, 2016), extrafont (Chang, 2014), etc.). For table customization I recommend to capitalise on one of LaTex's powerful default features for table construction and extending them with LaTex packages such as tabu (Chervet, 2010), booktabs (Els and Fear, 2016), multirow (Van Oostrum and Leichter, 2016). You may consider sourcing out all the design customization code by creating your own private (corporate) design packages in either R, LaTex or both. This not only makes your Rnw document more compact and easier to read, it also allows you to extend potential design adjustments to *all* of your publications effortlessly by changing only the code in your design package.

Once all the design work has been done, you should embed the R scripts that perform the previously explained data retrieval and data wrangling

steps. With the clean and processed data loaded into R, you can finally fill the report with content by creating dynamic texts, plots (`ggplot2`) and tables (`xtable` (Dahl, 2016), `kableExtra` (Zhu, 2017)). Interestingly, dynamic reporting seems to rely on this – compared to conventional, manual report production - reversed order of layout/design tasks and content creation.

As a final step, `knitr` compiles the Rnw and produces - if everything goes well - a publication-ready PDF.[1] Since the report source document contains dynamic code performing data retrieval, data wrangling and content creation, it is possible to update the report simply by letting knitr compile the document source code anew. Even this task of updating the reports can be automated, which is particularly convenient for periodical publications. The R packages `taskscheduleR` (Wijffels, 2017b) for Windows and `cronR` (Wijffels, 2017a) for unix-alike systems allow users to schedule the compilation of an updated report at predefined points in time.

Since the report source document contains the entire workflow in code, the project becomes not only comprehensively documented, but (potentially) fully reproducible, too. Publishing the report source document e.g. on GitHub alongside the report, enhances transparency and - assuming report source document actually is fully reproducible - encourages interested readers to use the data themselves.

Looking at the generic workflow (Figure 1), we realize that we have worked our way through almost the entire scheme. The only part that remains to be explained is the case of an R markdown (Rmd) report source document, which compiles to a report in HTML.[2]

The general proceeding remains the same as for the discussed R Sweave (Rnw) source document, only the languages used for narrative and design, as well as the recommended R packages for plot and table creation change. The narrative is written in markdown or plain HTML, while design and general appearance need to be specified in CSS and HTML. In contrast to PDF reports, HTML reports allow for elements to be interactive, such as zoomable graphics, sortable tables or foldable contents. Interactivity – when applied sensibly – enhances the user experience and offers many advantages over static representations (Kirk, 2016, pp.223). This is why, for HTML reports, I recommend to using interactive report templates (e.g. `rmdformats` (Barnier, 2017)), interactive plots (e.g. `plotly` (Sievert et al., 2017), `highcharter` (Kunst, 2017), `ggiraph` (Gohel, 2017)) and

1. For a technical description of how knitr compiles the report source document, see Xie (2015).

2. To be precise, it is not knitr, but pandoc that carries out the very last compilation to HTML. Pandoc is a universal document converter that enables knitr to create such a wide variety of output file types from Rmd source documents (MacFarlane, 2016).

interactive tables (e.g. `DT` (Xie, 2016)). Most of these R packages are simple yet incredibly convenient wrappers for their respective JavaScript libraries. In addition, `htmlwidgets` (Vaidyanathan et al., 2017) not only helps with the embedding of widgets into R Markdown documents, it also allows users to develop their own widgets to seamlessly bridge R and JavaScript. Generally speaking, `knitr` is capable of combining R and JavaScript applications such as D3.js, since it is possible to "write anything to the output, including JavaScript" (Xie, 2017b).

Arguably, having to code separate report source documents for PDF and HTML reports is not very convenient. Despite the general possibility to use the very same R Markdown source document to compile both report formats, this approach will inevitably reduce customizability and coherence of design and layout.

4. PRACTICAL EXAMPLE

4.1. Extending and renewing statistical country profiles

As part of its product portfolio on international statistics, the Federal Statistical Office of Germany publishes so called statistical country profiles for the members of the G20 "based on official international statistics as published by UN organisations, World Bank, OECD, IMF, etc" (Federal Statistical Office of Germany, 2017). The country profiles published in English and German cover a wide range of topics including economy, finance, demography, health, environment and others. The data is gathered from more than 15 different sources. Until now, seven page PDF format country profiles were produced in a semi-automated, yet not fully-automatable workflow, which is only partially recordable. Updates require a considerable amount of manual work, particularly when it comes to updating plots. Four expensive proprietary software packages are used along the product in process and data has to flow manually through different graphical user interfaces.

At the Federal Statistical Office in Germany it has long been our plan to extend and renew our country profile products. Our goal is to

- broaden the set of country profiles beyond the G20, hopefully covering all 193 members states of the United Nations,
- offer a HTML web based interactive version of the country profile in addition to the static PDF, and
- provide updated versions of all profiles as soon as new data gets published.

It stands to reason that the desired goal cannot be achieved without a considerable increase in the level of automation. Evidently, adhering to

the old workflow would require absurd amounts of manual labour. Hence, we took the extension and renewal of the country profiles as a first trial to practically implement an automated `knitr` workflow. Work on this project is ongoing and still requires some tweaking, yet I want to briefly explain how I concretized the generic workflow architecture proposed above and what challenges occurred in practice.

4.2. Concrete workflow and challenges faced

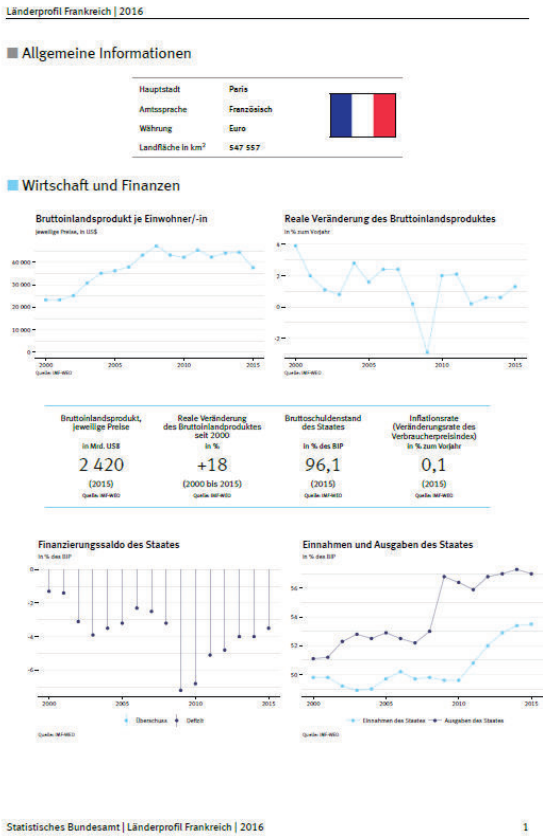
To produce a PDF and HTML country profile in each English and German, four separate source documents were created. A master script calls `knitr` to compile all of these four source documents resulting in the four desired versions of the country profile. This master script itself was put in a loop that iterates through all 193 member states of the United Nations. In order to keep the source documents as simple and short as possible, R codes for data retrieval, plot and table designs and plot generation were separated out into extra R files.

All data necessary for the country profiles could conveniently be retrieved with RODBC from a regularly updated in-house local MySQL database. The already relatively clean data was processed with `dplyr`. As it was proposed above for PDF reports, plots were produced with `ggplot2` plus extending packages and tables were set using `xtable`. The HTML reports were generated on the basis an `rmdformats` template. Since project progress of the HTML version lags behind, a lot of CSS tweaking to enhance the HTML report design still needs to be done. However, interactive plots and tables could already be produced with `highcharter` and `DT`. Unfortunately, scheduled automatic report compilation has not been implemented yet. Figure 2 and Figure 3 show, for the example of France, clippings of the German language PDF and HTML prototype versions of the new country profile.[1]

1. The new country profiles have not been officially published yet. Prototype versions including Rnw and Rmd report source documents are available upon request from the author (guido.schulz@destatis.de).

Clipping of the PDF country profile

Figure 2



Länderprofil Frankreich | 2016

Allgemeine Informationen

Hauptstadt	Paris
Amtssprache	Französisch
Währung	Euro
Landfläche in km²	547 557

Wirtschaft und Finanzen

Bruttoinlandsprodukt je Einwohner/-in

Reale Veränderung des Bruttoinlandsproduktes

Bruttoinlandsprodukt, jeweilige Preise	Reale Veränderung des Bruttoinlandsproduktes seit 2000	Bruttoschuldenstand des Staates	Inflationsrate (Veränderungsrate des Verbraucherpreisindex)
in Mrd. US$	in %	in % des BIP	in % zum Vorjahr
2 420	+18	96,1	0,1
(2015)	(2000 bis 2015)	(2015)	(2015)

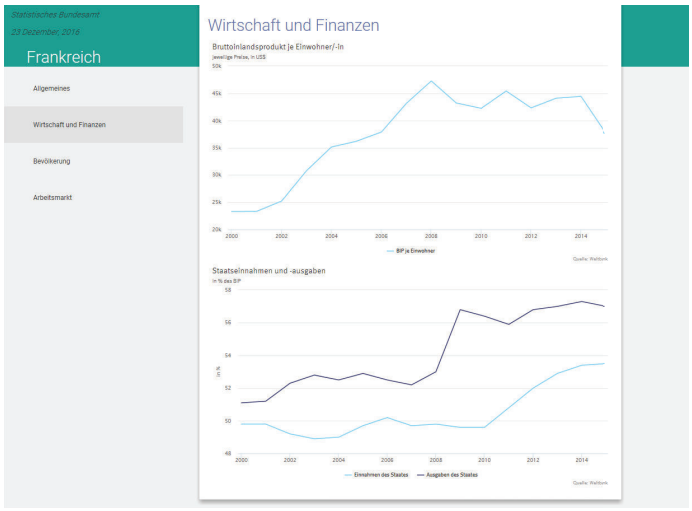
Finanzierungssaldo des Staates

Einnahmen und Ausgaben des Staates

Statistisches Bundesamt | Länderprofil Frankreich | 2016 1

Clipping of the HTML country profile

Figure 3



Unsurprisingly, many practical – mostly design-related - obstacles have paved the project so far. One central challenge was to softcode plot or table specifications flexibly enough, to cope with values of starkly differing size while adhering to corporate design requirements. The problem becomes palpable when thinking about the varying magnitudes of economic indicators for say, China and Lesotho. Satisfying all corporate design requirements, such as specific spacing in plots, guaranteeing font size coherence between plots and text or embedding a UTF8 encoded True Type font in plots required seemingly endless fiddling. The usual LaTex specific package option clashes (e.g. `xcolor` (Kern, 2016) and `pgf` (Tantau and Feuersänger, 2015)) came on top. Luckily, for all PDF report related problems more or less complicated solutions could be found, not least with the help of Stack Overflow. In contrast, a number of obstacles still need to be overcome for the HTML report. Unfortunately, to this day I could not find any decent practical example or tutorial online of how a `knitr` compiled HTML report may be properly embedded into a customized website. The recently released `blogdown` (Hill, Xie, and Amber, 2017; Xie, 2017a) package which particularly serves for website creation with R Markdown promises big advances on this front.

5. SUMMARY AND CHALLENGES AHEAD

This paper sought to show that it is both desirable and feasible to streamline the workflow of statistical publication production using the dynamic reporting engine `knitr`, potentially achieving full automation. R and `knitr` in particular promise labour savings, improved output quality, enhanced transparency and potentially cost reductions. A generic, thus highly customizable, architecture of a `knitr` based workflow was proposed to illustrate what a technical implementation of an automated workflow could look like. The three central operations of the workflow, namely data retrieval, data wrangling and report generation were described in detail, enabling the reader to adopt and reproduce each step according to the reader's specific needs. Finally, I presented the concrete workflow of an on-going project of publication automation - the remaking and extension of so called statistical country profiles, a product the Federal Statistical Office of Germany regularly publishes based on a wide range of official international sources. By means of this concrete example, I pointed out a variety of practical pitfalls that commonly occur in implementation.

Admittedly, there are preconditions and limits to the workflow proposed in this paper, and they should not go unmentioned. First of all, not every variation of the generic workflow was tested, hence I might be unaware of certain problems that occur. I also acknowledge the fact that the CRAN universe is so vast and evolves so rapidly that other R packages might suit the reader's needs better than ones suggested above. As with any process automation, the implementation of the `knitr` based workflow certainly depends on preliminary work. Just to name a few: all software (dependencies) need to be set up properly for all people involved in the project, data stored in databases need to be kept tidy enough to be processed automatically, a corporate design package potentially needs to be written beforehand or web content management systems need to be configured to correctly integrate the produced reports.

Though the most important preliminary task to be tackled is arguably also the biggest challenge – convince your superiors and colleagues to establish and learn a new workflow for publication production. Probably, your organisation's actual workflow is based on a well calibrated ecosystem of (proprietary) software and a well-coordinated personal distribution of tasks and skills. Yet, I hope the impressive list of advantages of a `knitr` based workflow may serve as a good argument in favour of change. Starting to implement a new workflow for just one specific publication that also does not involve too many people and using the project to exemplify `knitr`'s

advantages might be a promising way to embark on a wider transition towards R and `knitr`. Even partial implementations of the proposed workflow may already bring significant benefits. And it is worth recalling that there are good examples to follow. Recently, the UK Government and its Office for National Statistics (ONS) started to migrate from the proprietary SAS software to R (Sellors, 2017a; 2017b) and embarked on a project to speed up the production of official statistics for publications using `knitr` (Government Digital Service, 2017). Statistics Netherlands (CBS) has already managed to largely incorporate R into their workflows, though admittedly their dependence on proprietary statistical software before the introduction of R was comparatively small (Van Der Loo, 2012).

Despite all the challenges and practical pitfalls of a `knitr` based publication production workflow, I hope this paper inspired and motivated the reader for a transition. In any case, the proposed workflow is meant a starting point for a discussion on how modern statistical report production can and should look like – an invitation to have an exchange or even cooperation on how R and `knitr` may be promoted in the distribution of official statistics.

REFERENCES

1. **Arnold, J. B.**, 2017, *ggthemes: Extra Themes, Scales and Geoms for 'Ggplot2'* (version 3.4.0). https://cran.r-project.org/package=ggthemes.
2. **Auguie, B**., 2017, *GridExtra: Miscellaneous Functions for 'Grid' Graphics* (version 2.3). https://cran.r-project.org/package=gridExtra.
3. **Barnier, J**., 2017, *rmdformats: HTML Output Formats and Templates for 'Rmarkdown' Documents* (version 0.3.3). https://CRAN.R-project.org/package=rmdformats.
4. **Blondel, E.**, 2017, *rsdmx: Tools for Reading SDMX Data and Metadata* (version 0.5-9). https://cran.r-project.org/package=rsdmx.
5. **Chang, W.**, 2014, *extrafont: Tools for Using Fonts* (version 0.17). https://cran.r-project.org/package=extrafont.
6. **Chervet, F.**, 2010, *tabu: Flexible LaTeX Tabulars* (version 2.8). https://ctan.org/pkg/tabu?lang=en.
7. **Dahl, D. B.**, 2016, *xtable: Export Tables to LaTeX or HTML* (version 1.8-2). https://cran.r-project.org/package=xtable.
8. **De Jonge, E.**, 2016, *cbsodataR: Statistics Netherlands (CBS) Open Data API Client* (version 0.2.1). https://cran.r-project.org/package=cbsodataR.
9. **Dowle, M.**, **A. Srinivasan,** 2017, *data.table: Extension of 'Data.Frame'* (version 1.10.4-1). https://cran.r-project.org/package=data.table.
10. **Els, D.**, and **S. Fear.**, 2016, *booktabs: Publication Quality Tables in LaTeX* (version 1.618033). https://ctan.org/pkg/booktabs?lang=en.
11. **Eurostat**, 2011, *European Statistics Code of Practice (Revised Edition 2011)*. http://ec.europa.eu/eurostat/documents/3859598/5921861/KS-32-11-955-EN.PDF/5fa1ebc6-90bb-43fa-888f-dde032471e15.
12. **Federal Statistical Office of Germany**, 2017, *International Data - G20: Country Profiles*. https://www.destatis.de/EN/Publications/Specialized/InternationalData/CountryProfiles/Countryprofile.html.

13. **Gohel, D.**, 2017, *ggiraph: Make 'Ggplot2' Graphics Interactive* (version 0.4.1). https://cran.r-project.org/package=ggiraph.
14. **Government Digital Service**, 2017, *Reproducible Analytical Pipelines*. GDS Data Blog. https://gdsdata.blog.gov.uk/2017/03/27/reproducible-analytical-pipeline/.
15. **Hill, A. P., Y. Xie**, and **T. Amber**, 2017, *blogdown: Creating Websites with R Markdown*. https://bookdown.org/yihui/blogdown/.
16. **Kern, U.**, 2016, *xcolor: Driver-Independent Color Extensions for LaTeX and PdfLaTeX* (version 2.12). https://ctan.org/pkg/xcolor?lang=en.
17. **Kirk, A.**, 2016, *Data Visualisation: A Handbook for Data Driven Design*. Los Angeles London New Delhi Singapore Washington DC Melbourne: SAGE Publications Ltd.
18. **Knuth, D. E.**, 1984, *Literate Programming*. The Computer Journal 27 (2): p.97-111.
19. **Kunst, J**., 2017, *highcharter: A Wrapper for the 'Highcharts' Library* (version 0.5.0). https://cran.r-project.org/package=highcharter.
20. **Lahti, L., J. Huovari, M. Kainu,** and **P. Biecek**, 2017, *eurostat: Tools for Eurostat Open Data* (version 3.1.5). https://cran.r-project.org/package=eurostat.
21. **MacFarlane, J**., 2016, *Pandoc - A Universal Document Converter* (Version 1.19.2.1). https://pandoc.org/.
22. **Ooms, J., D. James, S. DebRoy, H. Wickham**, and **J. Horner**, 2017, *RMySQL: Database Interface and 'MySQL' Driver for R* (version 0.10.13). https://cran.r-project.org/package=RMySQL.
23. **Pedersen, T. L.**, 2016, *ggforce: Accelerating 'GGplot2'* (version 0.1.1). https://cran.r-project.org/package=ggforce.
24. **Persson, E**., 2016, *OECD: Search and Extract Data from the OECD* (version 0.2.2). https://cran.r-project.org/package=OECD.
25. **Piburn, J.**, 2016, *wbstats: Programmatic Access to Data and Statistics from the World Bank API* (version 0.1.1). https://cran.r-project.org/package=wbstats.
26. **Ripley, B.**, and **M. Lapsley,** 2017, *RODBC: ODBC Database Access* (version 1.3-15). https://cran.r-project.org/package=RODBC.
27. **Sandve, G. K., A. Nekrutenko, J. Taylor**, and **E. Hovig,** 2013, *Ten Simple Rules for Reproducible Computational Research*. PLOS Computational Biology 9 (10): e1003285.
28. **Sellors, M.**, 2017a, *Goodbye SAS, Hello R: Why You Need to Make the Switch and How Mango Can Help*. Mango Solutions. https://www.mango-solutions.com/blog/goodbye-sas-hello-r-why-you-need-to-make-the-switch-and-how-mango-can-help.
29. ***, 2017b, *Production R at ONS*. Mango Solutions. https://www.mango-solutions.com/blog/production-r-at-ons.
30. **Sievert, C., C. Parmer, T. Hocking, S. Chamberlain**, **K. Ram, M. Corvellec**, and **P. Despouy**, 2017, *plotly: Create Interactive Web Graphics via 'Plotly.Js'* (version 4.7.1). https://cran.r-project.org/package=plotly.
31. **Tantau, T.** and **C. Feuersänger**, 2015, *pgf: Create PostScript and PDF Graphics in TeX* (version 3.0.1a). https://ctan.org/pkg/pgf?lang=en.
32. **Vaidyanathan, R., Y. Xie, J. J. Allaire, J. Cheng**, and **K. Russell,** 2017, *htmlwidgets: HTML Widgets for R* (version 0.9). https://cran.r-project.org/package=htmlwidgets.
33. **Van Der Loo, M.**, 2012, *The Introduction and Use of R Software at Statistics Netherlands*. In: Proceedings of the Third International Conference of Establishment Surveys (CD-ROM)). Montréal, Canada: American Statistical Association.
34. **Van Oostrum, P.** and **J. Leichter,** 2016, *multirow: Create Tabular Cells Spanning Multiple Rows* (version 2.2). https://ctan.org/pkg/multirow.
35. **Wickham, H.**, 2016, *gtable: Arrange 'Grobs' in Tables* (version 0.2.0). https://cran.r-project.org/package=gtable.
36. ***, 2017,, *dbplyr: A 'dplyr' Back End for Databases* (version 1.1.0). https://cran.r-project.org/package=dbplyr.

37. **Wickham, H.** and **W. Chang,** 2016, *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics* (version 2.2.1). https://cran.r-project.org/package=ggplot2.
38. **Wickham, H.** and **L. Henry**, 2017, *tidyr: Easily Tidy Data with 'Spread()' and 'Gather()' Functions* (version 0.7.1). https://cran.r-project.org/package=tidyr.
39. **Wijffels, J.**, 2017a, *cronR: Schedule R Scripts and Processes with the 'Cron' Job Scheduler* (version 0.3.0). https://cran.r-project.org/package=cronR.
40. ***, 2017b, *taskscheduleR: Schedule R Scripts and Processes with the Windows Task Scheduler* (version 1.0). https://cran.r-project.org/package=taskscheduleR.
41. **Xie, Y.**, 2014, *knitr: A Comprehensive Tool for Reproducible Research in R*. In: Implementing Reproducible Research, edited by V. Stodden, F. Leisch, and R. D. Peng. The R Series. Boca Raton: CRC Press.
42. ***, 2015, *Dynamic Documents with R and Knitr, Second Edition*. 2 edition. Boca Raton London New York: Chapman and Hall/CRC.
43. ***, 2016, *DT: A Wrapper of the JavaScript Library 'DataTables'* (version 0.2). https://cran.r-project.org/package=DT.
44. ***, 2017a, *blogdown: Create Blogs and Websites with R Markdown* (version 0.1). https://CRAN.R-project.org/package=blogdown.
45. *** , 2017b, knitr Documentation. *knitr - Elegant, Flexible, and Fast Dynamic Report Generation with R*. https://yihui.name/knitr/.
46. ***, 2017c, *knitr: A General-Purpose Package for Dynamic Report Generation in R* (version 1.17). https://cran.r-project.org/package=knitr.
47. **Zhu, H.**, 2017, *kableExtra: Construct Complex Table with 'kable' and Pipe Syntax* (version 0.5.2). https://cran.r-project.org/package=kableExtra.