The R Journal

Volume 16/1, March 2024

A peer-reviewed, open-access publication of the
R Foundation for Statistical Computing

Contents

Prospective authors will find detailed and up-to-date submission instructions on the Journal's homepage.

R Journal Homepage:
http://journal.r-project.org/

Email of editors and editorial board:
r-journal@R-project.org

The R Journal is indexed/abstracted by EBSCO, DOAJ, Thomson Reuters.

Editorial

by Mark P.J. van der Loo

On behalf of the editorial board, I am pleased to present Volume 16 Issue 1 of the R Journal.

We would like to welcome our new Associate Editors Jouni Helseke, Christoph Sax, Thomas Fung, Wenjie Wang, Matthias Templ, Thiyanga Talagala, Xiaoqian Wang, Romain Lesur, and Ivan Svetunkov to the editorial team.

We also express our gratitude to Simon Urbanek, who worked as editor-in-chief, and will stay on as executive editor. Simon has not only worked as an editor of the journal but is also contributing to improving the submission infrastructure. The articles in this issue have been carefully copy edited by Adam Bartonicek and Harriet Mason. We thank Mitchell O'Hara-Wild for technical editing work on this issue.

We are deeply saddened that the publication of this issue also marks the passing of Friedrich 'Fritz' Leisch. Fritz was one of the founding fathers of *R News*, the peer-reviewed publication that would turn into the journal that you are reading today. The fact that *R News* started with just two people, and is now run by more than 30 volunteers is a testimony to his impact within the R community. As a member of the R Core Team, Fritz also committed `Sweave` to R, which must be seen as a visionary and pioneering step towards the reproducible research workflow for which R is now famous and that supports the publication of this Journal. For these and all his other contributions, we will forever be grateful. On behalf of the R Journal team, we extend our deepest condolences to his family, his friends, and others close to him.

To honour Fritz, the first paper in this issue by Bettina Grün, Kurt Hornik, Torsten Hothorn, Theresa Scharl, and Achim Zeilis, commemorates Fritz' work and life.

In this issue

News from CRAN, the R Foundation and Bioconductor are included in this issue.

This issue features 10 contributed research articles the majority of which relate to R packages on a diverse range of topics. All packages are available on CRAN. Supplementary material with fully reproducible code is available for download from the Journal website. Topics covered in this issue are the following.

Time Series, Stochastic Processes

- **bootCT**: An R Package for Bootstrap Cointegration Tests in ARDL Models
- **GenMarkov**: Modeling Generalized Multivariate Markov Chains in R
- **nortsTest**: An R Package for Assessing Normality of Stationary Processes

Survival Analyses

- **ebmstate**: An R Package For Disease Progression Analysis Under Empirical Bayes Cox Models
- Fitting a Quantile Regression Model for Residual Life with the R Package **qris**

Statistical Inference

- Prediction, Bootstrapping and Monte Carlo Analyses Based on Linear Mixed Models with **QAPE** 2.0 Package
- Bayesian Model Selection with Latent Group-Based Effects and Variances with the R Package **slgf**
- **BMRMM**: An R Package for Bayesian Markov (Renewal) Mixed Models

Programming and applications

- **text2sdg**: An R Package to Monitor Sustainable Development Goals from Text
- **shinymgr**: A Framework for Building, Managing, and Stitching Shiny Modules into Reproducible Workflows

Mark P.J. van der Loo
Statistics Netherlands and Leiden University

https://journal.r-project.org
r-journal@r-project.org

Remembering Friedrich "Fritz" Leisch

by Bettina Grün, Kurt Hornik, Torsten Hothorn, Theresa Scharl, and Achim Zeileis

Abstract This article remembers our friend and colleague Fritz Leisch (1968–2024) who sadly died earlier this year. Many of the readers of The R Journal will know Fritz as a member of the R Core Team and for many of his contributions to the R community. For us, the co-authors of this article, he was an important companion on our journey with the R project and other scientific endeavours over the years. In the following, we provide a brief synopsis of his career, present his key contributions to the R project and to the scientific community more generally, acknowledge his academic service, and highlight his teaching and mentoring achievements.

1 Career

Friedrich Leisch (see Figure 1) was born 1968 in Vienna (Austria) and died after serious illness in 2024 in Vienna. Everyone called him Fritz.

Figure 1: Fritz Leisch at his inaugural lecture at BOKU in 2011. Source: BOKU.

Starting in 1987, Fritz studied Applied Mathematics at Technische Universität Wien (TU Wien), earning his master's degree (Dipl.-Ing.) in 1993. Subsequently, he joined the Department of Statistics and Probability Theory at TU Wien as an assistant professor which he continued to be, with short intermissions, until 2006. During this time he also defended his doctoral thesis in Applied Mathematics (Dr.techn.) in 1999 and earned his habilitation (venia docendi) in Statistics in 2005.

In 1995, he visited the Knowledge-Based Engineering Systems Group at the University of South-Australia in Adelaide on a Kurt Gödel scholarship for postgraduate studies. From 1997 to 2004 he was a member of the SFB project "Adaptive Information Systems and Modeling in Economics and Management Science", coordinated at Wirtschaftsuniversität Wien (WU Wien). From 2002 to 2003 he was assistant professor at the Department of Statistics and Decision Support Systems, Universität Wien.

In 2006 Fritz moved to Munich, Germany, to become a professor for computational statistics at the Department of Statistics, Ludwig-Maximilians-Universität München (LMU), see Figure 2. He returned to Vienna in 2011 to join the BOKU University as head of the Institute of Statistics, see Figure 3.

2 Key contributions

Fritz' scientific contributions span an impressive range including theoretical and methodological work (especially in the field of clustering and finite mixture models) over software (mostly related to the R programming language) to applied work and cooperations (notably in marketing, biotechnology, and genomics, among many others). In the following sections we try to highlight his key contributions and scientific legacy.

Figure 2: Computational statistics group at LMU in 2007 (left to right): Sebastian Kaiser, Adrian Duffner, Manuel Eugster, Fritz Leisch. Source: Carolin Strobl.

Figure 3: Institute of Statistics at BOKU in 2022 (left to right, back to front): Johannes Laimighofer, Nur Banu Özcelik, Ursula Laa, Fritz Leisch, Bernhard Spangl, Gregor Laaha, Matthias Medl. Robert Wiedermann, Lena Ortega Menjivar, Theresa Scharl, Melati Avedis. Source: BOKU.

2.1 R Core & CRAN

During his stay in Australia, Fritz had learned about the existence of R. Back in Austria, he and Kurt started to explore this potentially good news more systematically. They soon stopped further work on a statistics toolbox they had developed for Octave (Eaton et al., 2024), and switched to R for their applied work, finding lots of room for further improvement, and thus sending polite emails with patches and more suggestions to Ross Ihaka and Robert Gentleman. Clearly these were acceptable in quality but too high in quantity, and it did not take very long that Ross and Robert gave Fritz and Kurt write access to the R sources (initially in CVS, then moved to SVN), and in 1997, they both officially became very early members of the R Core Team.

One of the main challenges then was that the functionality provided by R was rather limited. Contributed extensions for S were available from the Carnegie Mellon University Statlib S Archive[1],

[1]Unfortunately, the Statlib S Archive is currently not available anymore. A snapshot, including many of the actual source code files, is available on the Internet Archive at https://web.archive.org/web/20000815063825/http://lib.stat.cmu.edu/S/.

The Comprehensive R Archive Network

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for the R statistical package. Please use the CRAN site nearest to you to minimise network load.

CRAN contains:

- The R Frequently Asked Questions List
- The latest beta and alpha releases of R source, and binaries for some systems.
- R Documentation
- Information on the three R mailing lists
- Contributed R code: Additional toolboxes and data sets
- Pre-formatted Help Pages: Perl 5 is needed for building R help pages from the source files. If perl (version 5 or higher) is not installed on your system, you can always get pre-formatted help pages.
- Emacs speaks Statistics, an Emacs-Lisp interface to interactive statistical programming and data analysis languages, including: S dialects (such as R), LispStat dialects and SAS.

The Statlib S Archive is another source of useful code, as most S code runs under R.

---

wwwadmin@ci.tuwien.ac.at
Last modified: December 9, 1997 by Friedrich Leisch

Figure 4: Screenshot of the landing page of the CRAN master site at TU Wien on 1998-01-10, as last modified by Fritz on 1997-12-09. Source: Internet Archive.

and could typically be ported to R rather easily, but there was no mechanism for conveniently distributing or actually using these extensions. This fundamentally changed, when in 1997 Fritz and Kurt implemented the R package management system, using ideas from Debian's APT (advanced package tool, https://wiki.debian.org/AptCLI) they had successfully employed for managing their computer systems. They also set up the Comprehensive R Archive Network (CRAN, https://CRAN.R-project.org/, see also Hornik, 2012) as a means for redistributing R and its contributed extensions, and infrastructure for quality assurance of these extensions. These two contributions paved the way for the amazing growth and success of R through its wealth of high-quality contributed extensions. See https://stat.ethz.ch/pipermail/r-announce/1997/000001.html for the first announcement of CRAN, starting with 12 extension packages. Currently, there are more than 21,000. See Figure 4 for a screenshot[2] of the landing page of the CRAN master site at TU Wien, as last modified by Fritz on 1997-12-09.

The first SVN commit by Fritz is from 1997-10-02, the last from 2013-10-04. Overall, there are 651 commits by Fritz, mostly from the early years of R Core, and related to the R package management and CRAN mirror system, and the addition of the `Sweave` system (see Section 2.3 for more details).

2.2 DSC & useR! conferences

With establishing CRAN in Vienna at TU Wien, Fritz and Kurt laid the foundation for a special relationship between Vienna and R that they characterized as a story of "love and marriage" (Hornik and Leisch, 2002). In the decade after the creation of CRAN a number of seminal R-related meetings took place in Vienna, co-organized by Fritz as well as several of the co-authors of this paper.

The first workshop on "Distributed Statistical Computing" (DSC) took place from March 19-23, 1999, at TU Wien. The main motivations were bringing together the R Core Team for its first face-to-face meeting, discussing the roadmap for the release of R 1.0.0, as well as exploring potential synergies with other environments for statistical computing. There were around 30 participants and about 20 presentations, many of which were relatively short, leaving ample time for discussions (see Figure 5).

[2]This is from the earliest capture, from 1998-01-10, available on the Internet Archive at https://web.archive.org/web/19980110082558/http://www.ci.tuwien.ac.at/R/contents.html.

Figure 5: Discussions at DSC 1999 (top to bottom, left to right): Thomas Lumley, Fritz Leisch, Luke Tierney. Peter Dalgaard, Ross Ihaka, Paul Murrell. Brian Ripley, Martin Mächler, Robert Gentleman, Kurt Hornik. Source: Douglas Bates (DSC 1999 homepage).

Figure 6: Conference dinner at useR! 2006 (left to right): Fritz Leisch, Torsten Hothorn, Tim Hesterberg. Source: Carolin Strobl (useR! 2006 homepage).

Two more DSC workshops were organized at TU Wien in 2001 and 2003. While meetings focusing on R development issues (with the R Core Team and everyone else interested) were still an important part of these conferences, they also saw an increasing number of regular conference presentations on R packages and their different fields of application (e.g., establishing infrastructure for spatial data). In 2001 there were around 60 participants and about 30 presentations, most with corresponding papers in the online proceedings (Hornik and Leisch, 2001). In 2003 this increased to more than 150 participants and about 60 presentations, again with the majority in the online proceedings (Hornik et al., 2003).

The high demand for a platform, where R users from different fields could exchange ideas, prompted the creation of a new conference series called useR!. The first two installments again took place in Vienna in 2004 at TU Wien and in 2006 at WU Wien. Torsten Hothorn, David Meyer, and Achim Zeileis took the lead in the organization with support and advice from Fritz and Kurt in the background. An important contribution from the R Core Team at the useR! conferences were keynote lectures highlighting important developments, e.g., a keynote given by Fritz at useR! 2004 on S4 classes and methods. Both conferences continued the success of the earlier DSC workshops with the number of participants rising to more than 200 in 2004 and close to 350 in 2006. Similarly, the number of presentations grew to about 100 in 2004 and more than 150 in 2006.

In addition to the efforts initiated by Fritz and Kurt, another key factor to the success of these meetings was the city of Vienna with its culture, cafes, wine and beer pubs, etc. (see Hornik and Leisch, 2002, and also Figure 6).

2.3 Sweave & reproducibility

With `Sweave` (Leisch, 2002), Fritz pioneered what we now can understand as the technical foundation of reproducible research. `Sweave` was the main inspiration for **knitr** (Xie, 2015) which in turn led to **rmarkdown** (Xie et al., 2018) and **quarto** (Scheidegger et al., 2024). All these systems are used today to generate countless scientific articles, package vignettes, webpages, books, blogs, and much more in a dynamic and reproducible way.

Of course, Fritz was not the first one going in this direction. The concept of "literate programming" had been introduced by Knuth (1984), allowing to combine the source code for software and the corresponding documentation in the same file. The concepts of "tangling", that is, extracting the code for compilation, and "weaving", the process of generating a nicely looking document containing code next to prosa and formulae, have their roots in the `WEB` and `CWEB` systems (Knuth and Levy, 1993). As these packages were specific to code in Pascal (`WEB`) and C (`CWEB`), respectively, and documentation in LaTeX, Ramsey (1994) introduced his `noweb` system as a literate programming tool that is agnostic to the programming language used and also supports HTML in addition to LaTeX and a few other backends for documentation. The `noweb` syntax for code chunks is:

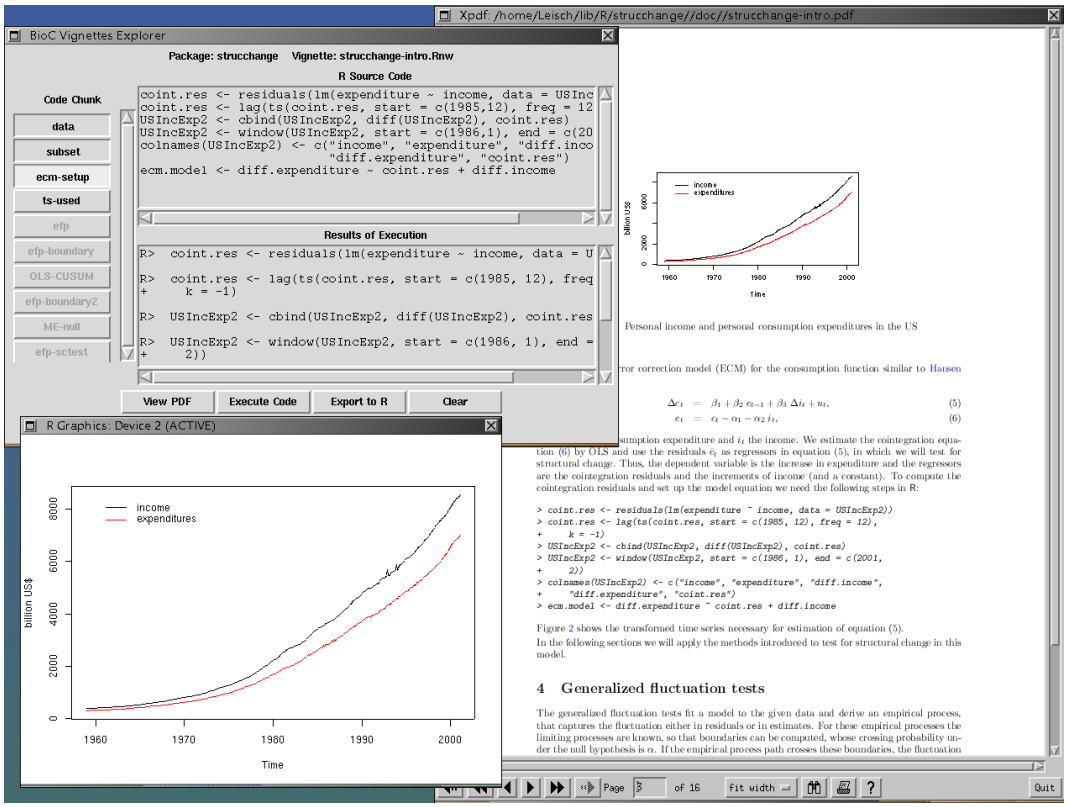


Figure 7: Screenshot of the strucchange package vignette, shown in a PDF viewer (right), along with the vExplorer from Bioconductor for interactive code execution (top left) with output in the active R graphics window (bottom left). Source: Leisch (2003, Figure 2).

```
<<code>>=
1 + 2
@
```

This will look familiar to users of `Sweave`. From this history, the naming decisions for the software and its file format can be understood: `Sweave` is the function that weaves code in S (or R - both languages still existed side by side at the time) with its output and documentation. And `Rnw` stands for files mixing R code with `noweb` syntax.

Starting in the mid-1990s to the early 2000s, interests shifted from just "literate programming" to "literate data analysis" (Leisch, 2002; Leisch and Rossini, 2003) as a core ingredient for reproducible research (Buckheit and Donoho, 1995). The seminal new idea was to have dynamic documents so *outputs* of code such as figures and tables could be updated automatically when the underlying data changed, which was pioneered by the late Günter Sawitzki in his `Voyager` system (Sawitzki, 1996).

Fritz amalgamated all of this into `Sweave` which was the first time that the power of dynamic reporting became easily available in a widely-used programming language for statistics in combination with the standard textprocessing system LaTeX. This turned out to be a "killer feature" of R at the time and the basis for further work towards reproducible research (Hothorn and Leisch, 2011; Stodden et al., 2014).

`Sweave` was also the basis for R package vignettes (Leisch, 2003) as an addition to the previously available technical manual pages. The first R package vignette published on CRAN in May 2002 was in the **strucchange** package, providing methods for testing, monitoring, and dating structural changes. The vignette was the `Sweave` adaptation of an introduction to the package that had been co-authored by Fritz and published a couple of months earlier in the *Journal of Statistical Software* (Zeileis et al., 2002). See Figure 7 for how Fritz used it to illustrate the idea of package vignettes in Leisch (2003) and that the R code from vignettes can be easily extracted (also interactively), explored, and re-run.

2.4 Clustering & mixture models

Fritz' theoretical and methodological work focused in particular on clustering and finite mixture models. Centroid-based partitioning methods as well as finite mixture models allow that their fitting

algorithm is embedded in a common estimation framework. In this framework, each of the steps is adapted in a modular way depending on the specific setup, e.g., the distance and centroid determining method or the component distribution used. Fritz exploited this for the implementation of the packages **flexclust** (Leisch, 2006) and **flexmix** (Leisch, 2004; Grün and Leisch, 2008), contributing to the clustering tools available for R (see the CRAN Task View *Cluster*). Both packages provide general infrastructure for (model-based) clustering and enable rapid prototyping and the simple extension to new variants taking into account complicated data structures or challenging model specifications (see, for example, **psychomix**, Frick et al., 2012).

2.5 Applied work

For many years, Fritz and Kurt actively participated in the Biological Psychiatry working group at Medizinische Universität Wien. The first paper co-authored by Fritz dates from 2000 (Bailer et al., 2000), the last from 2023 (Solmi et al., 2023). The joint research was mostly focused on linking genetic traits to psychiatric disorders and treatment success. This prompted many enhancements in the classical test infrastructure in base R - in surprising ways to some reviewers, who could not believe that Fisher's test really worked for tables with more than two rows or columns. It also established a strong need for conveniently reporting the results of the statistical analyses to the medical doctors in the group that went beyond providing annotated transcripts, which Fritz eventually managed to satisfy by inventing the `Sweave` system (see Section 2.3).

Fritz also intensively collaborated with Sara Dolnicar to advance data analytic methods for data-driven market segmentation analysis. They received the Charles R. Goeldner Article of Excellence Award for their work on extracting stable Winter tourist segments in Austria with bagged clustering (Dolnicar and Leisch, 2003). They focused on the evaluation of data structure and the selection of suitable segments based on segment stability as a key criterion (Dolnicar and Leisch, 2010, 2017). Finally, this joint work resulted in Dolnicar et al. (2018) which provides practical guidance for users of market segmentation solutions and for data analysts with respect to the technical and statistical aspects of market segmentation analysis.

As head of the Institute of Statistics, Fritz was involved in various interdisciplinary research projects covering almost the whole range of core areas of research at BOKU. He was key researcher at the Austrian Centre of Industrial Biotechnology (acib) (Scharl et al., 2009; Melcher et al., 2017) and faculty member of the doctoral schools on agricultural genomics and bioprocess engineering. Among others he contributed to the fields of zoology (Cech et al., 2022), forestry, transportation and tourism (Taczanowska et al., 2023) as well as chemistry, genomics and wildlife biology (Steiner et al., 2014).

3 Academic service

In addition to the services for the various conferences and proceedings already described above, he served the scientific community in various ways. In January 2001, he co-created *R News* which evolved into *The R Journal* eight years later. For the journal *Computational Statistics* he was an associate editor from 2005 to 2006 before he became editor-in-chief from 2007 to 2011 (see Symanzik et al., 2024, for more details). Other notable contributions include being editor for the *Journal of Statistical Software*, core member of the *Bioconductor* project for statistical software in bioinformatics, and first secretary general of the *R Foundation for Statistical Computing* when it was formed in 2002.

4 Teaching & mentoring

Fritz taught generations of students at bachelor, master, and PhD level and introduced hundreds of useRs to proper R development in his "Introduction to R Programming" short course. At TU Wien, LMU, and BOKU, he taught courses in applied statistics, statistical computing and computational statistics. He had the ability to explain even difficult content in a simple way and to inspire students with statistics and programming with R. He co-founded the "Munich R Courses" lecture series and was part of a group aiming to initiate a formal PhD program in statistics at LMU.

Fritz supervised Bettina Grün, Theresa Scharl, Sebastian Kaiser, Manuel Eugster, Christina Yassouridis, Rainer Dangl, Weksi Budiaji, Muhammad Atif and Simona Jokubauskaite as his PhD students. Based on his research, Fritz often discussed the state of and the need for reproducible research and taught his many students how to avoid the many small and innocent errors that have a tendency to pile up and invalidate reported statistical results, with potentially devastating consequences, as we all know.

5 Odds & ends

Fritz loved cooking, music, motorbike riding, playing cards with his friends, skiing and hiking. A late afternoon call to his office asking him to go along for a beer in Munich's English Garden almost never went unanswered, positively. Back in Vienna at BOKU, colleagues got to know Fritz as a very structured, thoughtful, calm person who involved everyone, listened to everyone and always endeavored to balance interests and ensure fairness. He strengthened cooperation and cohesion with his leadership style. Fritz was a friendly, always modest person who was free of airs and graces or vanity, despite or perhaps because of his great scientific successes. The R Core Team and the R community at large miss a contributor, collaborator, teacher, colleague, and friend.

References

U. Bailer, F. Leisch, K. Meszaros, E. Lenzinger, U. Willinger, R. Strobl, C. Gebhardt, E. Gerhard, K. Fuchs, W. Sieghart, S. Kasper, K. Hornik, and H. N. Aschauer. Genome scan for susceptibility loci for schizophrenia. *Neuropsychobiology*, 42(4):175–182, 2000. doi: 10.1159/000026690. [p11]

J. B. Buckheit and D. L. Donoho. WaveLab and reproducible research. In A. Antoniadis and G. Oppenheim, editors, *Wavelets in Statistics*, Lecture Notes in Statistics, pages 55–82. Springer-Verlag, New York, 1995. doi: 10.1007/978-1-4612-2544-7_5. [p10]

R. M. Cech, S. Jovanovic, S. Kegley, K. Hertoge, F. Leisch, and J. G. Zaller. Reducing overall herbicide use may reduce risks to humans but increase toxic loads to honeybees, earthworms and birds. *Environmental Sciences Europe*, 34(1):44, 2022. doi: 10.1186/s12302-022-00622-2. [p11]

S. Dolnicar and F. Leisch. Winter tourist segments in Austria: Identifying stable vacation styles using bagged clustering techniques. *Journal of Travel Research*, 41(3):281–292, 2003. doi: 10.1177/0047287502239037. [p11]

S. Dolnicar and F. Leisch. Evaluation of structure and reproducibility of cluster solutions using the bootstrap. *Marketing Letters*, 21(1):83–101, 2010. doi: 10.1007/s11002-009-9083-4. [p11]

S. Dolnicar and F. Leisch. Using segment level stability to select target segments in data-driven market segmentation studies. *Marketing Letters*, 28(3):423–436, 2017. doi: 10.1007/s11002-017-9423-8. [p11]

S. Dolnicar, B. Grün, and F. Leisch. *Market Segmentation Analysis: Understanding It, Doing It, and Making It Useful*. Management for Professionals. Springer-Verlag, 2018. doi: 10.1007/978-981-10-8818-6. [p11]

J. W. Eaton, D. Bateman, S. Hauberg, and R. Wehbring. *GNU Octave Version 9.2.0 Manual: A High-Level Interactive Language for Numerical Computations*, 2024. URL https://www.gnu.org/software/octave/doc/v9.2.0/. [p6]

H. Frick, C. Strobl, F. Leisch, and A. Zeileis. Flexible Rasch mixture models with package psychomix. *Journal of Statistical Software*, 48(7):1–25, 2012. doi: 10.18637/jss.v048.i07. [p11]

B. Grün and F. Leisch. Flexmix version 2: Finite mixtures with concomitant variables and varying and constant parameters. *Journal of Statistical Software*, 28(4):1–35, 2008. doi: 10.18637/jss.v028.i04. [p11]

K. Hornik. The Comprehensive R Archive Network. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(4):394–398, 2012. doi: 10.1002/wics.1212. [p7]

K. Hornik and F. Leisch, editors. *Proceedings of the 2nd International Workshop on Distributed Statistical Computing, Vienna, Austria*, 2001. URL https://www.R-project.org/conferences/DSC-2001/Proceedings/. ISSN 1609-395X. [p9]

K. Hornik and F. Leisch. Vienna and R: Love, marriage and the future. In R. Dutter, editor, *Festschrift 50 Jahre Österreichische Statistische Gesellschaft*, pages 61–70. Österreichische Statistische Gesellschaft, 2002. ISSN 1026-597X. [p7, 9]

K. Hornik, F. Leisch, and A. Zeileis, editors. *Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria*, 2003. URL https://www.R-project.org/conferences/DSC-2003/Proceedings/. ISSN 1609-395X. [p9]

T. Hothorn and F. Leisch. Case studies in reproducibility. *Briefings in Bioinformatics*, 12(3):288–300, 2011. doi: 10.1093/bib/bbq084. [p10]

D. E. Knuth. Literate programming. *The Computer Journal*, 27(2):97–111, 1984. doi: 10.1093/comjnl/27.2.97. [p9]

D. E. Knuth and S. Levy. *The CWEB System of Structured Documentation*. Addison-Wesley, Reading, 1993. [p9]

F. Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Rönz, editors, *COMPSTAT 2002 – Proceedings in Computational Statistics*, pages 575–580, Heidelberg, 2002. Physica Verlag. doi: 10.1007/978-3-642-57489-4_89. [p9, 10]

F. Leisch. Sweave, part II: Package vignettes. *R News*, 3(2):21–24, October 2003. URL https://CRAN.R-project.org/doc/Rnews/. [p10]

F. Leisch. FlexMix: A general framework for finite mixture models and latent class regression in R. *Journal of Statistical Software*, 11(8):1–18, 2004. doi: 10.18637/jss.v011.i08. [p11]

F. Leisch. A toolbox for k-centroids cluster analysis. *Computational Statistics and Data Analysis*, 51(2):526–544, 2006. doi: 10.1016/j.csda.2005.10.006. [p11]

F. Leisch and A. J. Rossini. Reproducible statistical research. *Chance*, 16(2):46–50, 2003. doi: 10.1080/09332480.2003.10554848. [p10]

M. Melcher, T. Scharl, M. Luchner, G. Striedner, and F. Leisch. Boosted structured additive regression for Escherichia coli fed-batch fermentation modeling. *Biotechnology and Bioengineering*, 114(2):321–334, 2017. doi: 10.1002/bit.26073. [p11]

N. Ramsey. Literate programming simplified. *IEEE Software*, 11(5):97–105, 1994. doi: 10.1109/52.311070. [p9]

G. Sawitzki. Extensible statistical software: On a voyage to Oberon. *Journal of Computational and Graphical Statistics*, 5(3):263–283, 1996. doi: 10.1080/10618600.1996.10474711. [p10]

T. Scharl, I. Voglhuber, and F. Leisch. Exploratory and inferential analysis of gene cluster neighborhood graphs. *BMC Bioinformatics*, 10(1):288, 2009. doi: 10.1186/1471-2105-10-288. [p11]

C. Scheidegger, C. Teague, C. Dervieux, J. J. Allaire, and Y. Xie. Quarto: An open-source scientific and technical publishing system, 2024. URL https://quarto.org/. Version 1.5. [p9]

M. Solmi, T. Thompson, A. Estradé, A. Agorastos, J. Radua, S. Cortese, E. Dragioti, F. Leisch, D. Vancampfort, L. C. Thygesen, H. Aschauer, M. Schlögelhofer, E. Aschauer, A. Schneeberger, C. G. Huber, G. Hasler, P. Conus, K. Q. Do Cuénod, R. von Känel, G. Arrondo, P. Fusar-Poli, P. Gorwood, P.-M. Llorca, M.-O. Krebs, E. Scanferla, T. Kishimoto, G. Rabbani, K. Skonieczna-Żydecka, P. Brambilla, A. Favaro, A. Takamiya, L. Zoccante, M. Colizzi, J. Bourgin, K. Kamiński, M. Moghadasin, S. Seedat, E. Matthews, J. Wells, E. Vassilopoulou, A. Gadelha, K.-P. Su, J. S. Kwon, M. Kim, T. Y. Lee, O. Papsuev, D. Manková, A. Boscutti, C. Gerunda, D. Saccon, E. Righi, F. Monaco, G. Croatto, G. Cereda, J. Demurtas, N. Brondino, N. Veronese, P. Enrico, P. Politi, V. Ciappolino, A. Pfennig, A. Bechdolf, A. Meyer-Lindenberg, K. G. Kahl, K. Domschke, M. Bauer, N. Koutsouleris, S. Winter, S. Borgwardt, I. Bitter, J. Balazs, P. Czobor, Z. Unoka, D. Mavridis, K. Tsamakis, V. P. Bozikas, C. Tunvirachaisakul, M. Maes, T. Rungnirundorn, T. Supasitthumrong, A. Haque, A. R. Brunoni, C. G. Costardi, F. B. Schuch, G. Polanczyk, J. M. Luiz, L. Fonseca, L. V. Aparicio, S. S. Valvassori, M. Nordentoft, P. Vendsborg, S. H. Hoffmann, J. Sehli, N. Sartorius, S. Heuss, D. Guinart, J. Hamilton, J. Kane, J. Rubio, M. Sand, A. Koyanagi, A. Solanes, A. Andreu-Bernabeu, A. S. J. Cáceres, C. Arango, C. M. Díaz-Caneja, D. Hidalgo-Mazzei, E. Vieta, J. Gonzalez-Peñas, L. Fortea, M. Parellada, M. A. Fullana, N. Verdolini, E. Andrlíková, K. Janků, M. J. Millan, M. Honciuc, A. Moniuszko-Malinowska, I. Łoniewski, J. Samochowiec, Ł. Kiszkiel, M. Marlicz, P. Sowa, W. Marlicz, G. Spies, B. Stubbs, J. Firth, S. Sullivan, A. E. Darcin, H. Aksu, N. Dilbaz, O. Noyan, M. Kitazawa, S. Kurokawa, Y. Tazawa, A. Anselmi, C. Cracco, A. I. Machado, N. Estrade, D. De Leo, J. Curtis, M. Berk, P. Ward, S. Teasdale, S. Rosenbaum, W. Marx, A. V. Horodnic, L. Oprea, O. Alexinschi, P. Ifteni, S. Turliuc, T. Ciuhodaru, A. Bolos, V. Matei, D. H. Nieman, I. Sommer, J. van Os, T. van Amelsvoort, C.-F. Sun, T. wei Guu, C. Jiao, J. Zhang, J. Fan, L. Zou, X. Yu, X. Chi, P. de Timary, R. van Winkel, B. Ng, E. Pena, R. Arellano, R. Roman, T. Sanchez, L. Movina, P. Morgado, S. Brissos, O. Aizberg, A. Mosina, D. Krinitski, J. Mugisha, D. Sadeghi-Bahmani, F. Sheybani, M. Sadeghi, S. Hadi, S. Brand, A. Errazuriz, N. Crossley, D. I. Ristic, C. López-Jaramillo, D. Efthymiou, P. Kuttichira, R. A. Kallivayalil, A. Javed, M. I. Afridi, B. James, O. J. Seb-Akahomen, J. Fiedorowicz, A. F. Carvalho, J. Daskalakis, L. N. Yatham, L. Yang, T. Okasha, A. Dahdouh, B. Gerdle, J. Tiihonen, J. I. Shin, J. Lee, A. Mhalla, L. Gaha, T. Brahim, K. Altynbekov, N. Negay, S. Nurmagambetova, Y. A. Jamei, M. Weiser, and C. U. Correll. Validation of the Collaborative Outcomes study on Health and Functioning during Infection Times (coh-fit) questionnaire for adults. *Journal of Affective Disorders*, 326:249–261, 2023. doi: 10.1016/j.jad.2022.12.022. [p11]

W. Steiner, F. Leisch, and K. Hackländer. A review on the temporal pattern of deer-vehicle accidents: Impact of seasonal, diurnal and lunar effects in cervids. *Accident Analysis & Prevention*, 66:168–181, 2014. doi: 10.1016/j.aap.2014.01.020. [p11]

V. Stodden, F. Leisch, and R. D. Peng. *Implementing Reproducible Research*. Chapman & Hall/CRC, Boca Raton, 2014. [p10]

J. Symanzik, Y. Mori, and P. Vieu. A memorial for the late Professor Friedrich Leisch. *Computational Statistics*, 39, 2024. Forthcoming. [p11]

K. Taczanowska, B. Latosinska, C. Brandenburg, F. Leisch, C. Czachs, and A. Muhar. Lobbying in social media as a new source of survey bias. *Journal of Outdoor Recreation and Tourism*, 44(A):100689, 2023. doi: 10.1016/j.jort.2023.100689. [p11]

Y. Xie. *Dynamic Documents with R and knitr*. Chapman & Hall/CRC, Boca Raton, 2nd edition, 2015. doi: 10.1201/9781315382487. [p9]

Y. Xie, J. J. Allaire, and G. Grolemund. *R Markdown: The Definitive Guide*. Chapman & Hall/CRC, Boca Raton, 2018. doi: 10.1201/9781138359444. [p9]

A. Zeileis, F. Leisch, K. Hornik, and C. Kleiber. strucchange: An R package for testing for structural change in linear regression models. *Journal of Statistical Software*, 7(2):1–38, 2002. doi: 10.18637/jss.v007.i02. [p10]

Bettina Grün
WU Wirtschaftsuniversität Wien
Austria
ORCiD: 0000-0001-7265-4773
Bettina.Gruen@wu.ac.at

Kurt Hornik
WU Wirtschaftsuniversität Wien
Austria
ORCiD: 0000-0003-4198-9911
Kurt.Hornik@R-project.org

Torsten Hothorn
Universität Zürich
Switzerland
ORCiD: 0000-0001-8301-0471
Torsten.Hothorn@R-project.org

Theresa Scharl
BOKU University
Austria
ORCiD: 0000-0001-8850-3312
Theresa.Scharl@boku.ac.at

Achim Zeileis
Universität Innsbruck
Austria
https://www.zeileis.org/
ORCiD: 0000-0003-0918-3766
Achim.Zeileis@R-project.org

ebmstate: An R Package For Disease Progression Analysis Under Empirical Bayes Cox Models

by Rui J. Costa and Moritz Gerstung

Abstract The new R package ebmstate is a package for multi-state survival analysis. It is suitable for high-dimensional data and allows point and interval estimation of relative transition hazards, cumulative transition hazards and state occupation probabilities, under clock-forward and clock-reset Cox models. Our package extends the package mstate in a threefold manner: it transforms the Cox regression model into an empirical Bayes model that can handle high-dimensional data; it introduces an analytical, Fourier transform-based estimator of state occupation probabilities for clock-reset models that is much faster than the corresponding, simulation-based estimator in mstate; and it replaces asymptotic confidence intervals meant for the low-dimensional setting by non-parametric bootstrap confidence intervals. Our package supports multi-state models of arbitrary structure, but the estimators of state occupation probabilities are valid for transition structures without cycles only. Once the input data is in the required format, estimation is handled automatically. The present paper includes a tutorial on how to use ebmstate to estimate transition hazards and state occupation probabilities, as well as a simulation study showing how it outperforms mstate in higher-dimensional settings.

1 Introduction

Multi-state models based on transition hazard functions are often used in the statistical analysis of longitudinal data, in particular disease progression data (Hougaard, 1999). The multi-state model framework is particularly suitable to accommodate the growing level of detail of modern clinical data: as long as a clinical history can be framed as a random process which, at any moment in time, occupies one of a few states, a multi-state model is applicable. Another strong point of this framework is that it can incorporate a *regression model*, i.e., a set of assumptions on how covariates, possibly time-dependent ones, affect the risk of transitioning between any two states of the disease. Once estimated, multi-state models with regression features allow the stratification of patients according to their transition hazards. In addition, it is possible, under some models, to generate disease outcome predictions. These come in the form of *state occupation probability* estimates, meaning estimates of the probability of being in each state of the disease over a given time frame.

The survival analysis 'task view' of the Comprehensive R Archive Network lists seven R packages that are able to fit *general* multi-state models and, at the same time, feature some kind of regression model or algorithm: `flexsurv` (Jackson, 2016), **msm** (Jackson, 2011), **SemiMarkov** (Listwon and Saint-Pierre, 2015), **survival** (Therneau, 2015), **mstate** (de Wreede et al., 2010), **mboost** (Hothorn et al., 2020) – as extended by **gamboostMSM** (Reulen, 2014) – and **penMSM** (Reulen, 2015). All of them implement relative risk regression models (as defined in Aalen et al., 2008, p. 133). The only exceptions are **survival**, which also fits Aalen's additive regression model (Aalen, 1989), and `flexsurv`, which also implements accelerated failure time models (see, for example, Aalen et al., 2008, p. 443).

Recall that a Cox regression model is a semi-parametric model in which every transition hazard is assumed to be the product of a baseline hazard function of unspecified form (the non-parametric component) and an exponential relative risk function (the parametric component) (Aalen et al., 2008, p. 133). Generally, the relative risk regression models implemented in these packages are Cox regression models. However, some models in `flexsurv`, as well as those in **msm** and **SemiMarkov**, also restrict the baseline hazards to specific parametric families, i.e. they are fully parametric. In **msm** and **SemiMarkov**, the stronger assumptions regarding the functional form of the hazard are leveraged to do away with other common assumptions: **SemiMarkov** drops the usual Markov property to implement homogeneous semi-Markov models; **msm** is suitable for *panel data*, i.e., data in which the state of each individual is known only at a finite series of times.

Packages **penMSM** and **gamboostMSM** are the best suited to deal with higher-dimensional covariate data. The first of these packages relies on a structured fusion lasso method, while the second implements (jointly with **mboost**) a boosting algorithm. Both methods induce sparsity in the number of non-zero covariate effects, as well as equality among the different transition effects of each covariate, and are thus especially useful to reduce complicated multi-state models to more interpretable ones. The remaining packages assume standard, fixed effects relative risk regression models and do not include regularisation or variable selection features.

It is also illustrative to order the seven packages mentioned according to how extensive their analysis workflow is. Packages **SemiMarkov** and **penMSM** are intended for the estimation of relative transition hazards only (i.e., for estimating the impact of covariates on each transition hazard). With the package **mboost** (as extended by **gamboostMSM**) it is also possible to estimate the baseline transition hazards. Finally, a more complete workflow including estimates of both relative and cumulative transition hazards, as well as state occupation probabilities, is implemented in `flexsurv`, **msm** and **mstate**, and has been under implementation in **survival** (version 3.0 or later).

The present paper provides an introduction to **ebmstate**, a new R package for multi-state survival analysis available for download on the Comprehensive R Archive Network (CRAN). The main goal of **ebmstate** is to provide an analysis framework for the Cox model that performs better with higher-dimensional covariate data and is also complete, in the sense of being able to generate point and interval estimates of relative transition hazards, cumulative transition hazards and state occupation probabilities, both under clock-forward and clock-reset models. A fundamental characteristic of **ebmstate** is that it re-implements and extends the analysis framework of **mstate**, which is complete in the sense just mentioned. In fact, to a large extent, our package was built by importing, adapting and replacing functions from the **mstate** package. This not only eliminates redundancies, but also makes our package more accessible to the numerous users of **mstate** (the three papers associated with **mstate** have jointly over 2000 citations).

To improve the performance of **mstate**'s multi-state Cox model when dealing with higher-dimensional covariate data, a ridge-type regularisation feature was added. We allow the regression coefficients of the model to be partitioned into groups, with each group having its own Gaussian prior. A group can gather, for example, all the regression coefficients for a given transition. Or, within a given transition, coefficients can be grouped according to the covariate type they refer to (for example, demographic, clinical or genomic type). The resulting hierarchical Bayes model is *empirical* in that a full prior elicitation is not required (the mean and variance hyper-parameters of the Gaussian are estimated from the data). Model fitting relies on the iterative algorithm introduced by Schall (1991), which typically converges after a small number of steps. A simulation study showing that Schall's algorithm performance compares well with that of other algorithms for ridge penalty optimisation, including one based on cross-validation, can be found in Perperoglou (2014).

The asymptotic confidence intervals generated by **mstate** are applicable when the number of observations is much larger than the number of parameters to be estimated (see section Interval estimation below). To preserve the completeness of **mstate**'s framework in higher-dimensional settings, we therefore implemented non-parametric bootstrap intervals of regression coefficients, cumulative transition hazards and state occupation probabilities.

The high computational cost implied by the non-parametric bootstrap motivated a third extension to **mstate**. We developed an estimator of state occupation probabilities under clock-reset Cox models that is based on a convolution argument (as in Spitoni et al., 2012) and the Fast Fourier transform (FFT). At present, the estimation of such probabilities for clock-forward Cox models can be carried out using the efficient, product-limit based algorithm available in **mstate**. However, for clock-reset Cox models, only a simulation-based estimator is available in this package (see also the `flexsurv` package for a similar, simulation-based estimator). The FFT estimator in **ebmstate** was conceived as a faster alternative to this simulation-based estimator, but its scope is currently restricted to multi-state models with transition structures that have no cycles, i.e. in which a transition between two states is either not possible or follows a unique sequence of states. Figure 1 provides a short graphical summary of **ebmstate**, with the main inputs – a genomic-clinical data set and an empirical Bayes multi-state Cox model – and the main outputs – the estimates of relative hazards and state occupation probabilities (cumulative transition hazards are omitted).

As already mentioned, our empirical Bayes method improves estimator performance in models with larger numbers of covariates (see section Estimator performance on estimator performance). Also, as a ridge-type regression method, it can be used as an alternative to the lasso method of **penMSM** in two particular cases: when the levels of correlation between covariates are high enough to compromise the stability of lasso-based covariate selection; or simply to improve prediction accuracy when interpretability is not essential and the number of covariates is not greater than the number of observations (Zou and Hastie, 2005). In addition, and perhaps more importantly, **ebmstate** goes beyond the regularised estimation of transition hazards offered by **penMSM** and **gamboostMSM**: point and interval estimates of state occupation probabilities under the regularised Cox model can also be computed.

2 Models

A multi-state Cox model is a continuous-time stochastic process with a finite (and usually small) state space $\mathcal{S}$. To better describe the models implemented in **ebmstate**, we define the following notation. We let t denote the time since some initiating event (usually diagnosis or disease onset). For $t \in [0, \infty)$, we define the following random variables: $X(t)$ represents the disease state of the patient, $S(t)$ the time spent in the current state, and $\vec{Z}(t)$ the value of a covariate vector. The realisation of each component of the process $\{\vec{Z}(t)\}$ is a step function, possibly approximating the evolution in time of a continuous covariate. In addition, $\{\vec{Z}(t)\}$ is assumed not-adapted to the filtration generated by $\{X(t)\}$ (an adapted covariate is one whose path until t is known once $\{X(u)\}$, $u \leq t$, is known). The transition hazard rate of a patient from state i to state j ($i \neq j$) at time t, conditional on the sojourn time and the covariate vector, is defined as

$$\alpha_{ij}(t|\mathbf{z}, s) := \lim_{h \downarrow 0} \frac{1}{h} \mathrm{P}\left[X(t+h) = j \,|\, X(t) = i, S(t) = s, \vec{Z}(t) = \mathbf{z}\right] \ , \ s \in [0, \infty) \ , \ t \in [s, \infty) \ .$$

Independent right-censoring and left-truncation are assumed throughout (Aalen et al., 2008, p. 57). The purpose of the present section is to give a (not necessarily exhaustive) description of the scope of **mstate** and **ebmstate** with respect to the multi-state Cox model. Using the terminology in de Wreede et al. (2011), a Cox model is termed a 'clock-reset' model when

$$\alpha_{ij}(t \,|\, \mathbf{z}, s) = \lambda_{ij}^{(0)}(s) \exp\left[\boldsymbol{\beta}_{ij}^{\mathsf{T}} \mathbf{z}\right] \quad , \tag{1}$$

and it is termed a 'clock-forward' model when

$$\alpha_{ij}(t \,|\, \mathbf{z}) = \alpha_{ij}^{(0)}(t) \exp\left[\boldsymbol{\beta}_{ij}^{\mathsf{T}} \mathbf{z}\right] \quad . \tag{2}$$

In both cases, $i, j \in \mathcal{S}$, with $i \neq j$; $\boldsymbol{\beta}_{ij}$ is an unknown vector of regression coefficient parameters, and both $\lambda_{ij}^{(0)}(\cdot)$ and $\alpha_{ij}^{(0)}(\cdot)$ are unknown (baseline hazard) functions, non-negative on $\mathbb{R}^+$. When, as in equation 1, $\alpha_{ij}(t|\mathbf{z}, s)$ is the same for all $t \geq s$, we simplify its notation to $\lambda_{ij}(s|\mathbf{z})$. As can be seen from equations 1 and 2, the 'clock-reset' and 'clock-forward' models are models for how the transition hazard rates are affected by time. In the former case, the only relevant time scale is the time s spent in the current state, whereas in the latter only the time t since the initiating event matters. While the 'clock-forward' model is arguably the default one in multi-state survival analysis (Andersen et al., 1993; Aalen et al., 2008), in some cases the 'clock-reset' model is more appropriate. For example, in some forms of cancer, it can be sensible to assume that the transition hazards from the state of complete remission depend on the sojourn time, rather than on the time since the initial diagnosis.

2.1 Relative transition hazards

The parametric component of the transition hazard from i to j, written $\exp\left[\boldsymbol{\beta}_{ij}^{\mathsf{T}} \mathbf{z}\right]$, is termed the relative transition hazard. In **mstate** and **ebmstate**, estimating the relative transition hazard amounts to estimating the regression coefficient vector $\boldsymbol{\beta}_{ij}$. In **mstate**, these parameters are assumed to be non-random. With **ebmstate**, the following prior distributions can be imposed.

Define $\mathcal{P}$ as the set of all pairs of states between which a direct transition is possible. Let $\{\boldsymbol{\beta}_{ij}\}$, for all $(i, j) \in \mathcal{P}$, be a partition of $\boldsymbol{\beta}$, a vector containing the regression coefficients for all direct transitions allowed. Each $\boldsymbol{\beta}_{ij}$ is further partitioned into $\{\boldsymbol{\beta}_{ijk}\}$, for $k \in \{1, 2, ..., n_{ij}\}$. In **ebmstate**, the most general model regarding the prior distribution of $\boldsymbol{\beta}$ makes two assumptions: a) the scalar components of $\boldsymbol{\beta}$ are independent and normally distributed; b) the scalar components of $\boldsymbol{\beta}_{ijk}$ have a common (and undetermined) mean μ_{ijk} and a common (and also undetermined) variance σ_{ijk}^2.

The purpose of the framework just described is to allow the clustering of covariate effects according to their prior distribution. If there is no prior knowledge about how this clustering should be done, a single Gaussian prior can be imposed on all regression coefficients at once. If prior knowledge allows the grouping of effects according to the transition they refer to, a different Gaussian prior can be assigned to the coefficients of each transition. Even within each transition, different groups of coefficients can be assigned different prior distributions. In the analysis of biomedical data, for example, there can be a split between genes which are known to affect the transition hazard, and other genes whose effect is unknown.

2.2 Cumulative transition hazard functions

Our package imports from **mstate** a Breslow estimator of two types of cumulative transition hazard: one on a global time scale, defined as

$$\mathrm{A}_{ij}\left(t \mid \mathbf{z}\right) := \int_0^t \alpha_{ij}^{(0)}\left(u\right) \exp\left[\boldsymbol{\beta}_{ij}^{\mathsf{T}} \mathbf{z}\right] \mathrm{d}u \quad ,$$

and another on a sojourn time scale, defined as

$$\Lambda_{ij}(s \mid \mathbf{z}) := \int_0^s \lambda_{ij}^{(0)}\left(u\right) \exp\left[\boldsymbol{\beta}_{ij}^{\mathsf{T}} \mathbf{z}\right] \mathrm{d}u \quad .$$

Note that, in either case, the covariate vector is assumed to remain constant.

2.3 State occupation probabilities

By state occupation probability, we mean the probability that a patient in state i at time 0 finds herself in state j at time t. The estimates of these probabilities can be seen as functionals of the estimated cumulative transition hazard functions. For this reason, the restriction to models with time-fixed covariates, which was just seen to be applicable to the estimators of cumulative transition hazards, carries over to the estimation of state occupation probabilities.

When conditioning on a given covariate path (time-fixed or not), state occupation probability estimates are not valid unless the covariates are *external* (Cortese and Andersen, 2010; Aalen et al., 2008, p. 142). Note that a vector of covariates $\{\vec{Z}(u)\}_{u\geq 0}$ is said to be *external* if, for all $t \in [0, \infty)$, each transition hazard at t, conditional on $\vec{Z}(t)$, is independent of $\{\vec{Z}(u)\}_{u>t}$ (i.e. independent of the future path of the covariate). Otherwise, it is said to be *internal* (for more details on the distinction between internal and external covariates, see Kalbfleisch and Prentice, 2002, chapter 6). When one does not wish (or is not possible due to $\vec{Z}$ being *internal*) to condition on a future covariate path of the covariate process, the uncertainty introduced by this process needs to be accounted for. This can be done by extending the state space of the disease process, so that it includes information on the disease *and* the covariate process (Andersen et al., 1993, p. 170). For example, to include a dichotomous transplant covariate (an internal covariate) in a simple survival model with two states, the state space is expanded from {alive, deceased} to {alive without transplant, alive with transplant, deceased}. One can then either assume that transplanted patients have a different baseline death hazard or, more simply, that transplantation scales the death hazard by some constant $\exp(\gamma)$. A similar but more detailed example can be found in de Wreede et al. (2010, section 2.3.2, 'model 3').

3 Estimation

In the current section, we present the estimation methods underlying the extensions of **mstate** implemented in **ebmstate**.

3.1 Relative and cumulative hazard functions

Let $\boldsymbol{\mu}_{ij}$, with $(i, j) \in \mathcal{P}$ (the set of direct transitions allowed), denote a vector whose scalar components are the parameters μ_{ijk}, $k \in \{1, 2, ..., n_{ij}\}$. Similarly, let $\boldsymbol{\sigma}_{ij}^2$ be composed of the parameters $\left\{\sigma_{ijk}^2\right\}_k$. The estimation of $\boldsymbol{\beta}$, $\boldsymbol{\mu} := \{\boldsymbol{\mu}_{ij}\}$ and $\boldsymbol{\sigma}^2 := \{\boldsymbol{\sigma}_{ij}^2\}$ relies on the restricted maximum-likelihood (REML) type algorithm described in Perperoglou (2014), and introduced by Schall (1991). The resulting estimate of $\boldsymbol{\beta}$ is a maximum *a posteriori* estimate; the estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are empirical Bayes estimates. In **ebmstate**, the estimator based on this algorithm is implemented in the function `CoxRFX` . The results of a simulation study showing its consistency are included in the Supporting Scripts and Data (file ESM_1.html, section 1).

The computation of cumulative hazard rates for given covariate values and an estimated regression coefficient vector relies on the function `msfit_generic`, which is essentially a wrapper for the function `mstate::msfit` (see section Computing cumulative transition hazard estimates). For the mathematical details of this computation, we refer therefore the reader to de Wreede et al. (2010).

3.2 State occupation probabilities

The package **mstate** includes a simulation-based estimator that can take as input either $\hat{\mathrm{A}}_{ij}(\cdot \mid \mathbf{z})$ or $\hat{\Lambda}_{ij}(\cdot \mid \mathbf{z})$ to generate estimates of state occupation probabilities under the clock-forward or the clock-reset model respectively. Another available estimator, an Aalen-Johansen-type estimator based on product integration, is far more efficient computationally and takes as input $\hat{\mathrm{A}}_{ij}(\cdot \mid \mathbf{z})$ only. As the scope of this estimator has been restricted to clock-forward Cox models (Andersen et al., 1993; Aalen et al., 2008), in our package we implemented a convolution-based estimator as a computationally efficient alternative (for models with a transition structure that has no cycles).

For convenience, let the sequence of states from 0 to n have the labels $0, 1, 2, ..., n$, where 0 is the initial state by definition, and n is some state that might (eventually) be reached by the process. In addition, define $X_0 := X(0)$ and $T_0 := 0$, and let (X_i, T_i), $i \in \{1, 2, ...\}$, denote the marked point process associated with $\{X(t)\}$, so that T_i is the time of the i^{th} transition and X_i is the state the process jumps to at time T_i. The inter-transition times are denoted by $\tau_{ij} := T_j - T_i$, for $j > i$. We can write the probability that a patient in state 0 at time 0 finds herself in state n at time t, conditional on $\vec{Z}(u) = \mathbf{z}$ for all $u \geq 0$, as

$$\begin{aligned}&\mathrm{P}\left[X(t) = n \mid X(0) = 0\,, \vec{Z}(u) = \mathbf{z},\, u \geq 0\right] \\ &= \mathrm{P}\left[X_n = n, \tau_{0,n} < t, \tau_{n,n+1} \geq t - \tau_{0,n} | X_0 = 0\,, \vec{Z}(u) = \mathbf{z},\, u \geq 0\right]\,.\end{aligned}$$

Recall that $\lambda_{i,i+1}(s \mid \mathbf{z})$ denotes the hazard rate of a transition to state $i+1$ at time s since arrival in state i, for a patient that has covariate vector $\mathbf{z}$. The cumulative hazard for the same transition between sojourn times 0 and s, if the patient's covariate vector remains constant at $\mathbf{z}$, is represented by $\Lambda_{i,i+1}(s \mid \mathbf{z}) := \int_0^s \lambda_{i,i+1}(x \mid \mathbf{z})\,\mathrm{d}x$. Similarly, we let $\lambda_i(s \mid \mathbf{z})$ represent the hazard rate of going to any state that can be reached directly from i, at time s since arrival in state i, for a patient with covariate vector $\mathbf{z}$. The cumulative hazard for the same event between sojourn times 0 and s, if the patient's covariate vector remains constant at $\mathbf{z}$, is represented by $\Lambda_i(s \mid \mathbf{z})$. The expressions $\hat{\Lambda}_i(s \mid \mathbf{z})$ and $\hat{\Lambda}_{i,i+1}(s \mid \mathbf{z})$ denote the Breslow estimators of the cumulative hazards just defined. In what follows, all references to probabilities, hazard rates and cumulative hazards are to be understood as conditional on $\vec{Z}(u) = \mathbf{z}$, for $u \geq 0$: this condition is omitted to simplify the notation.

In **ebmstate**, the function `probtrans_ebmstate` generates a set of state occupation probability estimates at equally spaced time points:

$$\{\hat{p}_{0n}(k)\}_k := \left\{\hat{\mathrm{P}}\left[X_n = n, \tau_{0,n} < t_k, \tau_{n,n+1} \geq t_k - \tau_{0,n} \mid X_0 = 0\right]\right\}_k\,,\ k = 0, 1, 2, ..., K\,;\ t_k = k \times \Delta t\,.$$

The number K of time intervals is 10,000 by default and t_K is a parameter set by the user. Defining the functions

$$q_{ij}(k) := \mathrm{P}\left[X_j = j, \tau_{ij} \in [t_k, t_{k+1}) \mid X_i = i\right]$$

and

$$r_i(k) := \mathrm{P}\left[\tau_{i,i+1} > t_k \mid X_i = i\right]\,,$$

and the finite difference

$$\Delta\hat{\Lambda}_{i,i+1}(t_k) := \hat{\Lambda}_{i,i+1}(t_{k+1}) - \hat{\Lambda}_{i,i+1}(t_k)\,,$$

the algorithm behind `probtrans_ebmstate` can be described as follows:

1. For $j = 1, 2, ..., n$, compute

$$\hat{q}_{j-1,j}(k) := \exp\left[-\hat{\Lambda}_{j-1}(t_k)\right]\Delta\hat{\Lambda}_{j-1,j}(t_k) \tag{3}$$

   for $k = 0, 1, ..., K-1$.

2. For $j = 2, 3, ..., n$, compute (iteratively)

$$\hat{q}_{0j}(k) := \sum_{l=0}^{k-1} \hat{q}_{j-1,j}(k-l-1)\,\hat{q}_{0,j-1}(l) \tag{4}$$

   for $k = 0, 1, ..., K-1$.

3. Finally, use the estimates obtained in the last iteration of step 2 to compute

$$\hat{p}_{0n}(k) := \sum_{l=0}^{k-1} \hat{r}_n(k-l-1)\,\hat{q}_{0,n}(l) \tag{5}$$

for $k = 0, 1, ..., K$, where $\hat{r}_n(\cdot) := \exp\left[-\hat{\Lambda}_n\left(t_{(\cdot)}\right)\right]$.

Substituting := for ≈ and removing the 'hats' in definitions 3 to 5, we get the approximate equalities that justify the algorithm. These approximate equalities are derived in the Supporting Scripts and Data (file ESM_1.html, section 2).

Apart from `probtrans_ebmstate`, the function `probtrans_fft` is also based on the convolution argument just shown. However, this function makes use of the convolution theorem, i.e., of the fact that the convolution of two (vectorized) functions in the time domain is equivalent to a pointwise product of the same functions in the frequency domain. The estimation of state occupation probabilities is thus simplified to

$$\hat{p}_{0n} := \mathcal{F}^{-1}\left\{\hat{\mathfrak{q}}_{0,1} \cdot \hat{\mathfrak{q}}_{1,2} \cdot ... \cdot \hat{\mathfrak{q}}_{n-1,n} \cdot \hat{\mathfrak{r}}_n\right\},$$

where $\mathcal{F}$ denotes the discrete Fourier transform, $\hat{\mathfrak{q}}_{j-1,j} := \mathcal{F}(\hat{q}_{j-1,j})$ and $\hat{\mathfrak{r}}_n := \mathcal{F}(\hat{r}_n)$. Conversion to and from the frequency domain is carried out using the fast Fourier transform algorithm implemented in the `fft` function of the base package `stats`. The Supporting Scripts and Data contain a short simulation study checking that state occupation probabilities can be accurately estimated with `probtrans_ebmstate` and `probtrans_fft` (see file ESM_1.html, sections 3 and 4).

Figure 2 consists of a grid of plots with estimated curves of state occupation probabilities. It compares, in terms of speed and accuracy, the estimator in `probtrans_fft` with an estimator in `mstate::mssample` that has the same target, but is simulation-based. Each plot contains a black curve and a superimposed red curve. The red curves in any given column of the grid are all based on the same run of a function: columns 1 to 3 are based on runs of `mssample` with the number of samples n equal to 100, 1000 and 10.000 respectively, while column 4 is based on a run of `probtrans_fft`. Each column in the grid reproduces the same 4 black curves. These are based on a single run of `mssample` with $n = 100.000$ and serve as benchmark. All function runs are based on the same input: a set of cumulative transition hazard estimates for a multi-state model with the 'linear' transition structure given in the leftmost diagram of figure 3. Plots in a given row refer to the same state of the model. The running times on top of each column refer to the estimation of red curves. The main conclusion suggested by this analysis of simulated data is that `probtrans_fft` is as accurate as `mssample` with $n = 10.000$, but it is almost 100 times faster (columns 3 and 4). With $n = 1000$, `mssample` achieves a good approximation to the true state occupation probabilities, but is still roughly 9 times slower. The details on how figure 2 and its underlying data were generated are given in the Supporting Scripts and Data (file ESM_1.html, section 5).

3.3 Interval estimation

Under any model estimated by **ebmstate** – as in general under a Bayesian model –, one can, if the sample size is large enough, approximate the posterior by a normal distribution with mean equal to the maximum *a posteriori* estimate and covariance matrix equal to the inverse of the generalised observed Fisher information (see, for example, Gelman et al., 2014, p. 83-84). This approximation has first-order accuracy and is thus outperformed by Laplace's method, which has second-order accuracy (Carlin and Louis, 2009, p. 110-111). However, as Carlin and Louis (2009, p. 112) observe, "for moderate- to high-dimensional $\boldsymbol{\theta}$ (say, bigger than 10), Laplace's method will rarely be of sufficient accuracy[...]". Carlin and Louis (2009, p. 244-251) also describe three methods of interval estimation in empirical Bayes settings, but all of them are designed for fully parametric models. These reasons, along with the fact that regularised methods such as the one implemented **ebmstate** are typically used to fit models with more than a dozen covariates, led us to choose the non-parametric bootstrap as the interval estimation method in **ebmstate**. Note that the non-parametric bootstrap can be given a Bayesian interpretation. Its interval estimates are approximately the same as those of a Bayesian model that assumes: a) a multinomial distribution for the data; and b) a non-informative Dirichlet prior distribution for the probability assigned to each category in the multinomial distribution. This is a specific case of the so-called Bayesian bootstrap (Hastie et al., 2009, p. 272). Further research is needed to determine the theoretical properties of the non-parametric bootstrap in the present setting, but this falls beyond the scope of the present paper. Interval estimates of regression coefficients, cumulative hazards and state occupation probabilities are implemented in the function `boot_ebmstate`.

4 Estimator performance

It is a well-documented fact in the statistical literature that standard least-squares or maximum-likelihood estimators can often be improved by regularisation or shrinkage (see, for example, Samworth, 2012). This improvement comes about when the model dimensionality is high enough that the bias introduced by regularisation is outweighed by the reduction in the estimator variance. In the current setting, one might therefore ask: what kind of dimensionality does a semi-parametric, multi-state Cox model need to have to be outperformed by its empirical Bayes counterpart? A simulation study we carried out offers a tentative answer to this question, by comparing estimators under both Cox models for an increasing number of covariates. The study also features a third method, based on a fully non-parametric model, as a null model method. This was included to give an idea of how many covariates the empirical Bayes model can deal with before it becomes no better than a simple non-regressive model.

4.1 Simulation setup

We assessed the performance of all estimators defined by the tuple $[a, m, G, n, p(n)]$, where $a \in$ {regression coefficients, relative hazards, state occupation probabilities} is the target of estimation, $m \in$ {standard Cox, empirical Bayes Cox, null} is the assumed hazard model, $G \in$ {linear, competing risks, 'm' structure} is the transition structure of the model (illustrated in figure 3) and $n \in \{100, 1000\}$ is the number of patients/disease histories in the training data set; the variable p denotes the number of coefficients/covariates per transition in the true model and its range depends on n: $p\,(100) \in \{10, 40, 70, 100\}$ whereas $p\,(100) \in \{10, 100, 200, 300, 400, 500\}$. By 'relative hazards' and 'state occupation probabilities', we mean here the relative transition hazards of an out-of-sample patient, and her state occupation probabilities at 7 chosen time points. We generated a batch of 300 independent absolute error observations ('NA' estimates included) for each estimator, where each observation is recorded after training the estimator on a newly simulated data set. Each boxplot in figures 6 ($n = 100$) and 7 ($n = 1000$) is based on one of these batches. As all estimators are *vector* estimators, each absolute error is actually an *average* absolute error, where the average is taken over the components of the vector.

All training data sets were simulated from clock-reset Cox models. Apart from G (the model transition structure), n and p, also the true baseline hazards are held fixed within each batch of 300 training data sets. The coefficient vectors used in the simulation are always non-sparse and are scaled by $\sqrt{\frac{10}{p}}$ to keep the log-hazard variance constant when the dimensionality grows. All covariates are dichotomous and mutually independent. To compute the coefficient errors for the non-parametric (null) model method, we think of it as a degenerate Cox model in which all regression coefficient estimates are fixed at zero. The estimation of regression coefficients under the standard Cox and the empirical Bayes Cox models was performed with `survival::coxph` and `ebmstate::CoxRFX` respectively; the estimation of state occupation probabilities is based on `mstate::probtrans` for the null model and on `ebmstate::probtrans_fft` for both the standard Cox and the empirical Bayes Cox models.

The reason we did not consider simulation scenarios with more than 500 covariates per transition, in data sets of 1000 patients, was simply computational cost. For example, generating the data and error observations for the scenario with $n = 1000$, $p = 100$ and $G =$'m' structure took less than one hour to generate using 20 CPU cores in parallel; the same scenario but with $p = 500$ took 6.5 days using 25 CPU cores. More details about the simulation setup can be found in the Supporting Scripts and Data (file ESM_1.html, section 6, subsection 'sample script').

4.2 Missing values

Whenever an estimator was able to compute a valid estimate of its target for each training data set, i.e., when it did not return any 'NA' estimates, its boxplots are based on 300 valid error observations. This was always the case with non-parametric estimators: the estimates of regression coefficients and relative hazards of this type of estimators are trivial (fixed at zero and one respectively) and hence it is also straightforward to compute absolute errors. It also happened that non-parametric estimators of state occupation probabilities had no 'NA' estimates (see file ESM_1.html, section 6, figure 6.3, in the Supporting Scripts and Data). The situation was similar for the empirical Bayes Cox model estimators, which showed no more than 5% missing estimates in any of the simulation scenarios studied (ibid., figures 6.1 and 6.2). However, for the standard Cox model ones, the number of 'NA' estimates depends to a large extent on the number of patients in the data set, as well as on the dimensionality and transition structure of the model (figures 4 and 5). In data sets of 100 patients, it fares well in models with fewer than 10 covariates per transition, or in models with up to 40 covariates,

if the transition structure is linear. Otherwise its failure rates range from roughly 25% to nearly 100%. In data sets of 1000 patients, the proportion of 'NA' estimates is never above 10%, if the transition structure is linear, but it can climb above 60% for other transition structures.

4.3 Comparison of estimators

With respect to the performance of the three methods studied, the boxplots in figures 6 and 7 suggest the following conclusions:

- As p/n grows, the empirical Bayes estimators quickly outperform the standard Cox model ones. They already fare substantially better at $p/n = 0.1$ for both $n = 100$ and $n = 1000$ and for all estimation targets. At the same time, the relative performance of the empirical Bayes method with respect to the null model one decreases. At $p/n = 0.5$, the difference between these two methods is already rather small for all simulation scenarios.
- The relative performance of the empirical Bayes method with respect to the null method decreases as the number of co-occurring transition hazards in the model grows. All other things equal, the empirical Bayes method has the best performance under the 'linear' structure model, which has no competing transitions; it performs less well under the 'm' structure transition model, where two transition hazards can co-occur; and has the worse relative performances under the 'competing risks' model, where three transition hazards co-occur. This trend is clearer for $n = 100$ (figure 6) but can also be detected in the relative hazard errors for $n = 1000$ (figure 7). In any case, the empirical Bayes method seems to be far more robust than the standard Cox model against increases in the number of co-occurring transition hazards.
- Having as target the regression coefficients or the state occupation probabilities, instead of relative hazards, makes the empirical Bayes method better in comparison to the null method. In fact, as p/n grows, the empirical Bayes method is never outperformed by the null method except in the estimation of relative hazards.

5 Survival analysis workflow

The features of `mstate` were illustrated in de Wreede et al. (2010) using a simple workflow. The starting point of this workflow is a data set in 'long format'. Such data set can be fed into `survival::coxph` to obtain estimates of the regression coefficients of a multi-state Cox model. The resulting model fit object can be passed on to `mstate::msfit`, along with a vector of covariates of a particular patient, to get personalised estimates of the cumulative hazard functions. Finally, state occupation probabilities for the same patient can be estimated if the object created by `mstate::msfit` is fed into `mstate::probtrans`. In this section, we describe how **ebmstate** extends the scope of this workflow, i.e., how it uses the packages **survival** and **mstate** to generate estimates under a multi-state empirical Bayes Cox model. A diagram summarising the extension is shown in figure 8. In the Model assessment subsection, we give some recommendations on how to assess and compare models, but for more detailed tutorials on how to analyse multi-state data using models defined by transition hazards, we refer the reader to Putter et al. (2007) and Putter (2011).

The main steps of the **ebmstate** workflow are here illustrated using a data set of patients with myelodysplastic syndromes (MDS) which has been described and studied in Papaemmanuil et al. (2013). A myelodysplastic syndrome is a form of leukemia in which the bone marrow is not able to produce enough mature blood cells, and which sometimes develops into a cancer of white blood cells with a quick and aggressive progression, i.e., into acute myeloid leukemia (AML). Figure 9a illustrates an illness-death type model for MDS patients and also gives a breakdown of the number of transition events. The conversion to a model with a transition structure that has no cycles (i.e., that can be handled by our convolution-based estimators) is shown in figure 9b. The data set used for model estimation, obtained after a number of pre-processing steps, contains the disease history of 576 patients, as well as measurements on 30 covariates. Of these 30 covariates, 11 are mutation covariates and the remaining are clinical or demographic (see figure 9c). The running time for the estimation of relative transition hazards does not exceed 10 seconds in a standard laptop computer. The same holds for the estimation of cumulative transition hazards or state occupation probabilities for a given patient. The complete R code underlying the data analysis in the current section can be found in the Supporting Scripts and Data (file ESM_2.html). For running only the R snippets shown below and reproduce their results, the best option is to use the R script in file ESM_3.R of the Supporting Scripts and Data.

id	from	to	trans	Tstart	Tstop	time	status	strata	ASXL1	DNMT3A	[...]
77	1	2	1	0	2029	2029	0	1	0	0	.
77	1	3	2	0	2029	2029	1	2	0	0	.
78	1	2	1	0	332	332	1	1	1	0	.
78	1	3	2	0	332	332	0	2	1	0	.
78	2	4	3	332	1449	1117	1	3	1	0	.

Table 1: A 5-row fragment of the MDS data set (in long format)

5.1 Input data

Table 1 shows a fragment of the MDS data set. The data is in 'long format', which means that each row refers to a period of risk for a given transition and patient. For example, row i tells us that, at time `Tstart[i]`, patient `id[i]` entered state `from[i]`, and thereby began to be at risk for transition `trans[i]`, i.e., at risk of going from state `from[i]` to state `to[i]`. If the first transition of patient `id[i]` after time `Tstart[i]` occurs before the last follow-up time for this patient, `Tstop[i]` records the time of this transition (regardless of whether the patient moved to state `to[i]` or not). Otherwise, `Tstop[i]` is set to the last follow-up time. The value of `status[i]` is set to 1 if and only if the first transition of patient `id[i]` after `Tstart[i]` is to state `to[i]` and occurs before the last follow-up (otherwise it is set to 0). The value of `time[i]` is defined simply as `Tstop[i]`−`Tstart[i]`, and `strata[i]` is the stratum of the baseline hazard for transition `trans[i]` (more about this variable in the following section). For `x` ∈ { `ASXL1`, `DNMT3A`, ... }, `x[i]` denotes the level of covariate `x` between `Tstart[i]` and `Tstop[i]` in patient `id[i]`. (In the MDS data set, we assume that the relative hazard of a patient is determined by her covariate vector at $t = 0$, i.e., we assume all covariates to be time-fixed.) If a patient enters a new state, and this state communicates directly with n other states, then, as long as the patient actually spends time in the new state (i.e. the time of transition is not the same as the last follow-up time), n rows must be added to the data set, with each row corresponding to a different possible transition.

From table 1, we know that patient 77 entered state 1 ('MDS') at time 0 and remained in this state until time 2029, when she moved to state 3 ('death before AML'). There are no rows to describe the evolution of patient 77 after entering state 3, as this state is an absorbing state. As to patient 78, she remained in state 1 until time 332, and moved from there to state 2 ('AML'). She lived with AML for 1117 days and moved to state 4 ('death after AML') at time 1449.

5.2 Fitting an empirical Bayes Cox model

Once the data is in 'long format', the estimation of an empirical Bayes model can be carried out using the function `CoxRFX`. A simple example of the first argument of `CoxRFX`, denoted '`Z`', is a data frame gathering the `trans`, `strata` and covariate columns of the data in long format:

```
outcome_covs <- c("id","from","to","trans","Tstart","Tstop","time","status",
                  "strata")
Z <- mstate_data[!names(mstate_data) %in% outcome_covs]
#(`mstate_data' has the data in long format)
```

The `strata` column determines which baseline hazard functions are assumed to be equal. In table 1, each transition is assumed to have a (potentially) different baseline hazard. The model's assumptions regarding how covariates affect the hazard are reflected on the format of the covariate columns of `Z`. When the `Z` argument is the one created in the previous block of code, `CoxRFX` returns a single regression coefficient estimate for each covariate. In other words, the impact of any covariate is assumed to be the same for every transition.

There are however ways of relaxing this assumption. One can replace the `ASXL1` column in Z (or any other covariate column) by several 'type-specific' `ASXL1` columns: the `ASXL1` column specific for type i would show the mutation status of `ASXL1` in rows belonging to transition of type i, and show zero in all other rows. This would force `CoxRFX` to estimate a (potentially) different `ASXL1` coefficient for each transition type. This process of covariate expansion by type can be based on any partition of the set of transitions. When each type corresponds to a single transition, we refer to it simply as 'covariate expansion by transition'. The output shown below illustrates the effect of expanding the covariates in 'mstate_data' by transition.

```
# Columns `id' and `trans' from `mstate_data' together with the first
# two expanded covariates (patients 77 and 78):
    id trans ASXL1.1 ASXL1.2 ASXL1.3 DNMT3A.1 DNMT3A.2 DNMT3A.3  [...]
    77     1       0       0       0        0        0        0     .
```

```
77    2    0    0    0    0    0    0    .
78    1    1    0    0    0    0    0    .
78    2    0    1    0    0    0    0    .
78    3    0    0    1    0    0    0    .
```

The example code given below shows how to use **mstate** to expand covariates by transition and how to create a `Z` argument that makes `CoxRFX` estimate a regression coefficient for each covariate for transitions 1 and 2, and assume a fully non-parametric hazard for transition 3.

```
# To expand covariates by transition using mstate::expand.covs,
# first set the class of `mstate_data' as
class(mstate_data) <- c("data.frame","msdata")

# then add the transition matrix as attribute:
attr(mstate_data,"trans") <- tmat
#(`tmat' is the output of mstate::transMat)

# Expand covariates by transition:
covariates_expanded_123 <- mstate::expand.covs(
    mstate_data,
    covs = names(mstate_data)[! names(mstate_data) %in% outcome_covs],
    append = F
)

# remove all covariates for transition 3 from `covariates_expanded_123'
# to fit a fully non-parametric model on this transition:
covariates_expanded_12 <- covariates_expanded_123[
    !grepl(".3",names(covariates_expanded_123),fixed = T)
]

#argument `Z' of coxrfx
Z_12 <- data.frame(covariates_expanded_12,strata = mstate_data$trans,
                   trans = mstate_data$trans)
```

The second argument of `CoxRFX` ('`surv`') is a survival object that can easily be built by feeding the outcome variable columns of the data to the function `Surv` (from the package **survival**). Whether `CoxRFX` fits a clock-forward model or a clock-reset model depends on the kind of survival object:

```
#argument `surv' for a  clock-forward model
surv <- Surv(mstate_data$Tstart,mstate_data$Tstop,mstate_data$status)

#argument `surv' for a clock-reset model
surv <- Surv(mstate_data$time,mstate_data$status)
```

The argument `groups` of `CoxRFX` is a vector whose length equals the number of covariates in the data. In other words, the length of `groups` is `ncol(Z)-2`, since the argument `Z` must include both the covariate data and the `strata` and `trans` columns. If, for $i \neq j$, `groups[i]=groups[j]` = 'foo', this means that the regression coefficients of the i^{th} and j^{th} covariates of `Z` both belong to a group named 'foo' of coefficients with the same prior. For the `Z` object built above, the `groups` argument created in the following block of code embodies the assumption that all coefficients associated with a given transition have the same prior distribution. The final line of code fits the empirical Bayes model.

```
#argument `groups' of coxrfx
groups_12 <- paste0(rep("group",ncol(Z)-2),c("_1","_2"))

#fit random effects model
model_12 <- CoxRFX(Z_12,surv,groups_12,tmat)
```

Figure 10 shows regression coefficient point estimates for a clock-reset, empirical Bayes model fitted with the code above. Also shown are 95% non-parametric bootstrap confidence intervals computed using `ebmstate::boot_ebmstate`. The x-axis scale is logarithmic to allow estimates to be read as relative hazards more easily. For example, a mutation in *RUNX1* is associated with a twofold increase in the hazard of progression from MDS to AML, and treatment centre 4 is associated with a 3-fold increase in the hazard of dying before progressing to AML, when compared to the baseline value of 'treatment centre' (treatment centre = 2 or 5). In covariates that have been log-transformed (age, platelet count and neutrophil count) or logit-transformed (proportions of myeloblasts and ring sideroblasts in the bone marrow), the interpretation of estimates is different. For example, an increase

in age by a factor of e (≈ 2.72) almost triples the hazard of dying before AML; the same increase in the ratio $bm\_blasts/(1-bm\_blasts)$ (where bm_blasts is the proportion of myeloblasts in the bone marrow) is associated with an increment in the hazard of dying before AML of approximately 16%.

5.3 Computing cumulative transition hazard estimates

The function `msfit_generic` is the generic function in **ebmstate** that computes cumulative transition hazards for a given set of covariate values and an estimated Cox model. It calls a different method according to the class of its `object` argument. The default method corresponds to the original `msfit` function of the **mstate** package and is appropriate for objects of class `coxph`, i.e., objects that contain the fit of a Cox model with fixed effects. The other available method for `msfit_generic`, `msfit_generic.coxrfx`, is just the original `msfit` function, (slightly) adapted to deal with objects generated by `CoxRFX`. Quite importantly, `msfit_generic.coxrfx` does not allow the variance of the cumulative hazards to be computed, as this computation relies on asymptotic results which may not be valid for an empirical Bayes model. As a result, it only has two other arguments apart from the object of class `coxrfx`: a data frame with the covariate values of the patient whose cumulative hazards we want to compute; and a transition matrix describing the states and transitions in the model (such as the one that can be generated using `transMat` from the package **mstate**). The following block of code exemplifies how these objects can be built and generates the `msfit` object containing the cumulative transition hazard estimates for a sample patient. Note that the object with the patient data must include a row for each transition, as well as a column specifying the transition stratum of each row of covariates.

```
# Build `patient_data' data frame with the covariate values for which
# cumulative hazards are to be computed (covariate values of patient 78):
patient_data <- mstate.data[mstate.data$id == 78,,drop = F][rep(1,3),]
patient_data$strata <- patient_data$trans <- 1:3
patient_data <- mstate::expand.covs(
    patient_data,
    covs = names(patient_data)[ ! names(patient_data) %in% outcome_covs],
    append = T
)
patient_data <- patient_data[ ! grepl(".3",names(patient_data),fixed = T)]

# The `patient_data' data frame has only 3 rows (one for each transition).
# The output below shows its `id' and `trans' columns
# and expanded covariates ASXL1 and DNMT3A:
    id trans ASXL1.1 ASXL1.2 DNMT3A.1 DNMT3A.2  [...]
    78     1       1       0        0        0     .
    78     2       0       1        0        0     .
    78     3       0       0        0        0     .

# compute cumulative hazards
msfit_object_12 <- msfit_generic(model_12,patient_data,tmat)
```

Figure 11 shows three plots of estimated cumulative transition hazards for the sampled patient, one for each transition in the model, along with 95% non-parametric bootstrap confidence intervals (computed with `ebmstate::boot_ebmstate`). Throughout the plotted period, the 'slope' of the cumulative hazard (i.e., the hazard rate) for the MDS to AML transition is lower than the one for the MDS to death transition, and this in turn is lower than the one for the AML to death transition. It should be recalled that the cumulative hazard estimate is strictly non-parametric for this last transition, i.e., it is the same for all patients. The central plot of figure 11 suggests that, as time since diagnosis goes by, the hazard of dying in MDS increases (possibly an effect of age). On the other hand, the hazard of dying in AML seems to decrease (slightly) with time (rightmost plot). Conclusions regarding the evolution of the AML hazard are hard to draw, since the confidence intervals for the corresponding cumulative hazard curve are very wide (leftmost plot).

If an object generated by `msfit_generic` is fed to `plot`, and the package **mstate** is loaded, the method `mstate:::plot.msfit` will be called. This is an efficient way of automatically plotting the cumulative hazard estimates for all transitions, but confidence interval lines (separately estimated) cannot be added.

5.4 Computing state occupation probability estimates

The functions `probtrans_mstate`, `probtrans_ebmstate` and `probtrans_fft` compute estimates of state occupation probabilities for a given `msfit` object. All three functions generate objects of class `probtrans` that can be fed to the `plot.probtrans` method from the package **mstate**. The first of these functions should only be used for clock-forward models, as it relies on product-limit calculations. It calls the method `probtrans_mstate.default`, if the `msfit` object was generated by `msfit_generic.default`, or the method `probtrans_mstate.coxrfx`, if it was generated by `msfit_generic.coxrfx`. Both methods are identical to the function `probtrans` in the **mstate** package, with the reserve that `probtrans_mstate.coxrfx` does not allow the computation of the variances or covariances of the state occupation probability estimator.

The functions `probtrans_ebmstate` and `probtrans_fft` are the functions in **ebmstate** for the computation of state occupation probability estimates under clock-reset models with a transition structure that has no cycles. When using `probtrans_fft` (the faster, but somewhat less stable, of these two functions), three arguments must be supplied: the initial state of the process whose state occupation probabilities one wishes to compute, the `msfit` object, and the upper time limit for the generation of estimates (`max_time`). Both functions are based on a discrete-time approximation to a series of convolutions. The default argument `nr_steps` controls the number of (equally spaced) time steps used in this approximation. The arguments `max_time` and `nr_steps` should be increased until the estimated curves become stable.

The following line of code computes point estimates of state occupation probabilities for the sample patient.

```
probtrans_object_12 <- probtrans_fft("MDS",msfit_object_12, max_time = 4000)
```

Estimates are shown in figure 12, along with 95% non-parametric, bootstrap confidence intervals. For this particular patient, the estimated probability of being dead after AML remains below 0.4 throughout a period of 10 years from the MDS diagnosis; if the patient does reach AML, death is expected to happen quickly thereafter, as reflected in the very low estimates for the probability of being in AML at any point in time. The following block of code shows how to compute confidence intervals with `boot_ebmstate`:

```
# Creating the object arguments for boot_ebmstate()

# `groups' arguments was already created, but we need to add names to it
names(groups_12) <- names(covariates_expanded_12)

# `mstate_data_expanded' argument (similar to `covariates_expanded' but
# including outcome variables)
mstate_data_expanded <- cbind(
  mstate_data[names(mstate_data) %in% outcome_covs],
  covariates_expanded_12
)

# create the non-parametric bootstrap confidence intervals
boot_ebmstate_object <- boot_ebmstate(
  mstate_data = mstate_data_expanded,
  which_group = groups_12,
  min_nr_samples = 100,
  patient_data = patient_data,
  tmat = tmat,
  initial_state = "MDS",
  time_model = "clockreset",
  input_file = NULL,
  coxrfx_args = list(max.iter = 200),
  probtrans_args = list(max_time = 4000)
)
```

5.5 Model assessment

For any model fitted with **ebmstate**, two performance metrics can be easily computed: the *concordance* statistic (Harrell et al., 1982; see also the help page of `survival::concordance` for the definition of concordance) and the *Bayesian Information Criterion* (BIC) score (Schwarz, 1978). As an example of how these two metrics can be obtained and used for model comparison, suppose we wish to compare

'model_12' fitted above – which consists of a Cox regression including all covariates for transitions 1 and 2 and a fully non-parametric model for transition 3 – with a model that combines Cox regressions of all covariates for each of the three transitions (denoted 'model_123' below). The following code snippet shows how to fit this second model.

```
# arguments `groups' and `Z' for fitting a Cox regression model on all transitions
Z_123 <- data.frame(
    covariates_expanded_123,
    strata = mstate_data$trans,
    trans = mstate_data$trans
)
groups_123 <- paste0(rep("group", ncol(Z_123) - 2), c("_1", "_2", "_3"))

# Fit a Cox regression model for all transitions
model_123 <- CoxRFX(Z = Z_123, surv = surv, groups = groups_123)
```

Running the concordance function in the **survival** package for each model yields the following output:

```
> concordance(model_12)
    Call:
    concordance.coxph(object = model_12)

    n= 1210
    Concordance= 0.8131 se= 0.01314
                concordant discordant tied.x tied.y tied.xy
    strata=1      18040       2783      0      1       0
    strata=2      37919       9678      0      7       0
    strata=3          0          0   1052      0       4

> concordance(model_123)
    Call:
    concordance.coxph(object = model_123)

    n= 1210
    Concordance= 0.8168 se= 0.01312
                concordant discordant tied.x tied.y tied.xy
    strata=1      18041       2782      0      1       0
    strata=2      37920       9677      0      7       0
    strata=3        784        268      0      4       0
```

The output shows that modelling transition 3 with a Cox model, instead of a fully parametric one, has a negligible impact on the overall concordance. However, this is due to the fact that there are far fewer observations for this transition. The concordance for transition 3 only, which corresponds to strata 3, is 0.5 under the fully parametric model (i.e., all patients are assigned the same transition hazard) and considerably higher under the Cox regression ($784/(784+268) = 0.75$). Ideally, the comparison of models of different complexity should be carried out on a test sample rather than on the training data. For this purpose, the test data can be input into to the concordance function (argument newdata). However, in the present case, only 61 patients were ever at risk of dying with AML (i.e. of undergoing transition 3), and of these only 41 actually died, so we might prefer to keep all patients in the training data, rather than saving a fraction of them for testing purposes. Such an option will yield more accurate coefficient estimates, at the expense of not allowing the computation of unbiased estimates of model performance. If the goal is only to compare models, we can make do without test data, by using an information score that penalises model complexity, such as the BIC. To facilitate model comparison, the BIC score is one of the attributes of the model fit object:

```
> model_12$BIC
    [1] 2508.37
> model_123$BIC
    [1] 2483.49
```

The best model is the one with the lowest score, so the choice of 'model_123' is confirmed.

6 Discussion

We have shown that **ebmstate** is suitable for higher-dimensional, multi-state survival analysis, and that it is both efficient and easy-to-use. To a significant extent, the user-friendliness of **ebmstate** stems from

the fact that it was not built 'from the ground up'. Instead, we produced a package that is more easily accessible to the many users of **mstate** by taking advantage of whichever features of this package were useful to our method and by eliminating redundancies. The connection between **ebmstate** and **mstate** is based on the fact that the function `CoxRFX` takes the same type of input and produces the same type of output as `coxph` from the package `survival`, and the function `probtrans_fft` (or `probtrans_ebmstate`) has the same type of input and output as `probtrans` from **mstate** (as shown in figure 8).

We also sought to improve our package's user-friendliness by making it as efficient as possible. The reduction of computational cost is based on two features. First, our empirical Bayes method relies on an expectation-maximisation algorithm that estimates both the parameters and the hyper-parameters of the model, i.e., no further tuning of the model is required. Second, in **ebmstate**, the computation of state occupation probability estimates relies on analytical results rather than on simulation: not only for clock-forward models, where we import from **mstate** a product-limit estimator, but also for clock-reset models, where we implement our own estimator based on a convolution argument and the fast Fourier transform.

To our knowledge, **ebmstate** is the first R package to put together a framework for multi-state model estimation that is complete and suitable for higher-dimensional data. It does so by implementing point and interval estimators of regression coefficients, cumulative transition hazards and state occupation probabilities, under regularised multi-state Cox models. In section Estimator performance, the results of the simulation study suggest that for data sets with 100 patients or more and a ratio of p (patients) to n (coefficients per transition) greater than 0.1, the standard Cox model estimator is clearly outperformed by the empirical Bayes one when it comes to the estimation of relative hazards and state occupation probabilities of an out-of-sample patient, or the regression coefficients of the model. However, the same study suggests that using an empirical Bayes method instead of a fully non-parametric one is of limited or no value in settings where $p/n \geq 1$. This loss of usefulness can already happen for $p/n \leq 1/2$ when it comes to the estimation of the relative hazards of an out-of-sample patient, especially for transition structures with multiple competing transitions.

As mentioned in previous sections, **ebmstate** imports a product-limit estimator from **mstate** that targets the state occupation probabilities of patients with *time-fixed* covariate vectors. However, these estimators are extendible to models with time-dependent covariates, as long as these are external and the estimates are conditional on specific covariate paths (Aalen et al., 2008, p. 142). For piecewise constant covariates, it is likely that such an adaptation could be obtained by combining transition probability estimates obtained for each period in which the covariates are fixed. While no significant theoretical obstacles are foreseen in this matter, the computer implementation for more than a single piecewise constant covariate is likely to be a laborious task. We have left it therefore for future work.

Acknowledgements

The authors are supported by grant NNF17OC0027594 from the Novo Nordisk Foundation. We thank an anonymous reviewer for their constructive comments and helpful suggestions which led to a much-improved manuscript.

Supporting Scripts and Data

In the supporting Scripts and Data, the file 'ESM_1.html' contains additional simulation results and theoretical demonstrations. Additional details on the analysis of the MDS data set are given in the file 'ESM_2.html'. The MDS data set is in files 'MDS.TPD.20Nov2012.csv' and 'mds.paper.clin.txt'. The file 'ESM_3.R' contains a simplified R script to run the code snippets in the present paper. The **ebmstate** package is available on CRAN.

7 Conflict of interest

The authors have declared no conflict of interest.

References

O. Aalen, O. Borgan, and H. Gjessing. *Survival and event history analysis*. Springer, 2008. URL https://link.springer.com/book/10.1007/978-0-387-68560-1. [p15, 17, 18, 19, 28]

O. O. Aalen. A linear regression model for the analysis of life times. *Statistics in Medicine*, 8(8):907–925, 1989. URL https://doi.org/10.1002/sim.4780080803. [p15]

P. Andersen, O. Borgan, R. Gill, and N. Keiding. *Statistical Models Based On Counting Processes*. Springer, 1993. URL https://link.springer.com/book/10.1007/978-1-4612-4348-9. [p17, 18, 19]

B. Carlin and T. Louis. *Bayesian Methods for Data Analysis*. CRC Press, 2009. URL https://doi.org/10.1201/b14884. [p20]

G. Cortese and P. K. Andersen. Competing risks and time-dependent covariates. *Biometrical Journal*, 52 (1):138–158, 2010. URL https://doi.org/10.1002/bimj.200900076. [p18]

L. C. de Wreede, M. Fiocco, and H. Putter. The mstate package for estimation and prediction in non- and semi-parametric multi-state and competing risks models. *Computer Methods and Programs in Biomedicine*, 99(3):261 – 274, 2010. ISSN 0169-2607. doi: https://doi.org/10.1016/j.cmpb.2010.01.001. URL http://www.sciencedirect.com/science/article/pii/S0169260710000027. [p15, 18, 22]

L. C. de Wreede, M. Fiocco, and H. Putter. mstate: An R package for the analysis of competing risks and multi-state models. *Journal of Statistical Software*, 38(7):1–30, 2011. URL http://www.jstatsoft.org/v38/i07/. [p17]

A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin. *Bayesian Data Analysis*. CRC Press, 2014. URL https://doi.org/10.1201/b16018. [p20]

J. Harrell, Frank E., R. M. Califf, D. B. Pryor, K. L. Lee, and R. A. Rosati. Evaluating the Yield of Medical Tests. *JAMA*, 247(18):2543–2546, 05 1982. ISSN 0098-7484. doi: 10.1001/jama.1982.03320430047030. URL https://doi.org/10.1001/jama.1982.03320430047030. [p26]

T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009. URL https://link.springer.com/book/10.1007/978-0-387-84858-7. [p20]

T. Hothorn, P. Buehlmann, T. Kneib, M. Schmid, and B. Hofner. mboost: Model-based boosting. *R package version*, pages 2.9–3, 2020. URL https://CRAN.R-project.org/package=mboost. [p15]

P. Hougaard. Multi-state models: a review. *Lifetime data analysis*, 5(3):239–264, 1999. URL https://doi.org/10.1023/A:1009672031531. [p15]

C. Jackson. flexsurv: A platform for parametric survival modeling in R. *Journal of Statistical Software*, 70(8):1–33, 2016. doi: 10.18637/jss.v070.i08. [p15]

C. H. Jackson. Multi-state models for panel data: the msm package for R. *Journal of Statistical Software*, 38(8):1–29, 2011. URL http://www.jstatsoft.org/v38/i08/. [p15]

J. D. Kalbfleisch and R. L. Prentice. *The statistical analysis of failure time data*. John Wiley & Sons, 2002. doi: 10.1002/9781118032985. [p18]

A. Listwon and P. Saint-Pierre. SemiMarkov: An R Package for Parametric Estimation in Multi-State Semi-Markov Models. *Journal of Statistical Software*, 66(6):784, 2015. doi: 10.18637/jss.v066.i06. URL https://hal.archives-ouvertes.fr/hal-00860244. [p15]

E. Papaemmanuil, M. Gerstung, L. Malcovati, S. Tauro, G. Gundem, P. Van Loo, C. J. Yoon, P. Ellis, D. C. Wedge, A. Pellagatti, et al. Clinical and biological implications of driver mutations in myelodysplastic syndromes. *Blood*, 122(22):3616–3627, 2013. URL https://doi.org/10.1182/blood-2013-08-518886. [p22]

A. Perperoglou. Cox models with dynamic ridge penalties on time-varying effects of the covariates. *Statistics in Medicine*, 33(1):170–180, 2014. URL https://doi.org/10.1002/sim.5921. [p16, 18]

H. Putter. Tutorial in biostatistics: Competing risks and multi-state models analyses using the mstate package. *Companion file for the mstate package*, 2011. URL https://mirror.las.iastate.edu/CRAN/web/packages/mstate/vignettes/Tutorial.pdf. [p22]

H. Putter, M. Fiocco, and R. B. Geskus. Tutorial in biostatistics: competing risks and multi-state models. *Statistics in Medicine*, 26(11):2389–2430, 2007. URL https://doi.org/10.1002/sim.2712. [p22]

H. Reulen. gamboostmsm. *R package version*, page 1.1.87, 2014. URL https://CRAN.R-project.org/package=gamboostMSM. [p15]

H. Reulen. penmsm. *R package version*, page 0.99, 2015. URL https://CRAN.R-project.org/package=penMSM. [p15]

R. J. Samworth. Stein's paradox. *Eureka*, 62:38–41, 2012. URL https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7eebd55f569395544f2b5d367d6aee614901d2c1. [p21]

R. Schall. Estimation in generalized linear models with random effects. *Biometrika*, 78(4):719–727, 1991. doi: 10.1093/biomet/78.4.719. URL http://dx.doi.org/10.1093/biomet/78.4.719. [p16, 18]

G. Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978. URL https://www.jstor.org/stable/2958889. [p26]

C. Spitoni, M. Verduijn, and H. Putter. Estimation and asymptotic theory for transition probabilities in markov renewal multi-state models. *The International Journal of Biostatistics*, 8(1), 2012. doi: doi:10.1515/1557-4679.1375. URL https://doi.org/10.1515/1557-4679.1375. [p16]

T. M. Therneau. *A Package for Survival Analysis in S*, 2015. URL https://CRAN.R-project.org/package=survival. version 2.38. [p15]

H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005. doi: https://doi.org/10.1111/j.1467-9868.2005.00503.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2005.00503.x. [p16]

Rui J. Costa
European Molecular Biology Laboratory
European Bioinformatics Institute (EMBL-EBI)
Hinxton, CB10 1SD
United Kingdom
ruibarrigana@hotmail.com

Moritz Gerstung
aff. 1: European Molecular Biology Laboratory
European Bioinformatics Institute (EMBL-EBI)
Hinxton, CB10 1SD
United Kindom
aff. 2: German Cancer Research Center (DKFZ)
Im Neuenheimer Feld 280
69120 Heidelberg
Germany
moritz.gerstung@dkfz.de

Figures

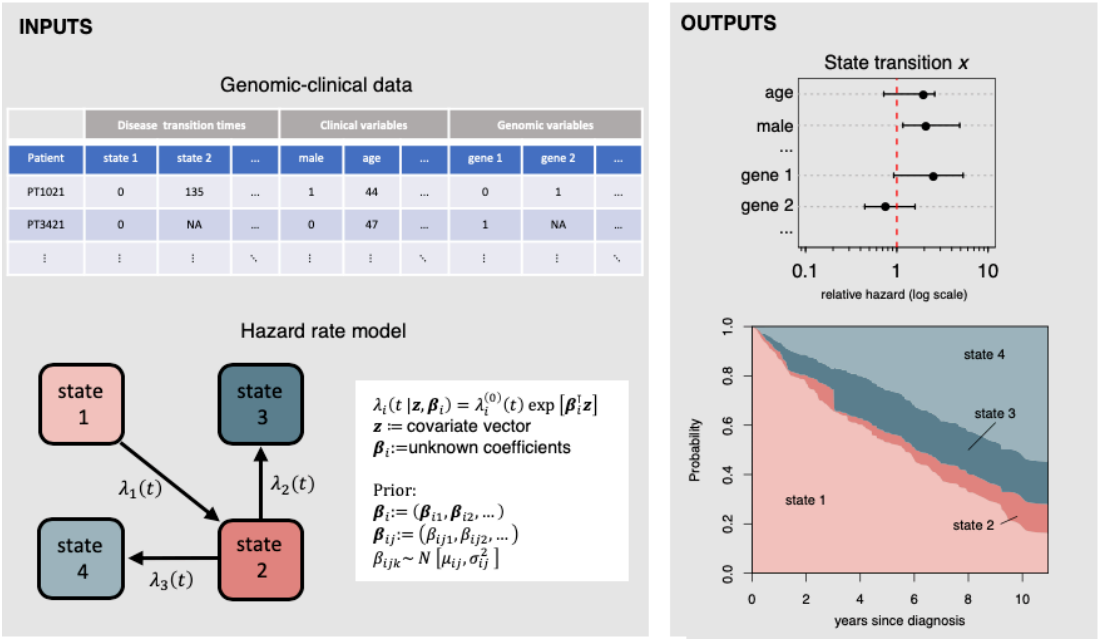


Figure 1: Summary of inputs and outputs of the package `ebmstate`. The input data set should be one that violates the assumption – commonly used in survival analysis – that the number of observations is much larger than the number of parameters to be estimated (a genomic-clinical data set is shown as a typical example). The input model is a multi-state Cox model defined by a transition structure and a prior distribution on the regression coefficients. This prior distribution is defined by partitioning the vector of regression coefficients into groups of regression coefficients, with each group having its own Gaussian prior with undetermined mean and variance. The outputs of `ebmstate` include estimates of the relative transition hazards associated with each covariate, as well as estimates of the probability that a specific patient (with specific covariate measurements) has of occupying each state of the model over some time period. Estimates of cumulative transition hazards are omitted from the figure.

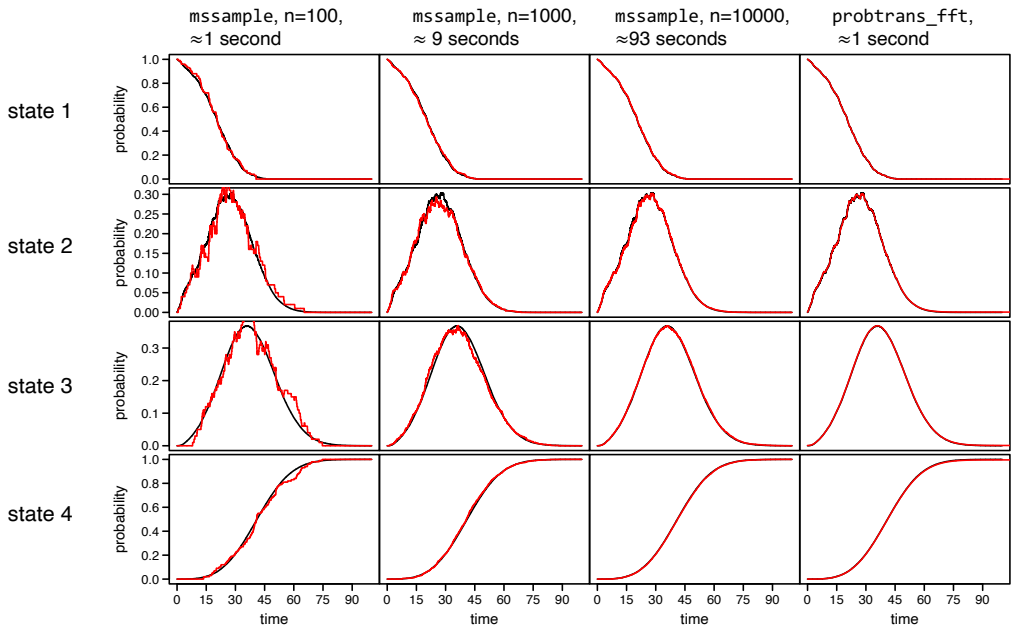


Figure 2: Comparison of running times and estimation accuracy of `mssample` and `probtrans_fft`. Each plot in the grid shows two estimated curves of state occupation probabilities. The black curves are based on a single run of `mstate::mssample` with $n = 100.000$ observations (approximately 17 minutes of running time) and are the same across columns. They serve as benchmark for precision assessment. In columns 1 to 3 of the grid, the superimposed red curves are based on a run of `mssample` with respectively 100, 1000, and 10.000 observations. In the rightmost column, the red curves are based on a run of `probtrans_fft`. All functions have as input the same set of cumulative transition hazards. These were estimated using a non-parametric multi-state model and a data set of 1000 patients generated according to a clock-reset Cox model with a 'linear' transition structure (leftmost diagram of figure 3). Plots in the same row refer to the same state of the model, while those in the same column refer to the same run of a function. Running times and, where appropriate, number of simulations (n) are given on top of each column.

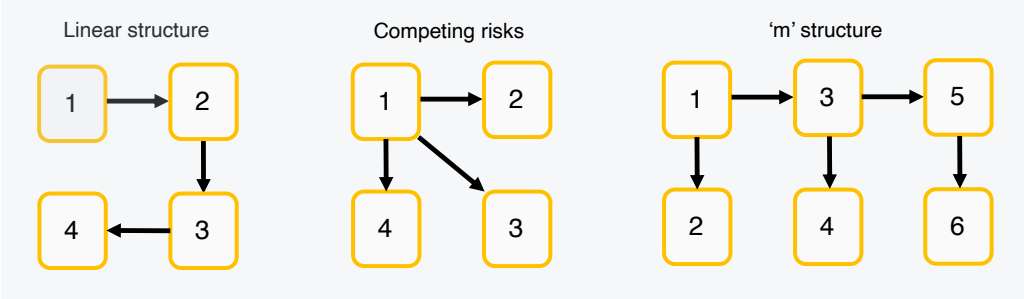


Figure 3: Model transition structures. We studied the performance of Cox model estimators, empirical Bayes Cox model estimators and fully non-parametric estimators with respect to these 3 transition structures.

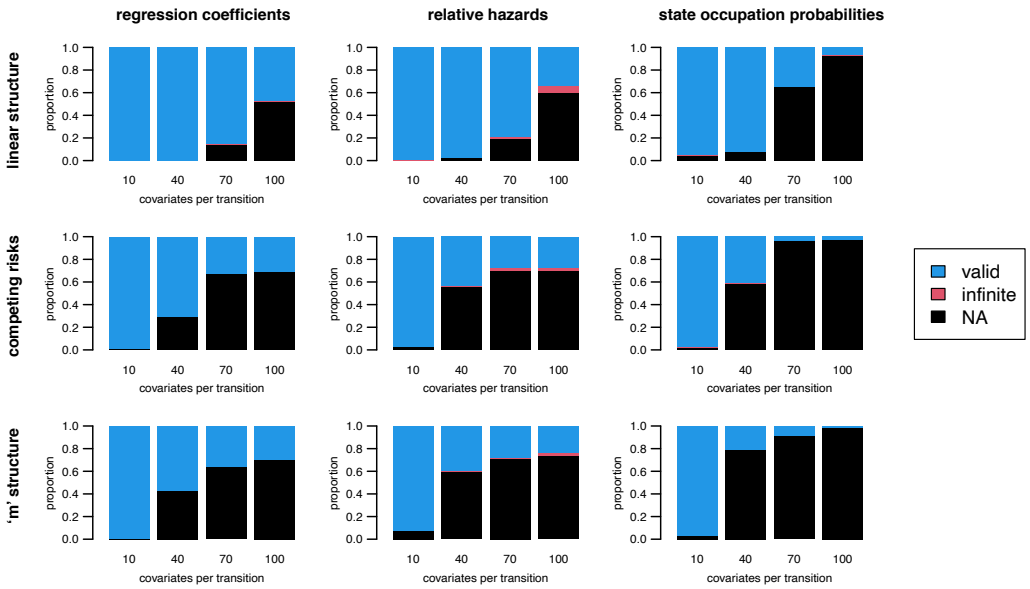


Figure 4: Proportions of valid, infinite and missing ('NA') estimates for the standard Cox model estimators in the simulation study of figure 6 (100 patients per simulated data set).

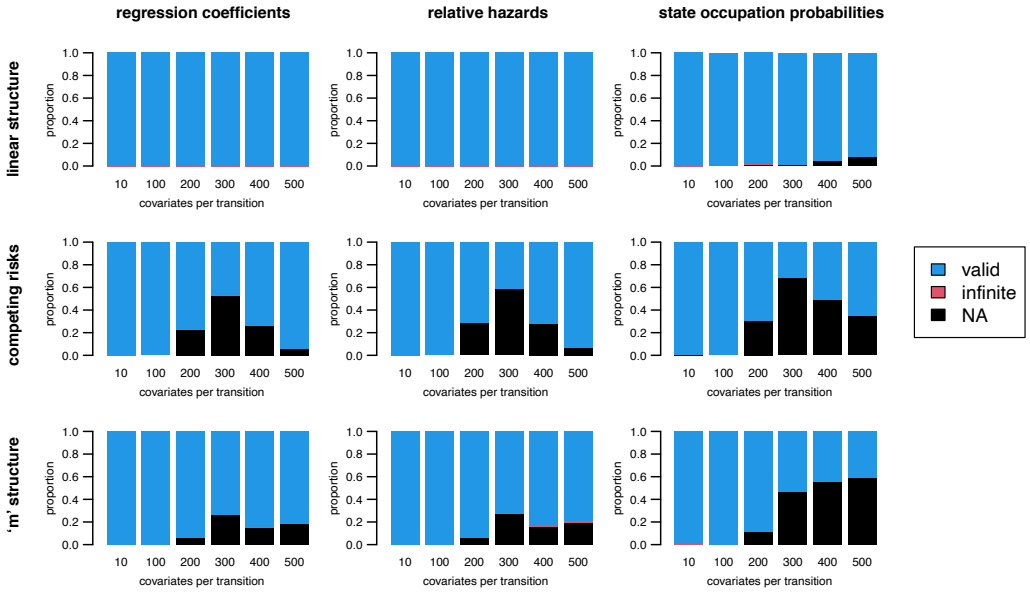


Figure 5: Proportions of valid, infinite and missing ('NA') estimates for the standard Cox model estimators in the simulation study of figure 7 (1000 patients per simulated data set).

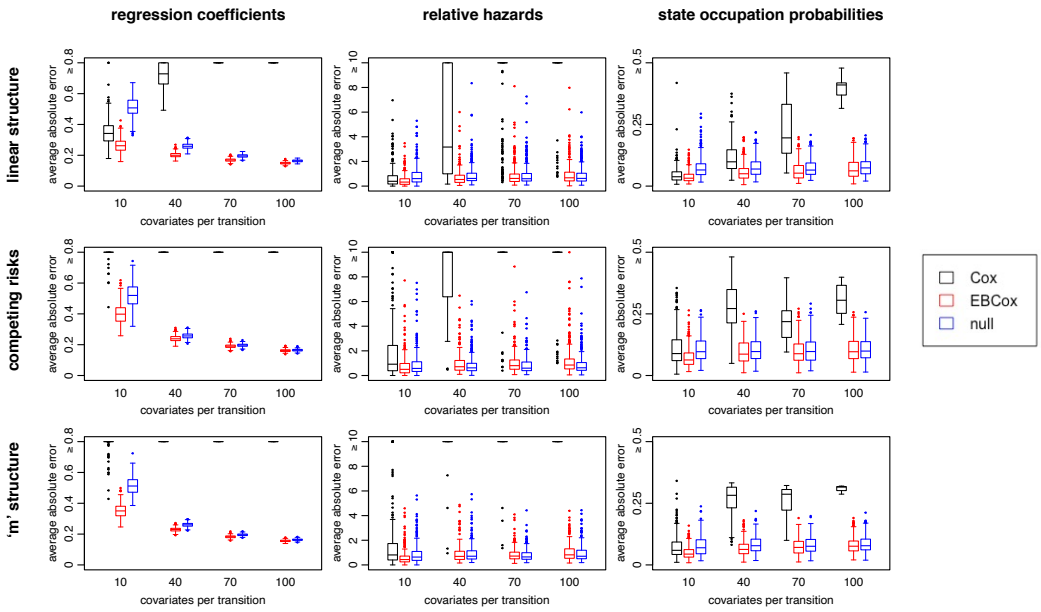


Figure 6: Performance comparison of standard Cox, empirical Bayes Cox, and fully non-parametric (null) estimators using training data sets with **100 observations** each. In the figure grid there is a boxplot corresponding to every tuple (a, m, G, p) such that $a \in \{$regression coefficients, relative hazards, state occupation probabilities$\}$ is the target of estimation, $m \in \{$standard Cox, empirical Bayes Cox, null$\}$ is the hazard model, $G \in \{$linear, competing risks, 'm' structure$\}$ is the transition structure of the model, and $p \in \{10, 40, 70, 100\}$ is the number of coefficients/covariates per transition. Each boxplot is based on at most 300 average absolute error observations. Figure 4, together with figures 6.1 and 6.3 in file ESM_1.html of the Supporting Scripts and Data, show the proportion of valid, missing and infinite estimates for each estimator. In each simulation scenario, the upper limit of the plot's y-axis defines a threshold above which observations are considered very large. Very large observations were replaced by the y-axis upper limit before the boxplots were built.

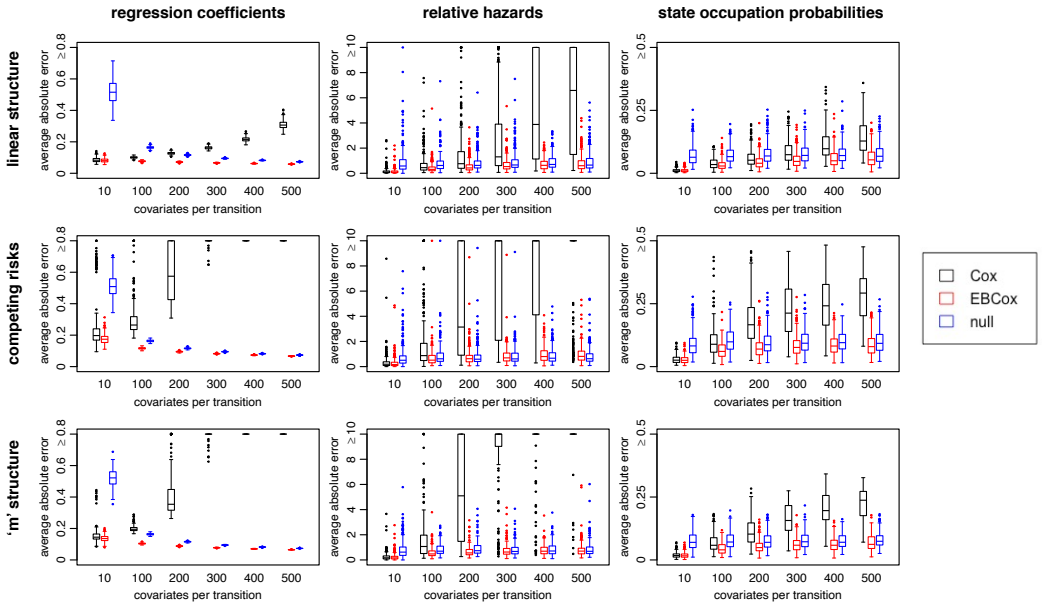


Figure 7: Performance comparison of standard Cox, empirical Bayes Cox, and fully non-parametric (null) estimators using training data sets with **1000 observations** each. In the figure grid there is a boxplot corresponding to every tuple (a, m, G, p) such that $a \in \{$regression coefficients, relative hazards, state occupation probabilities$\}$ is the target of estimation, $m \in \{$standard Cox, empirical Bayes Cox, null$\}$ is the hazard model, $G \in \{$linear, competing risks, 'm' structure$\}$ is the transition structure of the model, and $p \in \{10, 100, 200, 300, 400, 500\}$ is the number of coefficients/covariates per transition. Each boxplot is based on at most 300 average absolute error observations. Figure 5, together with figures 6.2 and 6.3 in file ESM_1.html of the Supporting Scripts and Data, show the proportion of valid, missing and infinite estimates for each estimator. In each simulation scenario, the upper limit of the plot's y-axis defines a threshold above which observations are considered very large. Very large observations were replaced by the y-axis upper limit before the boxplots were built.

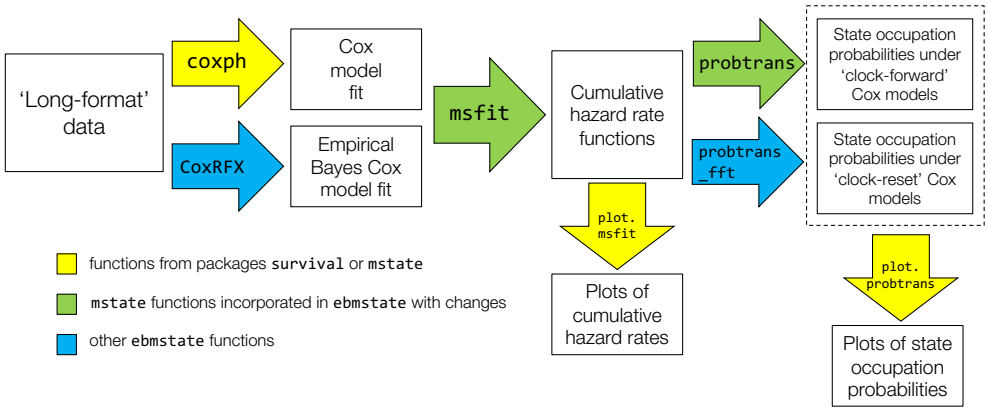
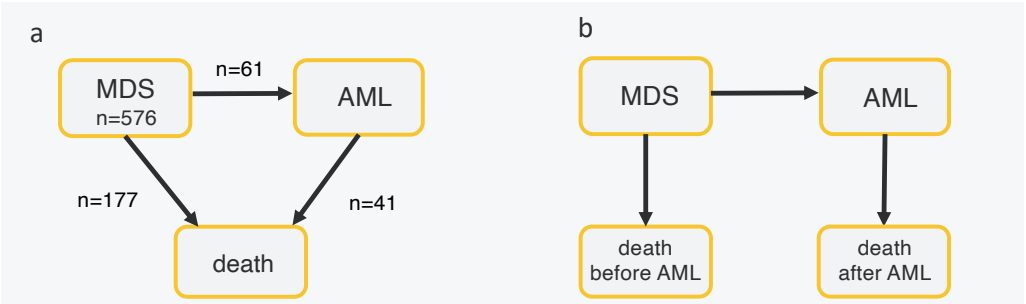


Figure 8: Extension of the `mstate` analysis framework by `ebmstate`. Arrows correspond to functions. Boxes correspond to inputs or outputs of functions. Functions `CoxRFX` and `probtrans_fft` from `ebmstate` compute point estimates only. Interval estimates can be obtained using the non-parametric bootstrap algorithm implemented in the function `ebmstate::boot_ebmstate`.



c

covariate	minimum	maximum	mean	std dev	data type	transformation
ASXL1 (point mutation)	0	1	0.13	0.33	binary (yes/no)	none
DNMT3A (point mutation)	0	1	0.09	0.28	binary (yes/no)	none
RUNX1 (point mutation)	0	1	0.07	0.26	binary (yes/no)	none
SF3B1 (point mutation)	0	1	0.27	0.45	binary (yes/no)	none
SRSF2 (point mutation)	0	1	0.14	0.35	binary (yes/no)	none
TET2 (point mutation)	0	1	0.25	0.43	binary (yes/no)	none
TP53 (point mutation)	0	1	0.05	0.22	binary (yes/no)	none
U2AF1 (point mutation)	0	1	0.06	0.23	binary (yes/no)	none
del5q mutation	0	1	0.1	0.31	binary (yes/no)	none
complex mutations	0	1	0.11	0.31	binary (yes/no)	none
other mutations	0	1	0.06	0.24	binary (yes/no)	none
age	17	95	67.42	12.61	numeric (years)	log
sex	0	1	0.44	0.5	binary (yes/no)	none
peripheral blood cytopenias	0	1	0.5	0.5	binary (yes/no)	none
hemoglobin level	3.2	17.6	9.88	1.96	numeric (g/dL)	none
absolute neutrophil count	0	38.2	3.08	3.65	numeric (count per nanolitre)	log
platelet count	5	1478	197	187.32	numeric (count per nanolitre)	log
myeloblasts in bone marrow	0.03	0.64	0.07	0.07	numeric (proportion)	logit
ring sideroblasts in bone marrow	0.03	0.93	0.17	0.23	numeric (proportion)	logit
International Prognostic Scoring System	1	4	1.79	0.84	ordinal (score)	none
WHO class RA	0	1	0.18	0.38	binary (yes/no)	none
WHO class RARS	0	1	0.13	0.34	binary (yes/no)	none
WHO class RCMD	0	1	0.16	0.37	binary (yes/no)	none
WHO class RCMD.RS	0	1	0.1	0.3	binary (yes/no)	none
WHO class RAEB	0	1	0.23	0.42	binary (yes/no)	none
WHO class CMML	0	1	0.1	0.3	binary (yes/no)	none
treatment center 1	0	1	0.12	0.33	binary (yes/no)	none
treatment center 3	0	1	0.54	0.5	binary (yes/no)	none
treatment center 4	0	1	0.28	0.45	binary (yes/no)	none
date of diagnosis	15/04/1981	18/01/2012	04/08/2004	NA	numeric	time since 15/04/1981 in units of 5 years

Figure 9: **a**: transition model implied by the data set of patients with myelodysplastic syndromes, together with transition event numbers; **b**: conversion to a transition structure without cycles; **c**: transformations applied to the MDS covariate data and summary statistics for the data before transformation. MDS stands for *myelodysplastic syndromes*; AML stands for *acute myeloid leukemia*.

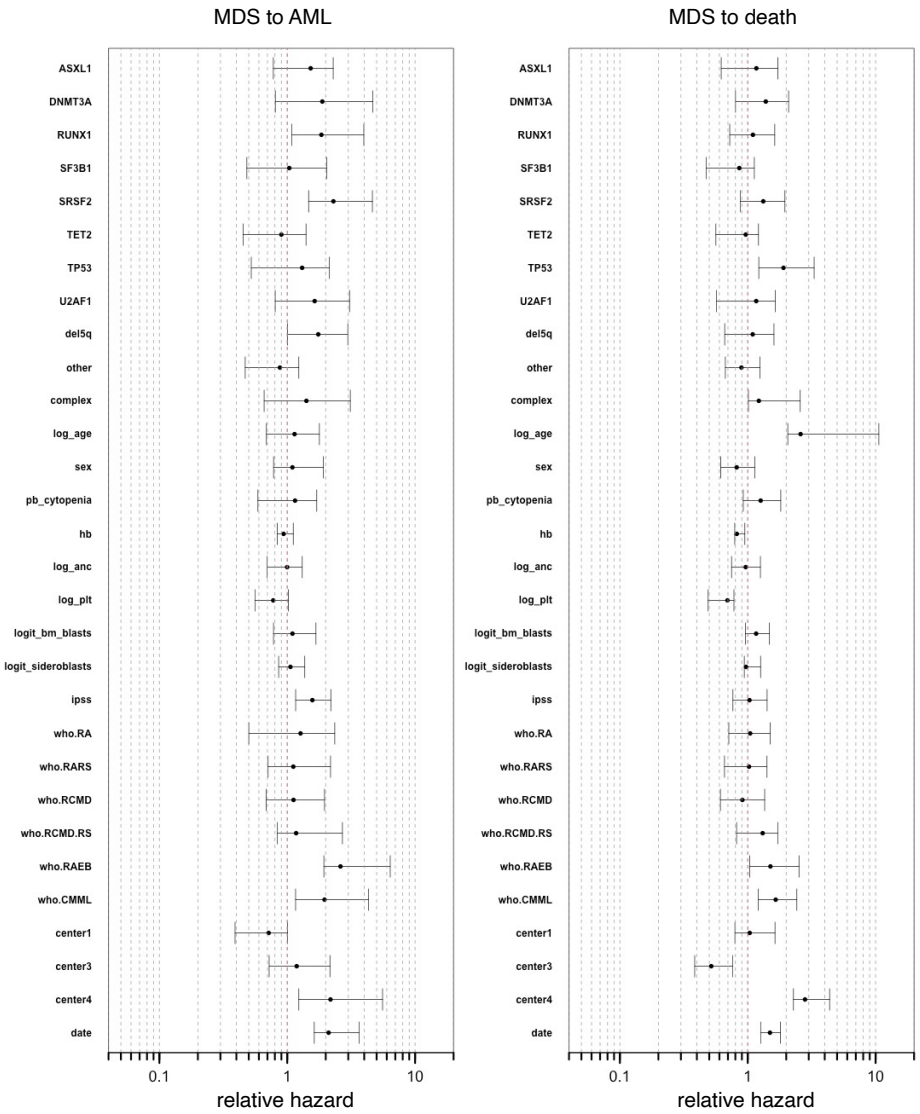


Figure 10: Point estimates of regression coefficients for the Cox model fitted to the MDS data, along with 95% non-parametric bootstrap confidence intervals. The x-axis scale is logarithmic so that coefficient estimates can be read as relative hazard estimates. If γ_{ij} is the element of $\hat{\boldsymbol{\beta}}_{ij}$ associated with a given covariate, $\exp\left(\gamma_{ij}\right)$ is the estimated relative hazard for this covariate in transition (i, j). In general, a relative hazard estimate r for a covariate z in transition (i, j) means that a one-unit increase in z is associated with an r-fold increase in the hazard of this transition. If z was obtained by log-transformation (as in age, platelet counts and neutrophil counts), a one-unit increase in z corresponds to scaling the original covariate by $e \approx 2.72$. In case z was obtained by logit-transformation (as in bone marrow blasts and sideroblasts proportions), the same one-unit increase corresponds to scaling the odds of the original covariate by e.

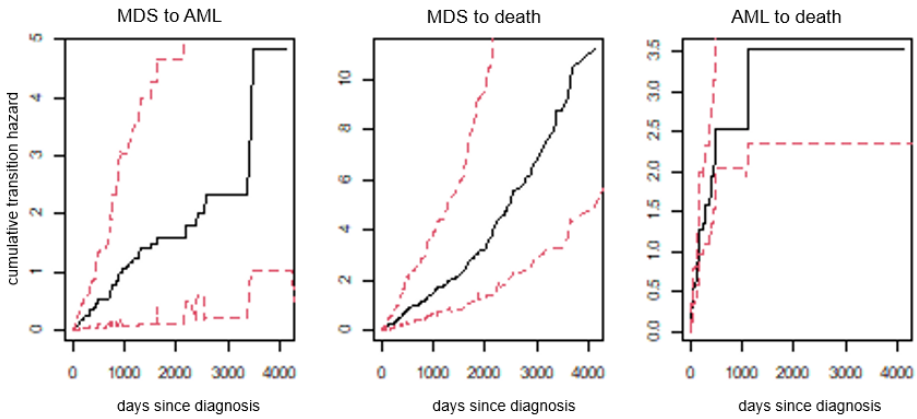


Figure 11: Point estimates of cumulative transition hazards for a sample patient with MDS (black curve), along with 95% non-parametric confidence intervals (dashed red lines).

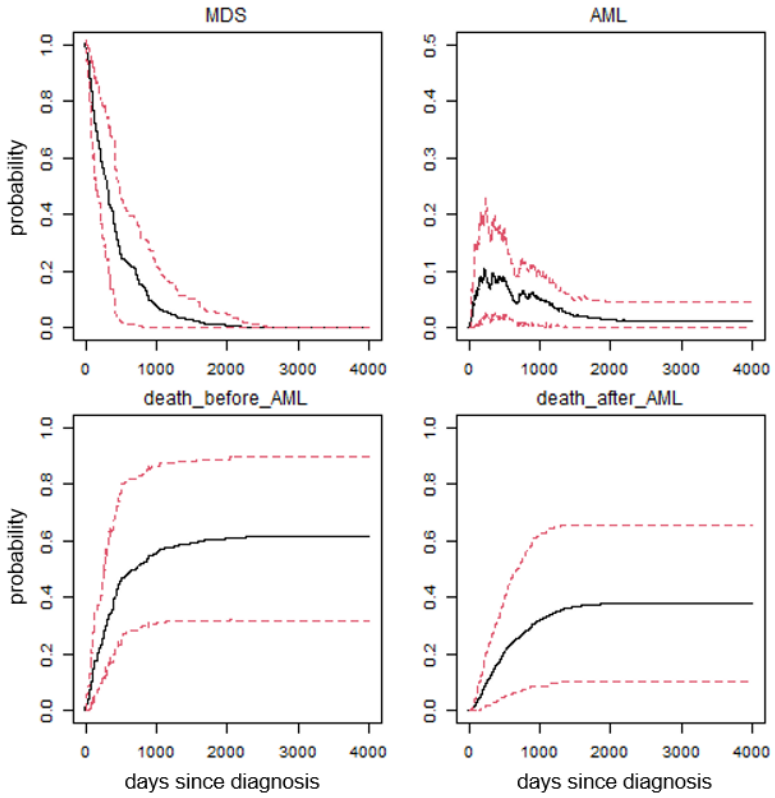


Figure 12: Point estimates of state occupation probabilities for a sample patient with MDS (black curve), along with 95% non-parametric confidence intervals (dashed red lines).

bootCT: An R Package for Bootstrap Cointegration Tests in ARDL Models

by Gianmarco Vacca, Maria Zoia, Stefano Bertelli

Abstract The Autoregressive Distributed Lag approach to cointegration or bound testing, proposed by Pesaran in 2001, has become prominent in empirical research. Although this approach has many advantages over the classical cointegration tests, it is not exempt from drawbacks, such as possible inconclusive inference and distortion in size. Recently, Bertelli and coauthors developed a bootstrap approach to the bound tests to overcome these drawbacks. This paper introduces the R package bootCT, which implements this method by deriving the bootstrap versions of the bound tests and of the asymptotic F-test on the independent variables proposed by Sam and coauthors in 2019. As a spinoff, a general method for generating random multivariate time series following a given VECM/ARDL structure is provided in the package. Empirical applications showcase the main functionality of the package.

1 Introduction

Cointegration and error correction are fundamental concepts in the analysis of economic data, insofar as they provide an appropriate framework for testing economic hypotheses about growth and fluctuation. Several approaches have been proposed in the literature to determine whether two or more non-stationary time series are cointegrated, meaning they share a common long-run relationship.
There are two basic types of tests for cointegration: single equation tests and VAR-based tests. The former check the presence of unit roots in cointegration residuals (see, e.g., Engle and Granger, 1987; Engle and Yoo, 1987; Mackinnon, 1991; Gabriel et al., 2002; Cook, 2006) or test the significance of the error-correction (EC) term coefficient (Kremers et al., 1992; Maddala and Kim, 1998; Arranz and Escribano, 2000; Ericsson and MacKinnon, 2002). The latter, such as the Johansen (1991) approach, tackle the problem of detecting cointegrating relationships in a VAR model. This latter approach, albeit having the advantage of avoiding the issue of normalization, as well as allowing the detection of multiple cointegrating vectors, is far from being perfect. In the VAR system all variables are treated symmetrically, as opposed to the standard univariate models that usually have a clear interpretation in terms of exogenous and endogenous variables. Furthermore, in a VAR system all the variables are estimated at the same time, which is problematic if the relation between some variables is flawed, that is affected by some source of error. In this case a simultaneous estimation process tends to propagate the error affecting one equation to the others. Furthermore, a multidimensional VAR models employs plenty of degrees of freedom.
The recent cointegration approach, known as Autoregressive Distributed Lag (ARDL) approach to cointegration or bound testing, proposed by Pesaran et al. (2001) (PSS), falls in the former strand of literature. It has become prominent in empirical research because it shows several advantages with respect to traditional methods for testing cointegration. First, it is applicable also in cases of mixed order integrated variables, albeit with integration not exceeding the first order. Thus, it evades the necessity of pre-testing the variables and, accordingly, avoids some common practices that may prevent finding cointegrating relationships, such as dropping variables or transforming them into stationary form (see McNown et al., 2018). Second, cointegration bound tests are performed in an ARDL model that allows different lag orders for each variable, thus providing a more flexible framework than other commonly employed approaches. Finally, unlike other cointegration techniques, which are sensitive to the sample size, the ARDL approach provides robust and consistent results for small sample sizes.
Notably, the ARDL bound testing methodology has quickly spread in economics and econometrics to study the cointegrating relationships between macroeconomic and financial variables, to evaluate the long-run impact of energy variables, or to assess recent environmental policies and their impact on the economy. Among the many applications, see for instance Haseeb et al. (2019); Reda and Nourhan (2020); Menegaki (2019); Yilanci et al. (2020); Hussain et al. (2019); Abbasi et al. (2021).
The original bound tests proposed by Pesaran et al. (2001) are an F-test for the significance of the coefficients of all lagged level variables entering the error correction term (F_{ov}), and a t-test for the coefficient of the lagged dependent variable. When either the dependent or the independent variables do not appear in the long-run relationship, a degenerate case arises. The bound t-test provides answers on the occurrence of a degenerate case of second type, while the occurrence of a degeneracy case of first type can be assessed by testing whether the dependent variable is of integration order I(1). This type of check violates the spirit and motivation of the bound tests, which are supposed to be applicable in situations of unknown order of integration for the variables.

Recently, McNown et al. (2018) pointed out how, due to the low power problem of unit root tests, investigating the presence of a first type degeneracy by testing the integration order of the dependent variable may lead to incorrect conclusions. Therefore, they suggested checking for its occurrence by testing the significance of the lagged levels of the independent variables via an extra F-test (F_{ind}), which was also worked out in its asymptotic version (SMK; Sam et al., 2019).
Besides problems in testing the occurrence of degenerate cases, in general, the main drawback of the bound tests is the occurrence of potentially inconclusive results, if the test statistic lies between the bounds of the test distribution under the null. Furthermore, the asymptotic distributions of the statistics may provide a poor approximation of the true distributions in small samples. Finite sample critical values, even if only for a subset of all possible model specifications, have been worked out in the literature (see Mills and Pentecost, 2001; Narayan and Smyth, 2004; Kanioura and Turner, 2005; Narayan, 2005), while Kripfganz and Schneider (2020) provided the quantiles of the asymptotic distributions of the tests as functions of the sample size, the lag order and the number of long-run forcing variables. However, this relevant improvement does not eliminate the uncertainty related to the inconclusive regions, or the existence of other critical issues related to the underlying assumptions of the bound test framework, such as the (weak) exogeneity of the independent variables or the non-stationarity of the dependent variable.
To overcome the mentioned bound test drawbacks, Bertelli et al. (2022) proposed bootstrapping the ARDL cointegration test. Inference can always be pursued with ARDL bootstrap tests, unlike what happens with both the PSS tests and the SMK test on the independent variables. Bootstrap ARDL tests were first put forward by McNown et al. (2018) in an unconditional ARDL model, which omits the instantaneous differences of the exogenous variables in the ARDL equation, rather than a conditional one, as originally proposed by Pesaran et al. (2001). The unconditional model is often used, for reason of practical convenience, in empirical research. Simulation results in Bertelli et al. (2022) have highlighted the importance of employing the appropriate specification, especially under degenerate cases. In fact, it has been pointed out that a correct detection of these cases requires the comparison of the test outcomes in both the conditional and unconditional settings. Erroneous conclusions, based exclusively on one model specification, can thus be avoided.
In this paper, bootstrap bound tests, thereby including the bootstrap versions of the F_{ov}, t and F_{ind} bound tests, are carried out in a conditional ARDL model setting. This approach allows to overcome the problem of inconclusive regions of the standard bound tests. A comparison with the outcomes engendered by the unconditional ARDL bootstrap tests is nevertheless provided for the F_{ind} test, to avoid erroneous inference in presence of degenerate cases.
The paper is organized as follows. Section 2.2 introduces the theoretical results of the ARDL cointegration bound tests. Section 2.3 details the steps carried out by the bootstrap procedure, which allows the construction of the (bootstrap) distribution - under the null - for the F_{ov}, t, conditional F_{ind} and unconditional F_{ind} tests. Section 2.4 introduces the R package **bootCT** (Vacca and Bertelli, 2023) and its functionalities: a method for the generation of random multivariate time series that follow a user-specified VECM/ARDL structure, with some examples, and the main function that carries out the aforementioned bootstrap tests, while also computing the PSS and SMK bound tests. The trade-off between accuracy and computational time of the bootstrap procedure is also investigated, under several scenarios in terms of sample size and number of replications. Notably, a function that performs the PSS bound tests is already available in the **dynamac** package (Jordan and Philips, 2020), while no R routine has so far been implemented for the SMK test, to the best of our knowledge. Section 2.5 gives some empirical applications that employ the core function of the package and its possible outputs. Section 2.6 concludes. Appendix 2.7 briefly delves into technical details of the conditional ARDL model and its possible specifications [1].

2 Cointegration bound tests in ARDL models

The starting point of the approach proposed by Pesaran et al. (2001) is a $(K+1)$ VAR(p) model

$$\mathbf{A}(L)(\mathbf{z}_t - \boldsymbol{\mu} - \boldsymbol{\eta} t) = \boldsymbol{\varepsilon}_t \quad \boldsymbol{\varepsilon}_t \sim N(\mathbf{0}, \boldsymbol{\Sigma}), \qquad \mathbf{A}(L) = \left(\mathbf{I}_{K+1} - \sum_{j=1}^{p} \mathbf{A}_j \mathbf{L}^j \right) \qquad t = 1, 2, \ldots, T. \quad (1)$$

[1] The R packages, either used in the creation of **bootCT** or employed in the analyses presented in this paper, are **magrittr** (Bache and Wickham, 2022), **gtools** (Bolker et al., 2022), **pracma** (Borchers, 2022), **Rcpp** (Eddelbuettel, 2013), **RcppArmadillo** (Eddelbuettel et al., 2023), **Rmisc** (Hope, 2022), **dynamac** (Jordan and Philips, 2020), **ARDL** (Natsiopoulos and Tzeremes, 2021), **aod** (Lesnoff et al., 2012), **vars** and **urca** (Pfaff, 2008a,b), **aTSA** (Qiu, 2015), **tseries** (Trapletti and Hornik, 2023), **reshape2**, **ggplot2** and **stringr** (Wickham, 2007, 2016, 2022), **tidyverse** and **dplyr** (Wickham et al., 2019, 2023).

Here, $\mathbf{A}_j$ are square $(K+1)$ matrices, $\mathbf{z}_t$ a vector of $(K+1)$ variables, $\boldsymbol{\mu}$ and $\boldsymbol{\eta}$ are $(K+1)$ vectors representing the drift and the trend respectively, and $\det(\mathbf{A}(z)) = 0$ for $|z| \geq 1$. If the matrix $\mathbf{A}(1) = \mathbf{I}_{K+1} - \sum_{j=1}^{p} \mathbf{A}_j$ is singular, the components of $\mathbf{z}_t$ turn out to be integrated and possibly cointegrated.
The VECM representation of (1) is given by (see Appendix 2.7.1 for details)

$$\Delta \mathbf{z}_t = \boldsymbol{\alpha}_0 + \boldsymbol{\alpha}_1 t - \mathbf{A}(1)\mathbf{z}_{t-1} + \sum_{j=1}^{p-1} \boldsymbol{\Gamma}_j \Delta \mathbf{z}_{t-j} + \boldsymbol{\varepsilon}_t. \tag{2}$$

Now, to study the adjustment to the equilibrium of a single variable y_t, given the other $\mathbf{x}_t$ variables, the vectors $\mathbf{z}_t$ and $\boldsymbol{\varepsilon}_t$ are partitioned

$$\mathbf{z}_t = \begin{bmatrix} y_t \\ (1,1) \\ \mathbf{x}_t \\ (K,1) \end{bmatrix}, \qquad \boldsymbol{\varepsilon}_t = \begin{bmatrix} \varepsilon_{yt} \\ (1,1) \\ \boldsymbol{\varepsilon}_{xt} \\ (K,1) \end{bmatrix}. \tag{3}$$

The matrix $\mathbf{A}(1)$, which is assumed to be singular to allow cointegration, is partitioned conformably to $\mathbf{z}_t$ as [2]

$$\mathbf{A}(1) = \begin{bmatrix} a_{yy} & \mathbf{a}'_{yx} \\ (1,1) & (1,K) \\ \mathbf{a}_{xy} & \mathbf{A}_{xx} \\ (K,1) & (K,K) \end{bmatrix}. \tag{4}$$

Under the assumption

$$\boldsymbol{\varepsilon}_t \sim N\left(\mathbf{0}, \begin{bmatrix} \sigma_{yy} & \boldsymbol{\sigma}'_{yx} \\ (1,1) & (1,K) \\ \boldsymbol{\sigma}_{xy} & \boldsymbol{\Sigma}_{xx} \\ (K,1) & (K,K) \end{bmatrix}\right), \tag{5}$$

the following holds

$$\varepsilon_{yt} = \boldsymbol{\omega}' \boldsymbol{\varepsilon}_{xt} + \nu_{yt} \sim N(0, \sigma_{y.x}), \tag{6}$$

where $\sigma_{y.x} = \sigma_{yy} - \boldsymbol{\omega}' \boldsymbol{\sigma}_{xy}$ with $\boldsymbol{\omega}' = \boldsymbol{\sigma}'_{yx} \boldsymbol{\Sigma}_{xx}^{-1}$, and ν_{yt} is independent of $\boldsymbol{\varepsilon}_{xt}$.
Substituting (6) into (2) and assuming that the $\mathbf{x}_t$ variables are exogenous towards the ARDL parameters (that is, setting $\mathbf{a}_{xy} = \mathbf{0}$ in $\mathbf{A}(1)$) yields the system (see Appendix 2.7.1 for details)

$$\Delta y_t = \alpha_{0.y} + \alpha_{1.y} t - a_{yy} EC_{t-1} + \sum_{j=1}^{p-1} \boldsymbol{\gamma}'_{y.x,j} \Delta \mathbf{z}_{t-j} + \boldsymbol{\omega}' \Delta \mathbf{x}_t + \nu_{yt} \tag{7}$$

$$\Delta \mathbf{x}_t = \boldsymbol{\alpha}_{0x} + \boldsymbol{\alpha}_{1x} t + \mathbf{A}_{(x)} \mathbf{z}_{t-1} + \boldsymbol{\Gamma}_{(x)}(L) \Delta \mathbf{z}_t + \boldsymbol{\varepsilon}_{xt}, \tag{8}$$

where

$$\boldsymbol{\gamma}'_{y.x,j} = \boldsymbol{\gamma}'_{y,j} - \boldsymbol{\omega}' \boldsymbol{\Gamma}_{(x),j} \tag{9}$$

$$\alpha_{0.y} = \alpha_{0y} - \boldsymbol{\omega}' \boldsymbol{\alpha}_{0x}, \qquad \alpha_{1.y} = \alpha_{1y} - \boldsymbol{\omega}' \boldsymbol{\alpha}_{1x}, \tag{10}$$

and where the error correction term, EC_{t-1}, expressing the long-run equilibrium relationship between y_t and $\mathbf{x}_t$, is given by

$$EC_{t-1} = y_{t-1} - \theta_0 - \theta_1 t - \boldsymbol{\theta}' \mathbf{x}_{t-1}, \tag{11}$$

with

$$\theta_0 = \mu_y - \boldsymbol{\theta}' \boldsymbol{\mu}_x, \;\; \theta_1 = \eta_y - \boldsymbol{\theta}' \boldsymbol{\eta}_x, \;\; \boldsymbol{\theta}' = -\frac{\widetilde{\mathbf{a}}'_{y.x}}{a_{yy}} = -\frac{\mathbf{a}'_{yx} - \boldsymbol{\omega}' \mathbf{A}_{xx}}{a_{yy}}. \tag{12}$$

Thus, no cointegration occurs when $\widetilde{\mathbf{a}}_{y.x} = \mathbf{0}$ or $a_{yy} = 0$. These two circumstances are referred to as degenerate case of second and first type, respectively. Degenerate cases imply no cointegration between y_t and $\mathbf{x}_t$.
To test the hypothesis of cointegration between y_t and $\mathbf{x}_t$, Pesaran et al. (2001) proposed an F-test, F_{ov} hereafter, based on the hypothesis system

$$H_0 : a_{yy} = 0 \;\cap\; \widetilde{\mathbf{a}}_{y.x} = \mathbf{0} \tag{13}$$

$$H_1 : a_{yy} \neq 0 \;\cup\; \widetilde{\mathbf{a}}_{y.x} \neq \mathbf{0}. \tag{14}$$

[2] If the explanatory variables are stationary $\mathbf{A}_{xx}$ is non-singular ($rk(\mathbf{A}_{xx}) = K$), while when they are integrated but without cointegrating relationship $\mathbf{A}_{xx}$ is a null matrix.

Note that H_1 covers also the degenerate cases

$$H_1^{y.x} : a_{yy} = 0 \ , \ \widetilde{\mathbf{a}}_{y.x} \neq \mathbf{0} \tag{15}$$

$$H_1^{yy} : a_{yy} \neq 0 \ , \ \widetilde{\mathbf{a}}_{y.x} = \mathbf{0}. \tag{16}$$

The exact distribution of the F statistic under the null is unknown, but it is limited from above and below by two asymptotic distributions: one corresponding to the case of stationary regressors, and another corresponding to the case of first-order integrated regressors. As a consequence, the test is called bound test and has an inconclusive area. [3]
Pesaran et al. (2001) worked out two sets of (asymptotic) critical values: one, $\{\tau_{L,F}\}$, for the case when $\mathbf{x}_t \sim I(0)$ and another, $\{\tau_{U,F}\}$, for the case when $\mathbf{x}_t \sim I(1)$. These values vary in accordance with the number of regressors in the ARDL equation, the sample size and the assumptions made about the deterministic components (intercept and trend) of the data generating process.
In this regard, Pesaran et al. (2001) introduced five different specifications for the ARDL model, depending on its deterministic components, which are (see Appendix 2.7.2 for details)

I. *No intercept and no trend*

$$\Delta y_t = -a_{yy} EC_{t-1} + \sum_{j=1}^{p-1} \boldsymbol{\gamma}'_{y.x,j} \Delta \mathbf{z}_{t-j} + \boldsymbol{\omega}' \Delta \mathbf{x}_t + \nu_{yt}, \tag{17}$$

where $EC_{t-1} = y_{t-1} - \boldsymbol{\theta}' \mathbf{x}_{t-1}$,

II. *Restricted intercept and no trend*

$$\Delta y_t = -a_{yy} EC_{t-1} + \sum_{j=1}^{p-1} \boldsymbol{\gamma}'_{y.x,j} \Delta \mathbf{z}_{t-j} + \boldsymbol{\omega}' \Delta \mathbf{x}_t + \nu_{yt}, \tag{18}$$

where $EC_{t-1} = y_{t-1} - \theta_0 - \boldsymbol{\theta}' \mathbf{x}_{t-1}$. The intercept extracted from the EC term is $\alpha_{0.y}^{EC} = a_{yy}\theta_0$.

III. *Unrestricted intercept and no trend*

$$\Delta y_t = \alpha_{0.y} - a_{yy} EC_{t-1} + \sum_{j=1}^{p-1} \boldsymbol{\gamma}'_{y.x,j} \Delta \mathbf{z}_{t-j} + \boldsymbol{\omega}' \Delta \mathbf{x}_t + \nu_{yt}, \tag{19}$$

where $EC_{t-1} = y_{t-1} - \boldsymbol{\theta}' \mathbf{x}_{t-1}$.

IV. *Unrestricted intercept, restricted trend*

$$\Delta y_t = \alpha_{0.y} - a_{yy} EC_{t-1} + \sum_{j=1}^{p-1} \boldsymbol{\gamma}'_{y.x,j} \Delta \mathbf{z}_{t-j} + \boldsymbol{\omega}' \Delta \mathbf{x}_t + \nu_{yt}, \tag{20}$$

where $EC_{t-1} = y_{t-1} - \theta_1 t - \boldsymbol{\theta}' \mathbf{x}_{t-1}$. The trend extracted from the EC term is $\alpha_{1.y}^{EC} = a_{yy}\theta_1$.

V. *Unrestricted intercept, unrestricted trend*

$$\Delta y_t = \alpha_{0.y} + \alpha_{1.y} t - a_{yy} EC_{t-1} + \sum_{j=1}^{p-1} \boldsymbol{\gamma}'_{y.x,j} \Delta \mathbf{z}_{t-j} + \boldsymbol{\omega}' \Delta \mathbf{x}_t + \nu_{yt}, \tag{21}$$

where $EC_{t-1} = y_{t-1} - \boldsymbol{\theta}' \mathbf{x}_{t-1}$.

The model in (7) proposed by Pesaran et al. (2001) represents the correct framework in which to carry out bound tests. However, bound test are often performed in an unconditional ARDL model setting, specified as

$$\Delta y_t = \alpha_{0.y} + \alpha_{1.y} t - a_{yy} EC_{t-1} + \sum_{j=1}^{p-1} \boldsymbol{\gamma}'_{j} \Delta \mathbf{z}_{t-j} + \varepsilon_{yt}, \tag{22}$$

which omits the term $\boldsymbol{\omega}' \Delta \mathbf{x}_t$.
Bertelli et al. (2022) have highlighted that bootstrap tests performed in these two ARDL specifications can lead to contrasting results. To explain this divergence, note that the conditional model makes use of the following vector in the EC term

$$\widetilde{\mathbf{a}}'_{y.x} = \mathbf{a}'_{yx} - \boldsymbol{\omega}' \mathbf{A}_{xx} \tag{23}$$

[3] The knowledge of the rank of the cointegrating matrix is necessary to overcome this impasse.

(divided by a_{yy}, see (12)) to carry out bound tests, while the unconditional one only uses the vector $\mathbf{a}'_{yx}$, (divided by a_{yy}), since it neglects the term $\boldsymbol{\omega}'\mathbf{A}_{xx}$. [4] This can lead to contrasting inference in two instances. The first happens when a degeneracy of first type occurs in the conditional model, that is

$$\widetilde{\mathbf{a}}'_{y.x} = \mathbf{0}, \tag{24}$$

because

$$\mathbf{a}'_{yx} = \boldsymbol{\omega}'\mathbf{A}_{xx}. \tag{25}$$

In this case, the conditional model rejects cointegration, while the unconditional one concludes the opposite. The other case happens when a degeneracy of first type occurs in the unconditional model, that is

$$\mathbf{a}'_{yx} = \mathbf{0}, \tag{26}$$

but

$$\widetilde{\mathbf{a}}'_{y.x} = \boldsymbol{\omega}'\mathbf{A}_{xx} \neq \mathbf{0}. \tag{27}$$

In this case, the unconditional model rejects cointegration, while the conditional one concludes for the existence of cointegrating relationships, which are however spurious. Only a comparison of the outcomes of the F_{ind} test performed in both the conditional and unconditional ARDL equation can help to disentangle this problem. [5]

In the following, bootstrap tests are carried out in the conditional ARDL model (7). However, when a degeneracy of first type occurs in the unconditional model, the outcomes of the F_{ind} bootstrap test performed in both the conditional and unconditional settings are provided. This, as previously outlined, is performed to avoid the acceptance of spurious long-run relationships among the dependent variable and the independent variables.

3 The new bootstrap procedure

The bootstrap procedure here proposed focuses on a ARDL model specified as in (17)-(21), depending on the assumptions on the deterministic components.

The bootstrap procedure consists of the following steps:

1. The ARDL model is estimated via OLS and the related test statistics F_{ov}, t or F_{ind} are computed.
2. In order to construct the distribution of each test statistic under the corresponding null, the same model is re-estimated imposing the appropriate restrictions on the coefficients according to the test under consideration.
3. Following McNown et al. (2018), the ARDL restricted residuals are then computed. For example, under Case III, the residuals are

$$\widehat{v}^{F_{ov}}_{yt} = \Delta y_t - \widehat{\alpha}_{0.y} - \sum_{j=1}^{p-1} \widehat{\gamma}'_{y.x,j}\Delta\mathbf{z}_{t-j} - \widehat{\boldsymbol{\omega}}'\Delta\mathbf{x}_t \tag{28}$$

$$\widehat{v}^{t}_{yt} = \Delta y_t - \widehat{\alpha}_{0.y} + \widehat{\widetilde{\mathbf{a}}}'_{y.x}\mathbf{x}_{t-1} - \sum_{j=1}^{p-1} \widehat{\gamma}'_{y.x,j}\Delta\mathbf{z}_{t-j} - \widehat{\boldsymbol{\omega}}'\Delta\mathbf{x}_t \tag{29}$$

$$\widehat{v}^{F_{ind}}_{yt} = \Delta y_t - \widehat{\alpha}_{0.y} + \widehat{a}_{yy}y_{t-1} - \sum_{j=1}^{p-1} \widehat{\gamma}'_{y.x,j}\Delta\mathbf{z}_{t-j} - \widehat{\boldsymbol{\omega}}'\Delta\mathbf{x}_t. \tag{30}$$

   Here, the apex "$\widehat{\cdot}$" denotes the estimated parameters. The other cases can be dealt with in a similar manner.
4. The VECM model

$$\Delta\mathbf{z}_t = \boldsymbol{\alpha}_0 - \mathbf{A}\mathbf{z}_{t-1} + \sum_{j=1}^{p-1} \boldsymbol{\Gamma}_j\Delta\mathbf{z}_{t-j} + \boldsymbol{\varepsilon}_t \tag{31}$$

   is estimated as well (imposing weak exogeneity), and the residuals

$$\widehat{\boldsymbol{\varepsilon}}_{xt} = \Delta\mathbf{x}_t - \widehat{\boldsymbol{\alpha}}_{0x} + \widehat{\mathbf{A}}_{xx}\mathbf{x}_{t-1} - \sum_{j=1}^{p-1} \widehat{\boldsymbol{\Gamma}}_{(x)j}\Delta\mathbf{z}_{t-j} \tag{32}$$

[4] The latter is introduced in the ARDL equation by the operation of conditioning y_t on the other variables $\mathbf{x}_t$ of the model

[5] In fact, as $\boldsymbol{\omega}'\mathbf{A}_{xx}\mathbf{x}_t \approx I(0)$, the conclusion that $y_t \approx I(0)$ must hold. This in turn entails that no cointegration occurs between y_t and $\mathbf{x}_t$.

are computed. This approach guarantees that the residuals $\widehat{\varepsilon}_{xt}$, associated to the variables $\mathbf{x}_t$ explained by the marginal model (8), are uncorrelated with the ARDL residuals $\widehat{v}_{yt}$.

5. A large set of B bootstrap replicates are sampled from the residuals calculated as in (28),(29), (30) and (32). In each replication, the following operations are carried out:

   (a) Each set of $(T-p)$ resampled residuals (with replacement) $\widehat{\boldsymbol{v}}_{zt}^{(b)} = (\widehat{v}_{yt}^{(b)}, \widehat{\boldsymbol{\varepsilon}}_{xt}^{(b)})$ is re-centered (see Davidson and MacKinnon, 2005)

   $$\dot{\widehat{v}}_{yt}^{(b)} = \widehat{v}_{yt}^{(b)} - \frac{1}{T-p}\sum_{t=p+1}^{T}\widehat{v}_{yt}^{(b)} \tag{33}$$

   $$\dot{\widehat{\boldsymbol{\varepsilon}}}_{x_i t}^{b} = \widehat{\boldsymbol{\varepsilon}}_{x_i t}^{(b)} - \frac{1}{T-p}\sum_{t=p+1}^{T}\widehat{\boldsymbol{\varepsilon}}_{x_i t}^{(b)} \qquad i = 1,\dots,K. \tag{34}$$

   (b) A sequential set of $(T-p)$ bootstrap observations, y_t^* , $\mathbf{x}_t^*$ $t = p+1,\dots,T$, is generated as follows

   $$y_t^* = y_{t-1}^* + \Delta y_t^*, \quad \mathbf{x}_t^* = \mathbf{x}_{t-1}^* + \Delta \mathbf{x}_t^*, \tag{35}$$

   where $\Delta\mathbf{x}_t^*$ are obtained from (32) and Δy_t^* from either (28), (29) or (30) after replacing in each of these equations the original residuals with the bootstrap ones.
   The initial conditions, that is the observations before $t = p+1$, are obtained by drawing randomly p observations in block from the original data, so as to preserve the data dependence structure.

   (c) An unrestricted ARDL model is estimated via OLS using the bootstrap observations, and the statistics $F_{ov}^{(b),H_0}$, $t^{(b),H_0}$ $F_{ind}^{(b),H_0}$ are computed.

6. The bootstrap distributions of $\{F_{ov}^{(b),H_0}\}_{b=1}^{B}$, $\{F_{ind}^{(b),H_0}\}_{b=1}^{B}$ and $\{t^{(b),H_0}\}_{b=1}^{B}$ under the null are then employed to determine the critical values of the tests. By denoting with M_b^* the ordered bootstrap test statistic, and with α the nominal significance level, the bootstrap critical values are determined as follows

   $$c_{\alpha,M}^* = \min\left\{c : \sum_{b=1}^{B}\mathbf{1}_{\{M_b^* > c\}} \leq \alpha\right\} \qquad M \in \{F_{ov}, F_{ind}\} \tag{36}$$

   for the F tests and

   $$c_{\alpha,t}^* = \max\left\{c : \sum_{b=1}^{B}\mathbf{1}_{\{t_b^* < c\}} \leq \alpha\right\} \tag{37}$$

   for the t test.
   Here, $\mathbf{1}_{\{x\in A\}}$ is the indicator function, which is equal to one if the condition in subscript is satisfied and zero otherwise.

The null hypothesis is rejected if the F statistic computed at step 1, F_{ov} or F_{ind}, is greater than the respective $c_{\alpha,M}^*$, or if the t statistic computed at the same step is lower than $c_{\alpha,t}^*$.

4 Illustration of the bootCT package

This section describes the main functionalities of the **bootCT** package. The functions included in the package are essentially of two types. The function `sim_vecm_ardl` generates data according to a given data generating process (DGP), assuming either the presence or the absence of cointegrating relationships between variables, or degenerate cases. The function `boot_ardl` tests the presence of cointegrating relationships employing the Pesaran ARDL bound tests (F_{ov} and t), the SMK bound test on lagged independent variables (F_{ind}), and the novel ARDL bootstrap testing procedure.

4.1 Generating a multivariate time series: the `sim_vecm_ardl` function

The function `sim_vecm_ardl` allows to simulate a multivariate time series from a given conditional ARDL specification for a dependent variable y_t and a VAR/VECM specification for the remaining independent variables $\mathbf{x}_t$. In this regard, it represents an interesting addition to extant data generating procedures for VAR/VECM models. The arguments of this function can be divided into two subgroups. A group of parameters pertains the VECM model (7) and (8), with $\mathbf{A}_{xx}$ identifying the matrix of the long-run relationships among the $\mathbf{x}_t$ variables, and $\boldsymbol{\Gamma}_j$'s, $j = 1, ..., p-1$ the short-run matrices of the system variables. Additionally, the parameter a_{yy} weighs the EC term for y_t, while $\mathbf{a}_{yx}'$ is the parameter

vector weighting the variables $\mathbf{x}_t$ in the ARDL equation. The vector $\mathbf{a}'_{yx}$, after conditioning y_t on the other variables ($\mathbf{x}_t$, see model 7) becomes $\widetilde{\mathbf{a}}'_{y.x} = \mathbf{a}'_{yx} - \boldsymbol{\omega}'\mathbf{A}_{xx}$.
The second group of parameters concerns the model intercept and trend of the VAR specification, $\boldsymbol{\mu}$ and $\boldsymbol{\eta}$, which in the VECM representation become $\boldsymbol{\alpha}_0 = \mathbf{A}\boldsymbol{\mu} + (\mathbf{I}_{K+1} - \sum_{i=1}^{p-1}\boldsymbol{\Gamma}_j - \mathbf{A})\boldsymbol{\eta}$ and $\boldsymbol{\alpha}_1 = \mathbf{A}\boldsymbol{\eta}$ and in the conditional ARDL become $\alpha_{0.y}^{EC} = a_{yy}(\mu_y - \widetilde{\mathbf{a}}'_{y.x}\boldsymbol{\mu}_x) + \boldsymbol{\gamma}'_{y.x}(1)\boldsymbol{\eta}$ and $a_{1.y}^{EC} = a_{yy}(\eta_y - \widetilde{\mathbf{a}}'_{y.x}\boldsymbol{\eta}_x)$.
As explained in Appendix 2.7.2, intercept and trend appear in the error correction (EC) term of the ARDL equation only when restricted. Accordingly, they both do not appear in the EC in the case I, the intercept does not appear in the EC term in cases III, IV and V (it is freely set to $\alpha_{0.y}$) while the trend appears in the EC term only in the case IV (it is freely set to $\alpha_{1.y}$ for case V). Accordingly, when these terms are not restricted, they need to be supplied by the user.
The approach used to specify the function inputs offers great control to the user, in terms of generating specific (conditional) ARDL-based cointegration structures.
The function `sim_vecm_ardl` takes the following arguments:

- `nobs`: number of observations to generate;
- `case`: indicates the conditional ARDL specification in terms of deterministic component (intercept and trend) among the five specifications proposed by Pesaran et al. (2001), given in (17)-(21).
- `sigma.in`: covariance matrix, $\boldsymbol{\Sigma}$, of the error term $\boldsymbol{\varepsilon}_t$;
- `gamma.in`: list of short-run parameter matrices $\boldsymbol{\Gamma}_j$;
- `axx.in`: cointegrating relationships, $\mathbf{A}_{xx}$, pertaining the independent variables in the marginal VECM model;
- `ayx.uc.in`: vector of parameters, as in $\mathbf{a}_{yx}$;
- `ayy.in`: the a_{yy} term, weighting the EC term in the ARDL equation;
- `mu.in`: mean vector, $\boldsymbol{\mu}$, in the starting VAR specification, used to define the VECM intercept for CASE II;
- `eta.in`: trend vector, $\boldsymbol{\eta}$, in the starting VAR specification, used to define the VECM trend for case IV;
- `azero.in`: unrestricted intercept of the VECM specification (valid only for cases III, IV and V), when the intercept is not involved in the EC term;
- `aone.in`: unrestricted coefficient of the trend in the VECM specification (valid only for case V), when the trend is not involved in the EC term;
- `burn.in`: additional observations burn-in observations to be generated. A total of `burn.in + nobs` observations are generated, but only the last `nobs` are kept in the data;
- `seed.in`: seed number for the generation of $\boldsymbol{\varepsilon}_t \sim N(\mathbf{0}, \boldsymbol{\Sigma})$.

If parameter values for `mu.in`, `eta.in`, `azero.in`, or `aone.in` and case number turn out to be in contradiction, an error message is displayed.
As output, the function gives out a list containing the data, both in level and first difference, along with all the parameter values given as input. Additionally, all intermediate transformation of parameters via VECM transformation or as a by-product of conditioning y_t on $\mathbf{x}_t$ are included in the output.
Figure 1 depicts three-time series, `dep_1_0`, `ind_1_0` and `ind_2_0`, generated using this function and affected by a cointegrating relationship, one panel for each case, from I to V. The variable `dep_1_0` represents the dependent variable y_t of the ARDL equation, while `ind_1_0` and `ind_2_0` the independent ones, x_{1t} and x_{2t}.
The code used to generate the data for case I is the following:

```
corrm = matrix(c(   0,     0, 0,
                 0.25,     0, 0,
                  0.4, -0.25, 0), nrow = 3, ncol = 3, byrow = T)

Corrm = (corrm + t(corrm)) + diag(3)

sds = diag(c(1.3, 1.2, 1))

sigma.in = (sds %*% Corrm %*% t(sds))

gamma1 = matrix(c(0.6,    0, 0.2,
                  0.1, -0.3,   0,
                    0, -0.3, 0.2), nrow = 3, ncol = 3,byrow=T)
gamma2= gamma1 * 0.3
```

```
omegat = sigma.in[1, -1] %*% solve(sigma.in[-1, -1])
axx.in = matrix(c( 0.3, 0.5,
                  -0.4, 0.3), nrow = 2, ncol = 2, byrow = T)
ayx.uc.in = c(0.4, 0.4)
ayy.in = 0.6

data.vecm.ardl_1 =
sim_vecm_ardl(nobs = 200,
              case = 1,
              sigma.in = sigma.in,
              gamma.in = list(gamma1, gamma2),
              axx.in = axx.in,
              ayx.uc.in = ayx.uc.in,
              ayy.in = ayy.in,
              mu.in = rep(0, 3),
              eta.in = rep(0, 3),
              azero.in = rep(0, 3),
              aone.in = rep(0, 3),
              burn.in = 100,
              seed.in = 999)
```

Additionally, Figure 2 displays other three time series, `dep_1_0` (y_t), `ind_1_0` (x_{1t}) and `ind_2_0` (x_{2t}), when a degeneracy of second type occurs ($a_{yy} = 0$) in the long-run relationship in the ARDL equation of `dep_1_0` on `ind_1_0`, `ind_2_0`. The five panels represents the behavior of these series in the Cases from I to V. It is worth noting the different scenario implied by these cases: case III depicts a trend for the y_t variable, case IV highlights the inclusion of a trend in the cointegrating relationship, and case V exhibits a quadratic trend in the y_t variable.
Finally, the flowchart in Figure 3 details the internal steps of the function `sim_vecm_ardl` and the data generation workflow. There, it is specified how the parameters of the VAR, VECM and ARDL equation are introduced. Attention is paid on whether the error correction mechanism involves either intercept or trend (or both) via the internal computation of the parameters θ_0 and θ_1 (and thus $\alpha_{0.y}^{EC}$ and $\alpha_{1.y}^{EC}$). When the EC term does not involve intercept and/or trend, $\boldsymbol{\alpha}_0$ and $\boldsymbol{\alpha}_1$ are supplied by the user, depending on the case under study.

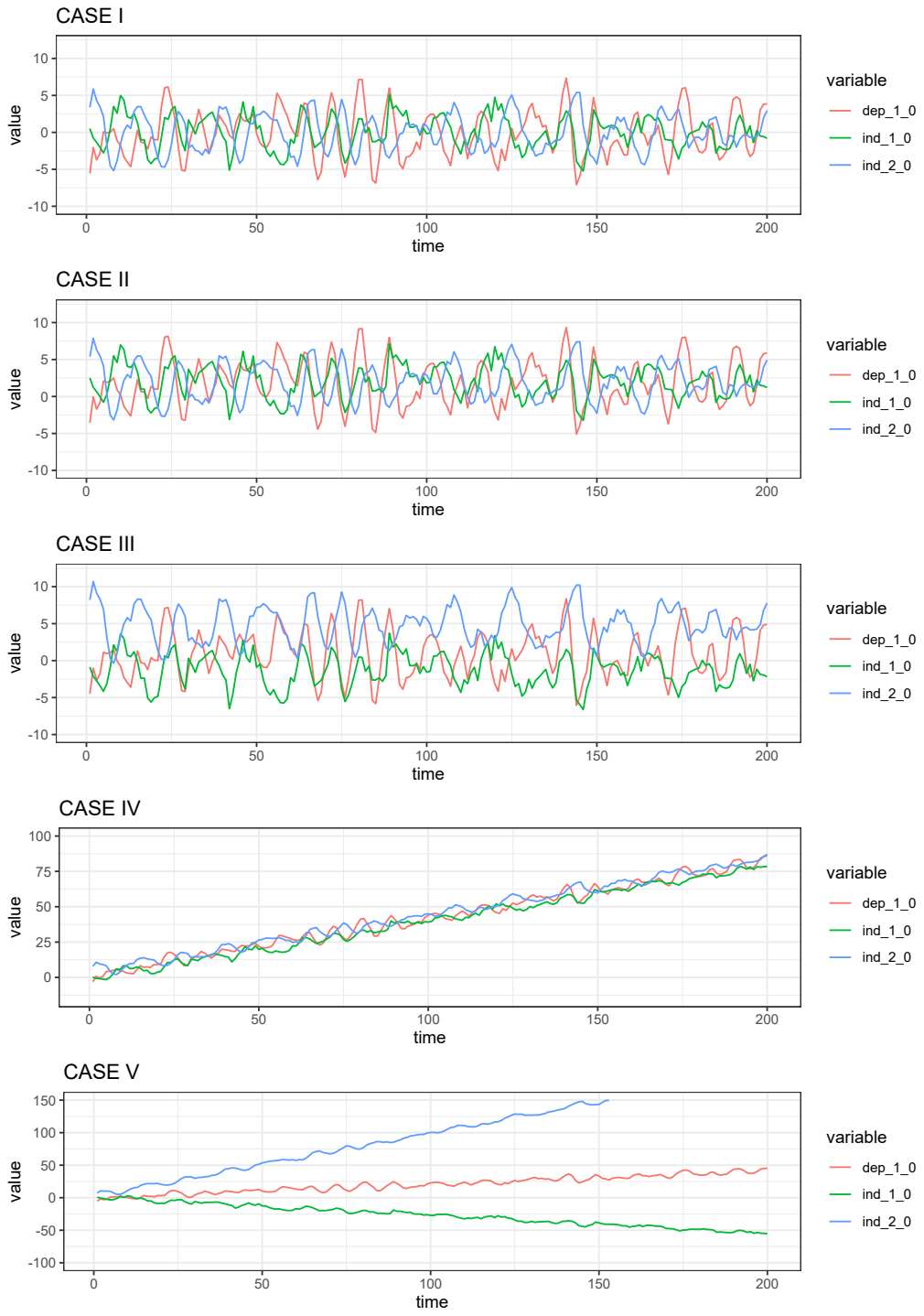


Figure 1: Simulated data from the VECM / conditional ARDL specifications, for every case. Made with **ggplot** (Wickham, 2016).

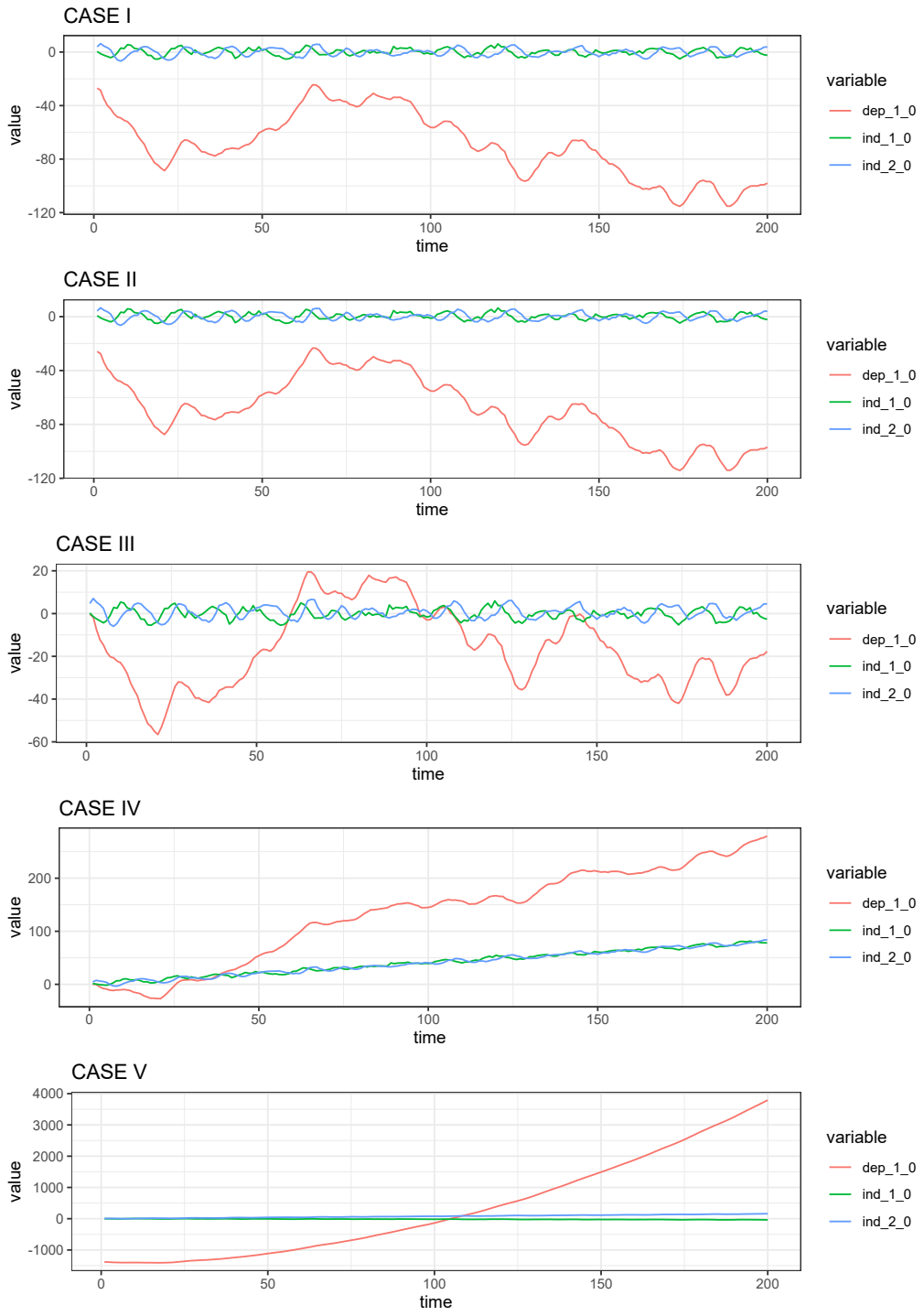


Figure 2: Simulated data from the VECM / conditional ARDL specifications (degenerate case of type 2, $a_{yy} = 0$), for every case. Made with **ggplot**.

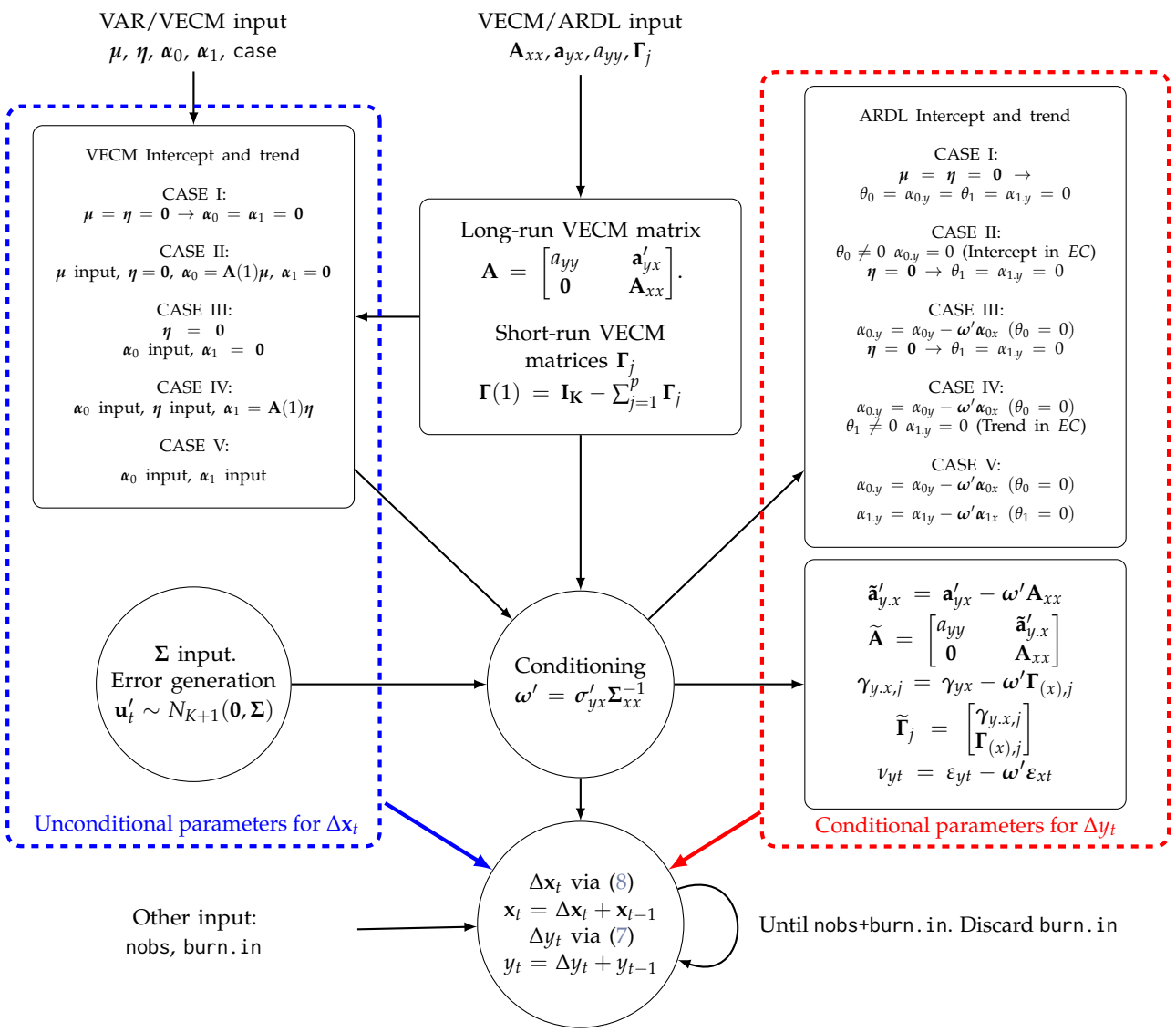


Figure 3: Flowchart of the `sim_vecm_ardl` function inner steps. When applying (7) and (8), $y_{t_j} = 0$, $\Delta y_{t_j} = 0$, $\mathbf{x}_{t_j} = \mathbf{0}$, $\Delta\mathbf{x}_{t_j} = \mathbf{0}$ for any $t_j < 1$. Boxes denote parameter definitions and transformations. Circles denote crucial actions, Empty nodes denote function inputs.

4.2 Bootstrapping the ARDL bound tests: the `boot_ardl` function

This function develops the bootstrap procedure detailed previously. As an option in the initial estimation phase, it offers the possibility of automatically choosing the best order for the lagged differences of all the variables in the ARDL and VECM models. This is done by using several criteria. In particular, AIC, BIC, AICc, R^2 and R^2_{adj} are used as lag selection criteria for the ARDL model, while the overall minimum between AIC, HQIC, SC and FPE is used for the lag selection for the VECM.
In particular, the `auto_ardl` function in the package **ARDL** (Natsiopoulos and Tzeremes, 2021) selects the best ARDL order in terms of the short-run parameter vectors $\gamma_{y.x,j}$, while the `VARselect` function in the package **vars** (Pfaff, 2008a) selects the best VECM order in terms of the short-run parameter matrices $\mathbf{\Gamma}_{(x),j}$. Furthermore, the user can input a significance threshold for the retention of single parameters in the $\mathbf{\Gamma}_j$ and in the $\gamma_{y.x,j}$ vectors.
The function `boot_ardl` takes the following arguments:

- `data`: input dataset. Must contain a dependent variable and a set of independent variables;
- `yvar`: name of the dependent variable enclosed in quotation marks. If unspecified, the first variable in the dataset is used;
- `xvar`: vector of names of the independent variables, each enclosed in quotation marks. If unspecified, all variables in the dataset except the first are used;
- `fix.ardl`: vector $(j_1, \ldots, j_K)$, containing the maximum orders of the lagged differences (i.e., $\Delta y_{t-j_1}, \Delta x_{1,t-j_2}, \ldots, \Delta x_{1,t-j_K}$) for the short term part of the ARDL equation, chosen in advance;
- `info.ardl`: (alternatively to `fix.ardl`) the information criterion used to choose the best lag order for the short term part of the ARDL equation. It must be one between `AIC` (default), `AICc`, `BIC`, `R2`, , `adjR2`;
- `fix.vecm`: scalar m containing the maximum order of the lagged differences (i.e., $\Delta \mathbf{z}_{t-m}$) for the short term part of the VECM equation, chosen in advance;
- `info.vecm`: (alternatively to `fix.vecm`) the information criterion used to choose the best lag order for the short term part of the VECM equation. Must be one among `AIC` (default), `HQIC`, `SC`, `FPE`;
- `maxlag`: (in conjunction with `info.ardl` / `info.vecm`) maximum number of lags for the `auto_ardl` function in the package **ARDL**, and for the `VARselect` function in the package **vars**;
- `a.ardl`: significance threshold for the short-term ARDL coefficients ($\gamma_{y.x,j}$) in the ARDL model estimation;
- `a.vecm`: significance threshold for the short-term VECM coefficients (in $\mathbf{\Gamma}_j$) in the VECM model estimation;
- `nboot`: number of bootstrap replications;
- `case`: type of the specification for the conditional ARDL in terms of deterministic components (intercept and trend) among the five proposed by Pesaran et al. (2001), given in (17)-(21);
- `a.boot.H0`: probability/ies α by which the critical quantiles of the bootstrap distribution(s) $c^*_{\alpha,F_{ov}}$, $c^*_{\alpha,t}$ and $c^*_{\alpha,F_{ind}}$ must be calculated;
- `print`: if set to `TRUE`, shows the progress bar.

`boot_ardl` makes use of the `lag_mts` function which produces lagged versions of a given matrix of time series, each column with a separate order. `lag_mts` takes as parameters the data included in a matrix `X` and the lag orders in a vector `k`, with the addition of a boolean parameter `last.only`, which allows to specify whether only the k-th order lags have to be retained, or all the lag orders from the first to the k-th.
`boot_ardl` also acts as a wrapper for the most common methodologies detecting cointegration, offering a comprehensive view on the testing procedures involved in the analysis. The resulting object, of class `bootCT`, contains all the information about

- The conditional ARDL model estimates, and the unconditional VECM model estimates;
- the bootstrap tests performed in the conditional ARDL model;
- the Pesaran, Shin and Smith bound testing procedure (F_{ov} and t-test, when applicable);
- the Sam, McNown and Goh bound testing procedure for F_{ind}, when applicable;
- the Johansen rank and trace cointegration tests on the independent variables.

Internally, the bootstrap data generation under the null is executed via a `Rcpp` function, employing the **Rcpp** and **RcppArmadillo** packages (Eddelbuettel, 2013), so as to greatly speed up computational times. As explained in the previous section, cointegration tests in the unconditional ARDL model are performed in order to uncover the presence of spurious cointegrating relationships.
To this end, the function provides

- the bootstrap critical values of the F_{ov}, t and F_{ind} tests in the conditional model, at level `a.boot.H0`, along with the same statistics computed in the conditional model.
- a flag, called `fakecoint`, that indicates divergence between the outcomes of the F_{ind} test performed in both the conditional and unconditional model. In this circumstance, as explained before, there is no cointegration (see Bertelli et al., 2022).

A `summary` method has been implemented to present the results in a visually clear manner. It accepts the additional argument "`out`" that lets the user choose which output(s) to visualize: `ARDL` prints the conditional ARDL model summary, `VECM` prints the VECM model summary, `cointARDL` prints the summary of the bound tests and the bootstrap tests, `cointVECM` prints the summary of the Johansen test on the independent variables.
A detailed flowchart showing the function's workflow is displayed in Figure 4. There, the expressions "C ARDL" and "UC ARDL" stand for conditional and unconditional ARDL model, respectively.

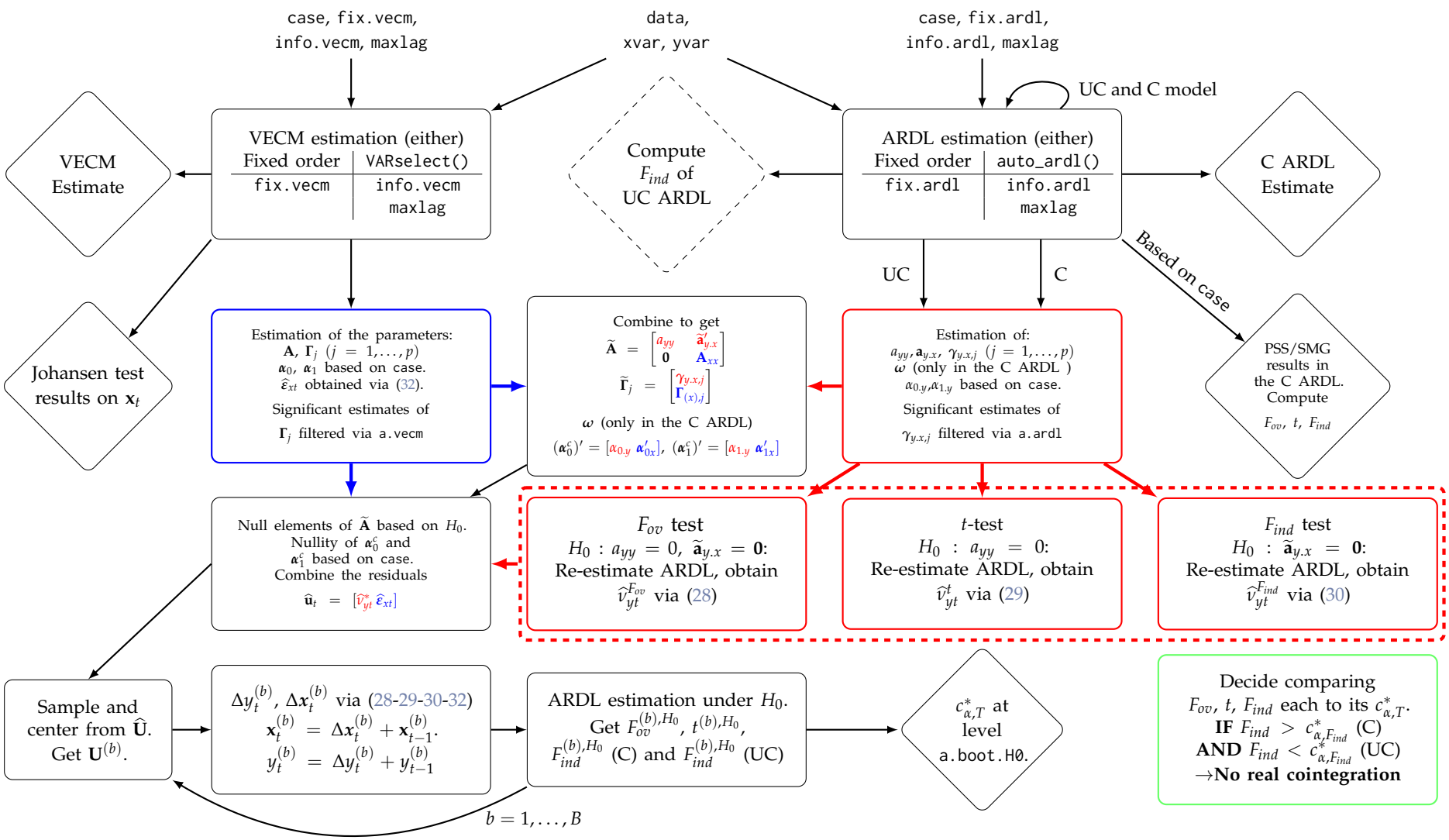


Figure 4: Flowchart of the `boot_ardl` function inner steps. Boxes denote parameter definitions and transformations. Diamonds denote function outputs. Dashed diamonds denote intermediate output (not shown after function call). Empty nodes denote function inputs. The first $p+1$ rows of $\mathbf{z}_t^{(b)}$ are set equal to the first $p+1$ rows of the original data. The best lag order for each difference variable in the ARDL model is determined via `auto_ardl()`. It is reported as a unique value p in $\gamma_{y.x,j}$ for brevity in the flowchart.

ISSN 2073-4859

4.3 Execution time and technical remarks

In order to investigate the sensitivity of the procedure to different sample sizes and number of bootstrap replicates, an experiment has been run using a three-dimensional time series of length $T = \{50, 80, 100, 200, 500\}$, generating 100 datasets for each sample size with the `sim_vecm_ardl` function (Case II, with cointegrated variables, and 2 lags in the short-run section of the model).
Then, the `boot_ardl` function has been called

```
boot_ardl(data = df_sim,
          nboot = bootr,
          case = 2,
          fix.ardl = rep(2, 3),
          fix.vecm = 2)
```

In the code above, `bootr` has been set equal to $B = \{200, 500, 1000, 2000\}$, the number of lags has been assumed known (`fix.ardl` and `fix.vecm`), while default values have been used for every other argument (such as `a.ardl`, `a.vecm` and `a.boot.H0`).
Table 1 shows the average running time per replication together with the coefficient of variation (%) of the bootstrap critical values of the F_{ov} test, for each value of T and B, across 100 replications for each scenario.
Naturally, the running time increases as both sample size and bootstrap replicates increase. However, it can be noticed how the coefficients of variation tend to stabilize for $B \geq 1000$, especially for $T > 80$, at the 5% significance level. Therefore, it is recommended a number of bootstrap replicates of at least $B = 1000$ for higher sample size, or at least $B = 2000$ for smaller samples. The analysis has been carried out using an Intel(R) Core(TM) i7-1165G7 CPU @ 2.80GHz processor, 16GB of RAM.

T	B	Exec. Time (sec)	$cv^{(F_{ov})}(5\%)$	$cv^{(F_{ov})}(2.5\%)$	$cv^{(F_{ov})}(1\%)$
50	200	23.38	8.648	10.925	13.392
50	500	48.37	6.312	6.952	8.640
50	1000	96.65	4.806	5.613	6.288
50	2000	231.15	4.255	4.226	4.946
80	200	23.46	7.251	8.936	11.263
80	500	50.19	4.998	6.220	7.946
80	1000	143.00	3.882	4.453	5.305
80	2000	255.64	2.912	3.623	4.518
100	200	37.89	7.707	8.583	10.955
100	500	52.86	4.691	5.304	7.557
100	1000	184.51	3.512	4.567	5.695
100	2000	212.65	3.519	3.674	4.185
200	200	35.46	6.644	7.173	10.365
200	500	76.78	4.734	5.355	6.225
200	1000	148.25	3.124	4.177	5.034
200	2000	484.51	2.811	3.361	3.907
500	200	54.47	6.641	8.694	10.414
500	500	133.17	5.137	5.816	6.408
500	1000	271.87	3.905	4.585	5.283
500	2000	561.71	3.221	3.490	4.145

Table 1: Average execution times (in seconds) of the `boot_ardl` function, for different combinations of sample size T and bootstrap replicates B. Coefficients of variation (cv) reported for the F_{ov} bootstrap critical values at level 5%, 2.5% and 1%.

5 Empirical applications

This section provides two illustrative application which highlight the performance of the bootstrap ARDL tests.

5.1 An application to the German macroeconomic dataset

In the first example, the occurrence of a long-run relationship between consumption [C], income [INC], and investment [INV] of Germany has been investigated via a set of ARDL models, where each variable takes in turn the role of dependent one, while the remaining are employed as independent. The models have been estimated by employing the dataset of Lütkepohl (2005) which includes quarterly data of the series over the years 1960 to 1982. The data have been employed in logarithmic form. Figure 5 displays these series over the sample period.
Before applying the bootstrap procedure, the order of integration of each series has been analyzed. Table 2 shows the results of ADF test performed on both the series and their first-differences ($k = 3$ maximum lags). The results confirm the applicability of the ARDL framework as no series is integrated of order higher than one.
The following ARDL equations have been estimated:

I First ARDL equation (C | INC, INV):

$$\begin{aligned}\Delta \log \mathrm{C}_t = {} & \alpha_{0.y} - a_{yy} \log \mathrm{C}_{t-1} - a_{y.x_1} \log \mathrm{INC}_{t-1} - a_{y.x_2} \log \mathrm{INV}_{t-1} + \\ & \sum_{j=1}^{p-1} \gamma_{y.j} \Delta \log \mathrm{C}_{t-j} + \sum_{j=1}^{s-1} \gamma_{x_1.j} \Delta \log \mathrm{INC}_{t-j} + \sum_{j=1}^{r-1} \gamma_{x_2.j} \Delta \log \mathrm{INV}_{t-j} + \\ & \omega_1 \Delta \log \mathrm{INC}_t + \omega_2 \Delta \log \mathrm{INV}_t + \nu_t.\end{aligned} \tag{38}$$

II Second ARDL equation (INC | C, INV):

$$\begin{aligned}\Delta \log \mathrm{INC}_t = {} & \alpha_{0.y} - a_{yy} \log \mathrm{INC}_{t-1} - a_{y.x_1} \log \mathrm{C}_{t-1} - a_{y.x_2} \log \mathrm{INV}_{t-1} + \\ & \sum_{j=1}^{p-1} \gamma_{y.j} \Delta \log \mathrm{INC}_{t-j} + \sum_{j=1}^{s-1} \gamma_{x_1.j} \Delta \log \mathrm{C}_{t-j} + \sum_{j=1}^{r-1} \gamma_{x_2.j} \Delta \log \mathrm{INV}_{t-j} + \\ & \omega_1 \Delta \log \mathrm{C}_t + \omega_2 \Delta \log \mathrm{INV}_t + \nu_t.\end{aligned} \tag{39}$$

III Third ARDL equation (INV | C, INC):

$$\begin{aligned}\Delta \log \mathrm{INV}_t = {} & \alpha_{0.y} - a_{yy} \log \mathrm{INV}_{t-1} - a_{y.x_1} \log \mathrm{C}_{t-1} - a_{y.x_2} \log \mathrm{INC}_{t-1} + \\ & \sum_{j=1}^{p-1} \gamma_{y.j} \Delta \log \mathrm{INV}_{t-j} + \sum_{j=1}^{s-1} \gamma_{x_1.j} \Delta \log \mathrm{C}_{t-j} + \sum_{j=1}^{r-1} \gamma_{x_2.j} \Delta \log \mathrm{INC}_{t-j} + \\ & \omega_1 \Delta \log \mathrm{C}_t + \omega_2 \Delta \log \mathrm{INC}_t + \nu_t.\end{aligned} \tag{40}$$

Table 3 shows the estimation results for each ARDL and VECM model. It is worth noting that the instantaneous difference of the independent variables are highly significant in each conditional ARDL model. Thus, neglecting these variables in the ARDL equation, as happens in the unconditional version of the model, may potentially lead to biased estimates and incorrect inference. For the sake of completeness, also the results of the marginal VECM estimation are reported for each model.
The code to prepare the data, available in the package as the `ger_macro` dataset, is:

```
data("ger_macro")
LNDATA = apply(ger_macro[,-1], 2, log)
col_ln = paste0("LN", colnames(ger_macro)[-1])
LNDATA = as.data.frame(LNDATA)
colnames(LNDATA) = col_ln
```

Then, the `boot_ardl` function is called, to perform the bootstrap tests. In the code chunk below, Model I is considered.

```
set.seed(999)
BCT_res_CONS = boot_ardl(data = LNDATA,
                         yvar = "LNCONS",
                         xvar = c("LNINCOME", "LNINVEST"),
                         maxlag = 5,
                         a.ardl = 0.1,
                         a.vecm = 0.1,
                         nboot = 2000,
                         case = 3,
                         a.boot.H0 = c(0.05),
                         print = T)
```

to which follows the call to the `summary` function

```
summary(BCT_res_CONS, out = "ARDL")
summary(BCT_res_CONS, out = "VECM")
summary(BCT_res_CONS, out = "cointVECM")
summary(BCT_res_CONS, out = "cointARDL")
```

The first summary line displays the output in the ARDL column of Table 3 and the second column of Table 4, Model I. The second line corresponds to the VECM columns of Table 3, Model I - only for the independent variables. The information on the rank of the $\mathbf{A}_{xx}$ in Table 3 is inferred from the third line. Finally, the fourth summary line corresponds to the test results in Table 4, Model I. A textual indication of the presence of spurious cointegration is displayed at the bottom of the `"cointARDL"` summary, if detected.
In this example, the bootstrap and bound testing procedures are in agreement only for model I, indicating the existence of a cointegrating relationship. Additionally, no spurious cointegration is detected for this model. As for models II and III, the null hypothesis is not rejected by the bootstrap tests, while the PSS and SMG bound tests fail to give a conclusive answer in the F_{ind} test.
The running time of the entire analysis is of roughly 11 minutes, using an Intel(R) Core(TM) i7-1165G7 CPU @ 2.80GHz processor, 16GB of RAM.

		level variable		first difference	
Series	lag	ADF	p.value	ADF	p-value
$\log \mathrm{C}_t$	0	-1.690	0.450	-9.750	< 0.01
	1	-1.860	0.385	-5.190	< 0.01
	2	-1.420	0.549	-3.130	0.030
	3	-1.010	0.691	-2.720	0.080
$\log \mathrm{INC}_t$	0	-2.290	0.217	-11.140	< 0.01
	1	-1.960	0.345	-7.510	< 0.01
	2	-1.490	0.524	-5.120	< 0.01
	3	-1.310	0.587	-3.290	0.020
$\log \mathrm{INV}_t$	0	-1.200	0.625	-8.390	< 0.01
	1	-1.370	0.565	-5.570	< 0.01
	2	-1.360	0.570	-3.300	0.020
	3	-1.220	0.619	-3.100	0.032

Table 2: ADF preliminary test (null hypothesis: random walk with drift).

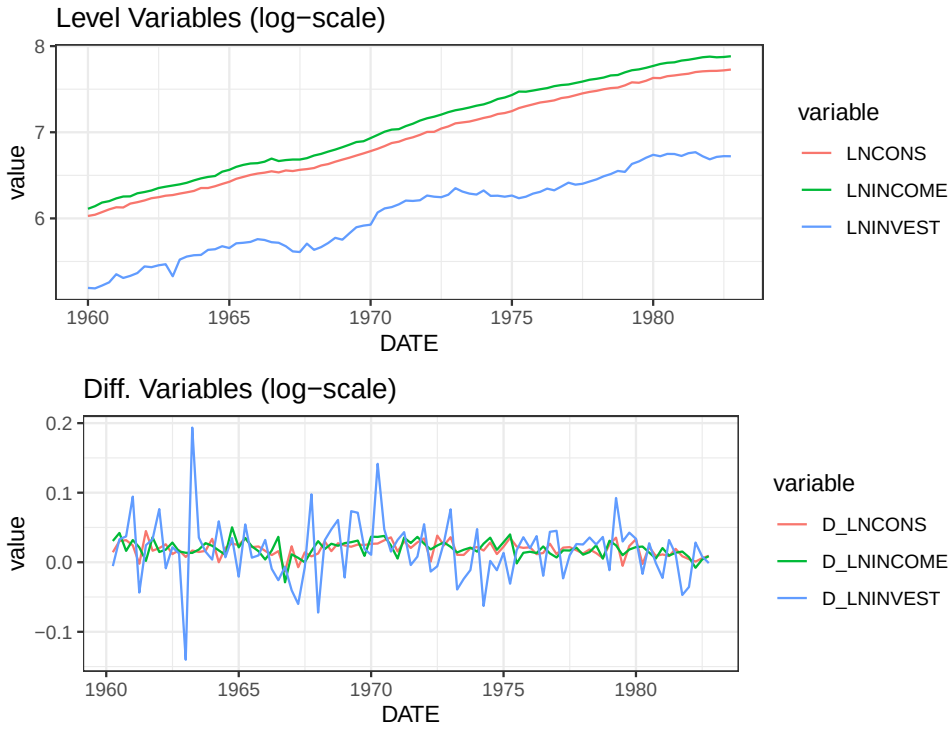


Figure 5: log-consumption/investment/income graphs (level variables and first differences). Made with ggplot.

	Model I			Model II			Model III		
	ARDL	VECM		ARDL	VECM		ARDL	VECM	
	$\Delta \log C_t$	$\Delta \log INV_t$	$\Delta \log INC_t$	$\Delta \log INC_t$	$\Delta \log C_t$	$\Delta \log INV_t$	$\Delta \log INV_t$	$\Delta \log C_t$	$\Delta \log INC_t$
$\log C_{t-1}$	-0.307 ***			0.168 *	-0.0011	0.1286*	0.611 .	-0.2727***	-0.0508
	(0.055)			(0.081)	(0.0126)	(0.0540)	(0.339)	(0.0704)	(0.0796)
$\log INC_{t-1}$	0.297 ***	0.124 *	-0.017	-0.183*			-0.491	0.2619***	0.0464
	(0.055)	(0.054)	(0.014)	(0.079)			(0.340)	(0.0681)	(0.0772)
$\log INV_{t-1}$	-0.001	-0.152 *	0.016	0.0209	-0.00107	-0.1531*	-0.1212*		
	(0.011)	(0.063)	(0.017)	(0.0135)	(0.0142)	(0.0607)	(0.060)		
$\Delta \log C_{t-1}$	-0.248 **	0.899 *	0.211 .	0.375***		0.9288*	1.113 *		0.2072 .
	(0.079)	(0.442)	(0.113)	(0.1086)		(0.442)	(0.441)		(0.1142)
$\Delta \log C_{t-2}$		0.744				0.8049 .			
		(0.431)				(0.4345)			
$\Delta \log INC_{t-1}$				-0.1404					
				(0.1095)					
$\Delta \log INC_{t-2}$					0.2675**			0.1522.	
					(0.0958)			(0.0912)	
$\Delta \log INV_{t-1}$		-0.18	0.035			-0.189 .	-0.175		0.0479 .
		(0.111)	(0.029)			(0.1097)	(0.1075)		(0.0282)
$\Delta \log INV_{t-2}$			0.049 .		0.0591*			0.0578*	0.0562*
			(0.027)		(0.0245)			(0.0223)	(0.0266)
$\Delta \log C_t$				0.7070***			1.8540***		
				(0.1093)			(0.5425)		
$\Delta \log INC_t$	0.471***						-0.445***		
	(0.074)						(0.4726)		
$\Delta \log INV_t$	0.065**			-0.0230					
	(0.019)			(0.025)					
const.	0.048 ***	0.036	0.033 *	0.002	0.0266 .	0.023	-0.056	0.0517**	0.0378*
	(0.013)	(0.066)	(0.017)	(0.018)	(0.0155)	(0.0666)	(0.072)	(0.0157)	(0.0177)
J-test		$rk(\mathbf{A_{xx}}) = 2$			$rk(\mathbf{A_{xx}}) = 2$			$rk(\mathbf{A_{xx}}) = 2$	

Table 3: Conditional ARDL and VECM results for the consumption/income/investment dataset, along with rank of the $\mathbf{A}_{xx}$ matrix via the Johansen (J) test. Significance codes: (***) 1%; (**) 5%; (.) 10%.

ISSN 2073-4859

				PSS / SMG Threshold			Outcome	
Model	Lags	Test	Boot. Critical Values	I(0) 5%	I(1) 5%	Statistic	Boot	Bound
I	(1,0,0)	F_{ov}	3.79	3.79	4.85	10.75	Y	Y
		t	-2.88	-2.86	-3.53	-5.608		
		F_{ind}	4.92	3.01	5.42	15.636		
II	(1,1,0)	F_{ov}	5.79	3.79	4.85	2.867	N	U
		t	-3.69	-2.86	-3.53	-2.315		
		F_{ind}	7.38	3.01	5.42	3.308		
III	(1,1,0)	F_{ov}	5.50	3.79	4.85	3.013	N	U
		t	-3.32	-2.86	-3.53	-2.020		
		F_{ind}	6.63	3.01	5.42	4.189		

Table 4: Cointegration analysis for the three ARDL equations in the German macroeconomic data. The optimal number of ARDL lags in the short-run - in the form (y, x_1, x_2), matching the model definition - bootstrap critical values, bound test thresholds and test statistics for each test are shown (case III). The outcome columns draw conclusions on each type of model (bootstrap or bound): Y = cointegrated, N = not cointegrated, D1 = degenerate of type 1, D2 = degenerate of type 2, U = inconclusive inference.

5.2 An application on Italian Macroeconomic Data

Following Bertelli et al. (2022), the relationship between foreign direct investment [FDI], exports [EXP], and gross domestic product [GDP] in Italy is investigated. The data of these three yearly variables have been retrieved from the World Bank Database and cover the period from 1970 to 2020. In the analysis, the log of the variables has been used and [EXP] and [FDI] have been adjusted using the GDP deflator. Figure 6 displays these series over the sample period.

Figure 6: log-GDP/export/investment graphs (level variables and first differences). Made with **ggplot**.

Table 5 shows the outcomes of the ADF test performed on each variable, which ensures that the integration order is not higher than one for all variables. Table 6 shows the results of bound and bootstrap tests performed in ARDL model by taking each variable, in turn, as the dependent one. The following ARDL equations have been estimated:

I First ARDL equation (GDP | EXP, FDI):

$$\begin{aligned}\Delta \log \mathrm{GDP}_t = {} & \alpha_{0.y} - a_{yy} \log \mathrm{GDP}_{t-1} - a_{y.x_1} \log \mathrm{EXP}_{t-1} - a_{y.x_2} \log \mathrm{FDI}_{t-1} + \\ & \sum_{j=1}^{p-1} \gamma_{y.j} \Delta \log \mathrm{GDP}_{t-j} + \sum_{j=1}^{s-1} \gamma_{x_1.j} \Delta \log \mathrm{EXP}_{t-j} + \sum_{j=1}^{r-1} \gamma_{x_2.j} \Delta \log \mathrm{FDI}_{t-j} + \\ & \omega_1 \Delta \log \mathrm{EXP}_t + \omega_2 \Delta \log \mathrm{FDI}_t + \nu_t\end{aligned} \tag{41}$$

. For this model, a degenerate case of the first type can be observed, while the simpler bound testing procedure does not signal cointegration.

II Second ARDL equation (EXP | GDP, FDI):

$$\begin{aligned}\Delta \log \mathrm{EXP}_t = {} & \alpha_{0.y} - a_{yy} \log \mathrm{EXP}_{t-1} - a_{y.x_1} \log \mathrm{GDP}_{t-1} - a_{y.x_2} \log \mathrm{FDI}_{t-1} + \\ & \sum_{j=1}^{p-1} \gamma_{y.j} \Delta \log \mathrm{EXP}_{t-j} + \sum_{j=1}^{s-1} \gamma_{x_1.j} \Delta \log \mathrm{GDP}_{t-j} + \sum_{j=1}^{r-1} \gamma_{x_2.j} \Delta \log \mathrm{FDI}_{t-j} + \\ & \omega_1 \Delta \log \mathrm{GDP}_t + \omega_2 \Delta \log \mathrm{FDI}_t + \nu_t.\end{aligned} \tag{42}$$

For this model, the ARDL bootstrap test indicates absence of cointegration, while the bound testing approach is inconclusive for the F_{ind} test.

III Third ARDL equation (FDI | GDP, EXP):

$$\begin{aligned}\Delta \log \mathrm{FDI}_t = {} & \alpha_{0.y} - a_{yy} \log \mathrm{FDI}_{t-1} - a_{y.x_1} \log \mathrm{GDP}_{t-1} - a_{y.x_2} \log \mathrm{EXP}_{t-1} + \\ & \sum_{j=1}^{p-1} \gamma_{y.j} \Delta \log \mathrm{FDI}_{t-j} + \sum_{j=1}^{s-1} \gamma_{x_1.j} \Delta \log \mathrm{GDP}_{t-j} + \sum_{j=1}^{r-1} \gamma_{x_2.j} \Delta \log \mathrm{EXP}_{t-j} + \\ & \omega_1 \Delta \log \mathrm{GDP}_t + \omega_2 \Delta \log \mathrm{EXP}_t + \nu_t.\end{aligned} \tag{43}$$

For this model, the long-run cointegrating relationship is confirmed using both boostrap and bound testing. No spurious cointegration is detected.

The code to load the data and perform the analysis (e.g. for Model I) is:

```
data("ita_macro")
BCT_res_GDP = boot_ardl(data = ita_macro,
                        yvar = "LGDP",
                        xvar = c("LEXP", "LFI"),
                        maxlag = 5,
                        a.ardl = 0.1,
                        a.vecm = 0.1,
                        nboot = 2000,
                        case = 3,
                        a.boot.H0 = c(0.05),
                        print = T)
```

For the sake of simplicity, the conditional ARDL and VECM marginal models outputs included in each cointegrating analysis is omitted. The summary for the cointegration tests for Model I is called via

```
summary(BCT_res_GDP, out = "ARDL") # extract lags
summary(BCT_res_GDP, out ="cointARDL") # ARDL cointegration
```

This empirical application further highlights the importance of dealing with inconclusive inference via the bootstrap procedure, while naturally including the effect of conditioning in the ARDL model, as highlighted in Bertelli et al. (2022).

6 Conclusion

The **bootCT** package allows the user to perform bootstrap cointegration tests in ARDL models by overcoming the problem of inconclusive inference which is a well-known drawback of standard bound tests. The package makes use of different functions. The function `boot_ardl` performs the bootstrap

	No Drift, No Trend				Drift, No Trend				Drift and Trend			
Variable	Lag = 0	Lag = 1	Lag = 2	Lag = 3	Lag = 0	Lag = 1	Lag = 2	Lag = 3	Lag = 0	Lag = 1	Lag = 2	Lag = 3
$\log \text{GDP}_t$	0.99	0.974	0.941	0.796	< 0.01	< 0.01	< 0.01	0.084	0.99	0.99	0.99	0.99
$\log \text{FDI}_t$	0.572	0.599	0.675	0.725	< 0.01	0.0759	0.3199	0.5174	< 0.01	0.013	0.151	0.46
$\log \text{EXP}_t$	0.787	0.71	0.698	0.684	0.479	0.288	0.467	0.433	0.629	0.35	0.463	0.379
$\Delta \log \text{GDP}_t$	< 0.01	< 0.0164	0.0429	0.0402	< 0.01	0.0861	0.3989	0.4267	< 0.01	< 0.01	0.0166	0.017
$\Delta \log \text{FDI}_t$	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01
$\Delta \log \text{EXP}_t$	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	< 0.01	0.0336	0.0315

Table 5: ADF preliminary test for the second example.

				PSS / SMG Threshold			Outcome	
Model	Lags	Test	Boot. Critical Values	I(0) 5%	I(1) 5%	Statistic	Boot	Bound
I	(1,1,0)	F_{ov}	3.730	4.070	5.190	9.758	D1	N
		t	-2.020	-2.860	-3.530	-2.338		
		F_{ind}	3.710	3.220	5.620	2.273		
II	(1,0,0)	F_{ov}	5.400	4.070	5.190	2.649	N	U
		t	-3.380	-2.860	-3.530	-1.889		
		F_{ind}	5.630	3.220	5.620	3.481		
III	(1,0,0)	F_{ov}	5.360	4.070	5.190	6.716	Y	Y
		t	-3.550	-2.860	-3.530	-4.202		
		F_{ind}	6.500	3.220	5.620	7.017		

Table 6: Cointegration analysis for the three ARDL equations in the Italian macroeconomic data. The optimal number of ARDL lags in the short-run - in the form (y, x_1, x_2), matching the model definition - bootstrap critical values, bound test thresholds and test statistics for each test are shown (case III). The outcome columns draw conclusions on each type of model (bootstrap or bound): Y = cointegrated, N = not cointegrated, D1 = degenerate of type 1, D2 = degenerate of type 2, U = inconclusive inference.

tests, and it acts as a wrapper of both the bootstrap and the standard bound tests, including also the Johansen test on the independent variables of the model. Finally, it also performs the bound F-test on the lagged independent variables, so far not available in other extant `R` packages. The function `sim_vecm_ardl`, which allows the simulation of multivariate time series data following a user-defined DGP, enriches the available procedures for multivariate data generation, while the function `lag_mts` provides a supporting tool in building datasets of lagged variables for any practical purpose. Finally, the use of Rcpp functions gives a technical advantage in terms of computational speed, performing the bootstrap analysis within an acceptable time frame.

7 Appendix

7.1 Section A - the methodological framework of (conditional) VECM and ARDL models

Expanding the matrix polynomial $\mathbf{A}(z)$ about $z = 1$, yields

$$\mathbf{A}(z) = \mathbf{A}(1)z + (1-z)\mathbf{\Gamma}(z), \tag{44}$$

where

$$\mathbf{A}(1) = \mathbf{I}_{K+1} - \sum_{j=1}^{p} \mathbf{A}_j \tag{45}$$

$$\mathbf{\Gamma}(z) = \mathbf{I}_{K+1} - \sum_{i=1}^{p-1} \mathbf{\Gamma}_i z^i, \quad \mathbf{\Gamma}_i = -\sum_{j=i+1}^{p} \mathbf{A}_j. \tag{46}$$

The VECM model (2) follows accordingly, and

$$\boldsymbol{\alpha}_0 = \mathbf{A}(1)\boldsymbol{\mu} + (\mathbf{\Gamma}(1) - \mathbf{A}(1))\boldsymbol{\eta}, \qquad \boldsymbol{\alpha}_1 = \mathbf{A}(1)\boldsymbol{\eta}. \tag{47}$$

Assuming that $\mathbf{A}(1)$ is singular and that the variables $\mathbf{x}_t$ are cointegrated. This entails the following

$$\mathbf{A}(1) = \begin{bmatrix} a_{yy} & \mathbf{a}'_{yx} \\ (1,1) & (1,K) \\ \mathbf{a}_{xy} & \mathbf{A}_{xx} \\ (K,1) & (K,K) \end{bmatrix} = \underset{(K+1,r+1)}{\mathbf{B}} \underset{(r+1,K+1)}{\mathbf{C}'} = \begin{bmatrix} b_{yy} & \mathbf{b}'_{yx} \\ \mathbf{b}_{xy} & \mathbf{B}_{xx} \end{bmatrix} \begin{bmatrix} c_{yy} & \mathbf{c}'_{yx} \\ \mathbf{c}_{xy} & \mathbf{C}'_{xx} \end{bmatrix} =$$

$$= \begin{bmatrix} b_{yy}c_{yy} + \mathbf{b}'_{yx}\mathbf{c}_{xy} & b_{yy}\mathbf{c}'_{yx} + \mathbf{b}'_{yx}\mathbf{C}'_{xx} \\ \mathbf{b}_{xy}c_{yy} + \mathbf{B}_{xx}\mathbf{c}_{xy} & \mathbf{b}_{xy}\mathbf{c}'_{yx} + \mathbf{A}_{xx} \end{bmatrix}, \qquad rk(\mathbf{A}(1)) = rk(\mathbf{B}) = rk(\mathbf{C}), \tag{48}$$

where $\mathbf{B}$ and $\mathbf{C}$ are full column rank matrices arising from the rank-factorization of $\mathbf{A}(1) = \mathbf{B}\mathbf{C}'$ with $\mathbf{C}$ matrix of the long-run relationships of the process and $\mathbf{B}_{xx}$, $\mathbf{C}_{xx}$ arising from the rank factorization of $\mathbf{A}_{xx} = \mathbf{B}_{xx}\mathbf{C}'_{xx}$, with $rk(\mathbf{A}_{xx}) = rk(\mathbf{B}_{xx}) = rk(\mathbf{C}_{xx}) = r$ [6].
By partitioning the vectors $\boldsymbol{\alpha}_0$, $\boldsymbol{\alpha}_1$, the matrix $\mathbf{A}(1)$ and the polynomial matrix $\mathbf{\Gamma}(L)$ conformably to $\mathbf{z}_t$, as follows

$$\boldsymbol{\alpha}_0 = \begin{bmatrix} \alpha_{0y} \\ (1,1) \\ \boldsymbol{\alpha}_{0x} \\ (K,1) \end{bmatrix}, \qquad \boldsymbol{\alpha}_1 = \begin{bmatrix} \alpha_{1y} \\ (1,1) \\ \boldsymbol{\alpha}_{1x} \\ (K,1) \end{bmatrix} \tag{49}$$

$$\mathbf{A}(1) = \begin{bmatrix} \mathbf{a}'_{(y)} \\ (1,K+1) \\ \mathbf{A}_{(x)} \\ (K,K+1) \end{bmatrix} = \begin{bmatrix} a_{yy} & \mathbf{a}'_{yx} \\ (1,1) & (1,K) \\ \mathbf{a}_{xy} & \mathbf{A}_{xx} \\ (K,1) & (K,K) \end{bmatrix}, \qquad \mathbf{\Gamma}(L) = \begin{bmatrix} \boldsymbol{\gamma}'_y(L) \\ (1,K+1) \\ \mathbf{\Gamma}_{(x)}(L) \\ (K,K+1) \end{bmatrix} = \begin{bmatrix} \gamma_{yy}(L) & \boldsymbol{\gamma}'_{yx}(L) \\ (1,1) & (1,K) \\ \boldsymbol{\gamma}_{xy}(L) & \mathbf{\Gamma}_{xx}(L) \\ (K,1) & (K,K) \end{bmatrix} \tag{50}$$

, and substituting (6) into (2) yields

$$\Delta\mathbf{z}_t = \begin{bmatrix} \Delta y_t \\ \Delta\mathbf{x}_t \end{bmatrix} = \begin{bmatrix} \alpha_{0.y} \\ \boldsymbol{\alpha}_{0x} \end{bmatrix} + \begin{bmatrix} \alpha_{1.y} \\ \boldsymbol{\alpha}_{1x} \end{bmatrix} t - \begin{bmatrix} \mathbf{a}'_{(y).x} \\ \mathbf{A}_{(x)} \end{bmatrix} \begin{bmatrix} y_{t-1} \\ \mathbf{x}_{t-1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\gamma}'_{y.x}(L) \\ \mathbf{\Gamma}_{(x)}(L) \end{bmatrix} \Delta\mathbf{z}_t + \begin{bmatrix} \boldsymbol{\omega}'\Delta\mathbf{x}_t \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \nu_{yt} \\ \boldsymbol{\varepsilon}_{xt} \end{bmatrix} \tag{51}$$

, where

$$\alpha_{0.y} = \alpha_{0y} - \boldsymbol{\omega}'\boldsymbol{\alpha}_{0x}, \qquad \alpha_{1.y} = \alpha_{1y} - \boldsymbol{\omega}'\boldsymbol{\alpha}_{1x} \tag{52}$$

$$\mathbf{a}'_{(y).x} = \mathbf{a}'_{(y)} - \boldsymbol{\omega}'\mathbf{A}_{(x)}, \qquad \boldsymbol{\gamma}'_{y.x}(L) = \boldsymbol{\gamma}'_y(L) - \boldsymbol{\omega}'\mathbf{\Gamma}_{(x)}(L). \tag{53}$$

According to (51), the long-run relationships of the VECM turn out to be now included in the matrix

$$\begin{bmatrix} \mathbf{a}'_{(y).x} \\ \mathbf{A}_{(x)} \end{bmatrix} = \begin{bmatrix} a_{yy} - \boldsymbol{\omega}'\mathbf{a}_{xy} & \mathbf{a}'_{yx} - \boldsymbol{\omega}'\mathbf{A}_{xx} \\ \mathbf{a}_{xy} & \mathbf{A}_{xx} \end{bmatrix}. \tag{54}$$

To rule out the presence of long-run relationships between y_t and $\mathbf{x}_t$ in the marginal model, the $\mathbf{x}_t$ variables are assumed to be exogenous with respect to the ARDL parameters, that is $\mathbf{a}_{xy}$ is assumed to be a null vector. Accordingly, the long-run matrix in (54) becomes

$$\widetilde{\mathbf{A}} = \begin{bmatrix} a_{yy} & \mathbf{a}'_{yx} - \boldsymbol{\omega}'\mathbf{A}_{xx} \\ \mathbf{0} & \mathbf{A}_{xx} \end{bmatrix} = \begin{bmatrix} a_{yy} & \widetilde{\mathbf{a}}'_{y.x} \\ \mathbf{0} & \mathbf{A}_{xx} \end{bmatrix} = \begin{bmatrix} b_{yy}c_{yy} & b_{yy}\mathbf{c}'_{yx} + (\mathbf{b}'_{yx} - \boldsymbol{\omega}'\mathbf{B}_{xx})\mathbf{C}'_{xx} \\ \mathbf{0} & \mathbf{B}_{xx}\mathbf{C}'_{xx} \end{bmatrix}. \tag{55}$$

[6] If the explanatory variables are stationary $\mathbf{A}_{xx}$ is non-singular ($rk(\mathbf{A}_{xx}) = K$), while when they are integrated but without cointegrating relationship $\mathbf{A}_{xx}$ is a null matrix

After these algebraic transformations, the ARDL equation for Δy_t can be rewritten as in (7).
In light of the factorization (48) of the matrix $\mathbf{A}(1)$, the long-run equilibrium vector $\boldsymbol{\theta}$ can be expressed as

$$\boldsymbol{\theta}' = -\frac{1}{a_{yy}}\underset{(1,r+1)}{\left[b_{yy} \quad (\mathbf{b}_{yx} - \boldsymbol{\omega}'\mathbf{B}_{xx})\right]}\underset{(r+1,K)}{\begin{bmatrix}\mathbf{c}'_{yx}\\ \mathbf{C}'_{xx}\end{bmatrix}}, \tag{56}$$

where $\widetilde{\mathbf{a}}_{y.x} = \mathbf{a}_{yx} - \boldsymbol{\omega}'\mathbf{A}_{xx}$.
Bearing in mind that $\mathbf{C}'_{xx}$ is the cointegrating matrix for the variables $\mathbf{x}_t$, the equation (56) leads to the following conclusion

$$rk\begin{bmatrix}\mathbf{c}'_{yx}\\ \mathbf{C}'_{xx}\end{bmatrix} = \begin{cases} r \rightarrow & y_t \sim I(0) \\ r+1 \rightarrow & y_t \sim I(1) \end{cases}, \tag{57}$$

where $r = rk(\mathbf{A}_{xx})$ and $0 \leq r \leq K$.

7.2 Section B - Intercept and trend specifications

Pesaran et al. (2001) introduced five different specifications for the ARDL model, which depend on the deterministic components that can be absent or restricted to the values they assume in the parent VAR model. In this connection, note that, in light of (47), the drift and the trend coefficient in the conditional VECM (51) are defined as

$$\boldsymbol{\alpha}_{\mathbf{0}}^{c} = \widetilde{\mathbf{A}}(1)(\boldsymbol{\mu} - \boldsymbol{\eta}) + \widetilde{\boldsymbol{\Gamma}}(1)\boldsymbol{\eta}, \quad \boldsymbol{\alpha}_{\mathbf{1}}^{c} = \widetilde{\mathbf{A}}(1)\boldsymbol{\eta}, \tag{58}$$

where $\widetilde{\mathbf{A}}(1)$ is as in (55) and $\widetilde{\boldsymbol{\Gamma}}(1) = \begin{bmatrix}\boldsymbol{\gamma}'_{y.x}(1)\\ \boldsymbol{\Gamma}_{(x)}(1)\end{bmatrix}$.
Accordingly, after partitioning the mean and the drift vectors as

$$\underset{(1,K+1)}{\boldsymbol{\mu}'} = \underset{(1,1)}{[\,\mu_y\,,}\ \underset{(1,K)}{\boldsymbol{\mu}'_x\,]}, \quad \underset{(1,K+1)}{\boldsymbol{\eta}'} = \underset{(1,1)}{[\,\eta_y\,,}\ \underset{(1,K)}{\boldsymbol{\eta}'_x\,]}, \tag{59}$$

the intercept and the coefficient of the trend of the ARDL equation (7) are defined as

$$\alpha_{0.y}^{EC} = \mathbf{e}'_1\boldsymbol{\alpha}_{\mathbf{0}}^{c} = a_{yy}\mu_y - \widetilde{\mathbf{a}}'_{y.x}\boldsymbol{\mu}_x + \boldsymbol{\gamma}'_{y.x}(1)\boldsymbol{\eta} = a_{yy}(\mu_y - \boldsymbol{\theta}'\boldsymbol{\mu}_x) + \boldsymbol{\gamma}'_{y.x}(1)\boldsymbol{\eta}, \ \ \boldsymbol{\theta}' = -\frac{\widetilde{\mathbf{a}}'_{y.x}}{a_{yy}} \tag{60}$$

$$\alpha_{1.y}^{EC} = \mathbf{e}'_1\boldsymbol{\alpha}_{\mathbf{1}}^{c} = a_{yy}\eta_y - \widetilde{\mathbf{a}}'_{y.x}\boldsymbol{\eta}_x = a_{yy}(\eta_y - \boldsymbol{\theta}'\boldsymbol{\eta}_x), \tag{61}$$

where $\mathbf{e}_1$ is the $K+1$ first elementary vector.
In the error correction term

$$EC_{t-1} = y_{t-1} - \theta_0 - \theta_1 t - \boldsymbol{\theta}'\mathbf{x}_{t-1} \tag{62}$$

the parameters that partake in the calculation of intercept and trend are

$$\theta_0 = \mu_y - \boldsymbol{\theta}'\boldsymbol{\mu}_x, \ \ \theta_1 = \eta_y - \boldsymbol{\theta}'\boldsymbol{\eta}_x. \tag{63}$$

In particular, these latter are not null only when they are assumed to be restricted in the model specification.
The five specifications proposed by Pesaran et al. (2001) are

I *No intercept and no trend*:

$$\boldsymbol{\mu} = \boldsymbol{\eta} = \mathbf{0}. \tag{64}$$

It follows that

$$\theta_0 = \theta_1 = \alpha_{0.y} = \alpha_{1.y} = 0. \tag{65}$$

Accordingly, the model is as in (17).

II *Restricted intercept and no trend*:

$$\boldsymbol{\alpha}_{\mathbf{0}}^{c} = \widetilde{\mathbf{A}}(1)\boldsymbol{\mu}, \quad \boldsymbol{\eta} = \mathbf{0}, \tag{66}$$

which entails

$$\theta_0 \neq 0 \quad \alpha_{0.y}^{EC} = a_{yy}\theta_0, \quad \alpha_{0.y} = \theta_1 = \alpha_{1.y} = 0. \tag{67}$$

Therefore, the intercept stems from the EC term of the ARDL equation. The model is specified as in (18)

III *Unrestricted intercept and no trend*:

$$\boldsymbol{\alpha}_{\mathbf{0}}^{c} \neq \widetilde{\mathbf{A}}(1)\boldsymbol{\mu}, \quad \boldsymbol{\eta} = \mathbf{0}. \tag{68}$$

Thus,

$$\alpha_{0.y} \neq 0, \quad \theta_0 = \theta_1 = \alpha_{1.y} = 0. \tag{69}$$

Accordingly, the model is as in (19).

IV *Unrestricted intercept, restricted trend*:

$$\boldsymbol{\alpha}_{\mathbf{0}}^{c} \neq \widetilde{\mathbf{A}}(1)(\boldsymbol{\mu} - \boldsymbol{\eta}) + \widetilde{\boldsymbol{\Gamma}}(1)\boldsymbol{\eta} \quad \boldsymbol{\alpha}_{1}^{c} = \widetilde{\mathbf{A}}(1)\boldsymbol{\eta}, \tag{70}$$

which entails

$$\alpha_{0.y} \neq 0, \quad \theta_0 = 0 \quad \theta_1 \neq 0 \quad \alpha_{1.y}^{EC} = a_{yy}\theta_1 \quad \alpha_{1.y} = 0. \tag{71}$$

Accordingly, the trend stems from the EC term of the ARDL equation. The model is as in (20).

V *Unrestricted intercept, unrestricted trend*:

$$\boldsymbol{\alpha}_{\mathbf{0}}^{c} \neq \widetilde{\mathbf{A}}(1)(\boldsymbol{\mu} - \boldsymbol{\eta}) + \widetilde{\boldsymbol{\Gamma}}(1)\boldsymbol{\eta} \quad \boldsymbol{\alpha}_{1}^{c} \neq \widetilde{\mathbf{A}}(1)\boldsymbol{\eta}. \tag{72}$$

Accordingly,

$$\alpha_{0.y} \neq 0 \quad \alpha_{1.y} \neq 0, \quad \theta_0 = \theta_1 = 0. \tag{73}$$

The model is as in (21).

References

K. R. Abbasi, M. Shahbaz, Z. Jiao, and M. Tufail. How energy consumption, industrial growth, urbanization, and co2 emissions affect economic growth in pakistan? a novel dynamic ardl simulations approach. *Energy*, 221:119793, 2021. doi: 10.1016/j.energy.2021.119793. [p39]

M. A. Arranz and A. Escribano. Cointegration testing under structural breaks: A robust extended error correction model. *Oxford Bulletin of Economics and Statistics*, 62(1):23–52, 2000. doi: 10.1111/1468-0084.00158. [p39]

S. M. Bache and H. Wickham. *magrittr: A Forward-Pipe Operator for R*, 2022. URL https://CRAN.R-project.org/package=magrittr. R package version 2.0.3. [p40]

S. Bertelli, G. Vacca, and M. Zoia. Bootstrap cointegration tests in ardl models. *Economic Modelling*, 116:105987, 2022. doi: 10.1016/j.econmod.2022.105987. [p40, 42, 51, 58, 59]

B. Bolker, G. R. Warnes, and T. Lumley. *gtools: Various R Programming Tools*, 2022. URL https://CRAN.R-project.org/package=gtools. R package version 3.9.4. [p40]

H. W. Borchers. *pracma: Practical Numerical Math Functions*, 2022. URL https://CRAN.R-project.org/package=pracma. R package version 2.4.2. [p40]

S. Cook. The power of single equation tests for cointegration. *Applied Economics Letters*, 13(5):265–267, 2006. doi: 10.1080/13504850500398534. [p39]

R. Davidson and J. G. MacKinnon. The case against jive. *Journal of Applied Econometrics*, 21(6):827–833, 2005. doi: 10.1002/jae.873. [p44]

D. Eddelbuettel. *Seamless R and C++ Integration with Rcpp*. Springer, New York, 2013. doi: 10.1007/978-1-4614-6868-4. ISBN 978-1-4614-6867-7. [p40, 50]

D. Eddelbuettel, R. Francois, D. Bates, B. Ni, and C. Sanderson. *RcppArmadillo: 'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library*, 2023. URL https://CRAN.R-project.org/package=RcppArmadillo. R package version 0.12.4.0.0. [p40]

R. F. Engle and C. W. Granger. Co-integration and error correction: representation, estimation, and testing. *Econometrica: journal of the Econometric Society*, pages 251–276, 1987. doi: 10.2307/1913236. [p39]

R. F. Engle and B. S. Yoo. Forecasting and testing in co-integrated systems. *Journal of Econometrics*, 35 (1):143–159, 1987. doi: 10.1016/0304-4076(87)90085-6. [p39]

N. R. Ericsson and J. G. MacKinnon. Distributions of error correction tests for cointegration. *The Econometrics Journal*, 5(2):285–318, 2002. doi: 10.1111/1368-423X.00085. [p39]

V. J. Gabriel, Z. Psaradakis, and M. Sola. A simple method of testing for cointegration subject to multiple regime changes. *Economics Letters*, 76(2):213–221, 2002. [p39]

M. Haseeb, I. S. Z. Abidin, Q. M. A. Hye, and N. H. Hartani. The impact of renewable energy on economic well-being of malaysia: Fresh evidence from auto regressive distributed lag bound testing approach. *International Journal of Energy Economics and Policy*, 9(1):269, 2019. doi: 10.32479/ijeep.7229. [p39]

R. M. Hope. *Rmisc: Ryan Miscellaneous*, 2022. URL https://CRAN.R-project.org/package=Rmisc. R package version 1.5.1. [p40]

H. I. Hussain, M. A. Salem, A. Z. A. Rashid, and F. Kamarudin. Environmental impact of sectoral energy consumption on economic growth in malaysia: Evidence from ardl bound testing approach. *Ekoloji Dergisi*, (107), 2019. [p39]

S. Johansen. Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models. *Econometrica: journal of the Econometric Society*, pages 1551–1580, 1991. doi: 10.2307/2938278. [p39]

S. Jordan and A. Q. Philips. *dynamac: Dynamic Simulation and Testing for Single-Equation ARDL Models*, 2020. URL https://CRAN.R-project.org/package=dynamac. R package version 0.1.11. [p40]

A. Kanioura and P. Turner. Critical values for an f-test for cointegration in a multivariate model. *Applied Economics*, 37(3):265–270, 2005. doi: 10.1080/00036840412331315051. [p40]

J. J. Kremers, N. R. Ericsson, and J. J. Dolado. The power of cointegration tests. *Oxford bulletin of economics and statistics*, 54(3):325–348, 1992. doi: 10.1111/j.1468-0084.1992.tb00005.x. [p39]

S. Kripfganz and D. C. Schneider. Response surface regressions for critical value bounds and approximate p-values in equilibrium correction models 1. *Oxford Bulletin of Economics and Statistics*, 82(6): 1456–1481, 2020. doi: 10.1111/obes.12377. [p40]

Lesnoff, M., Lancelot, and R. *aod: Analysis of Overdispersed Data*, 2012. URL https://cran.r-project.org/package=aod. R package version 1.3.2. [p40]

H. Lütkepohl. *New introduction to multiple time series analysis*. Springer Science & Business Media, 2005. doi: 10.1007/978-3-540-27752-1. [p54]

J. G. Mackinnon. Critical values for cointegration tests. In *Eds.), Long-Run Economic Relationship: Readings in Cointegration*. Oxford Press, 1991. [p39]

G. S. Maddala and I.-M. Kim. Unit roots, cointegration, and structural change. 1998. doi: 10.1017/CBO9780511751974. [p39]

R. McNown, C. Y. Sam, and S. K. Goh. Bootstrapping the autoregressive distributed lag test for cointegration. *Applied Economics*, 50(13):1509–1521, 2018. doi: 10.1080/00036846.2017.1366643. [p39, 40, 43]

A. N. Menegaki. The ardl method in the energy-growth nexus field; best implementation strategies. *Economies*, 7(4):105, 2019. doi: 10.3390/economies7040105. [p39]

T. C. Mills and E. J. Pentecost. The real exchange rate and the output response in four eu accession countries. *Emerging Markets Review*, 2(4):418–430, 2001. doi: 10.1016/S1566-0141(01)00027-9. [p40]

P. K. Narayan. The saving and investment nexus for china: evidence from cointegration tests. *Applied economics*, 37(17):1979–1990, 2005. doi: 10.1080/00036840500278103. [p40]

P. K. Narayan and R. Smyth. Crime rates, male youth unemployment and real income in australia: evidence from granger causality tests. *Applied Economics*, 36(18):2079–2095, 2004. doi: 10.1080/0003684042000261842. [p40]

K. Natsiopoulos and N. Tzeremes. *ARDL: ARDL, ECM and Bounds-Test for Cointegration*, 2021. URL https://CRAN.R-project.org/package=ARDL. R package version 0.1.1. [p40, 50]

M. H. Pesaran, Y. Shin, and R. J. Smith. Bounds testing approaches to the analysis of level relationships. *Journal of applied econometrics*, 16(3):289–326, 2001. doi: 10.1002/jae.616. [p39, 40, 41, 42, 45, 50, 62]

B. Pfaff. Var, svar and svec models: Implementation within R package vars. *Journal of Statistical Software*, 27(4), 2008a. URL https://www.jstatsoft.org/v27/i04/. [p40, 50]

B. Pfaff. *Analysis of Integrated and Cointegrated Time Series with R*. Springer, New York, second edition, 2008b. URL https://www.pfaffikus.de. ISBN 0-387-27960-1. [p40]

D. Qiu. *aTSA: Alternative Time Series Analysis*, 2015. URL https://CRAN.R-project.org/package=aTSA. R package version 3.1.2. [p40]

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2022. URL https://www.R-project.org/. [p]

A. M. Reda and E. Nourhan. Using the ardl bound testing approach to study the inflation rate in egypt. *Economic consultant*, (3 (31)):24–41, 2020. doi: 10.46224/ecoc.2020.3.2. [p39]

C. Y. Sam, R. McNown, and S. K. Goh. An augmented autoregressive distributed lag bounds test for cointegration. *Economic Modelling*, 80:130–141, 2019. doi: 10.1016/j.econmod.2018.11.001. [p40]

A. Trapletti and K. Hornik. *tseries: Time Series Analysis and Computational Finance*, 2023. URL https://CRAN.R-project.org/package=tseries. R package version 0.10-54. [p40]

G. Vacca and S. Bertelli. *bootCT: Bootstrapping the ARDL Tests for Cointegration*, 2023. R package version 2.0.0. [p40]

H. Wickham. Reshaping data with the reshape package. *Journal of Statistical Software*, 21(12):1–20, 2007. URL http://www.jstatsoft.org/v21/i12/. [p40]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p40, 47]

H. Wickham. *stringr: Simple, Consistent Wrappers for Common String Operations*, 2022. URL https://CRAN.R-project.org/package=stringr. R package version 1.5.0. [p40]

H. Wickham, M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. doi: 10.21105/joss.01686. [p40]

H. Wickham, R. François, L. Henry, K. Müller, and D. Vaughan. *dplyr: A Grammar of Data Manipulation*, 2023. URL https://CRAN.R-project.org/package=dplyr. R package version 1.1.2. [p40]

V. Yilanci, S. Bozoklu, and M. S. Gorus. Are brics countries pollution havens? evidence from a bootstrap ardl bounds testing approach with a fourier function. *Sustainable Cities and Society*, 55: 102035, 2020. doi: 10.1016/j.scs.2020.102035. [p39]

Gianmarco Vacca
Department of Economic Policy. Università Cattolica del Sacro Cuore
Largo Gemelli, 1, Milan.
Italy
(0000-0002-8996-5524)
gianmarco.vacca@unicatt.it

Maria Zoia
Department of Economic Policy. Università Cattolica del Sacro Cuore
Largo Gemelli, 1, Milan.
Italy
(0000-0002-8169-781X)
maria.zoia@unicatt.it

Stefano Bertelli
CRO Area, Internal Validation and Controls Department, Operational Risk and ICAAP Internal Systems, Intesa Sanpaolo, Milan
Viale Stelvio, 55/57, Milan.
Italy
stefano.bertelli@intesasanpaolo.com

Prediction, Bootstrapping and Monte Carlo Analyses Based on Linear Mixed Models with QAPE 2.0 Package

by Alicja Wolny–Dominiak and Tomasz Żądło

Abstract The paper presents a new R package **qape** for prediction, accuracy estimation of various predictors and Monte Carlo simulation studies of properties of both predictors and estimators of accuracy measures. It allows to predict any population and subpopulation characteristics of the response variable based on the Linear Mixed Model (LMM). The response variable can be transformed, e.g. to logarithm and the data can be in the cross-sectional or longitudinal framework. Three bootstrap algorithms are developed: parametric, residual and double, allowing to estimate the prediction accuracy. Analyses can also include Monte Carlo simulation studies of properties of the methods used. Unlike other packages, in the prediction process the user can flexibly define the predictor, the model, the transformation function of the response variable, the predicted characteristics and the method of accuracy estimation.

1 Introduction

One of the tasks in application of mixed models in the real-life problems is the prediction of random effects. Then, the predicted values give the possibility for further prediction, e.g. characteristics of interest such as sum, mean or quantiles or the future value of the response variable for cross-sectional or longitudinal data.

Three main predictors of these characteristics are proposed in the literature: Empirical Best Linear Unbiased Predictors - EBLUPs (see e.g. Henderson (1950) and Royall (1976)), PLUG-IN predictors (see e.g. Boubeta et al. (2016), Chwila and Żądło (2019), Hobza and Morales (2016)) and Empirical Best Predictors - EBPs (see e.g. Molina and Rao (2010)). Each assumes the LMM to model the response variable.

The numerous successful applications of these three predictors for cross-sectional and longitudinal data can be found in the model approach in survey sampling, including the small area estimation. In paper Fay III and Herriot (1979) the Authors introduce the prediction of the mean income for small places based on the special case of the LMM model called Fay-Herriot model and the EBLUP. The analysis of poverty is extended in many works, e.g. in Molina and Rao (2010) and Christiaensen et al. (2012). In turn, in Battese et al. (1988) the Authors analyse the total crop areas based on survey and satellite data using EBLUPs. The proposed LMM model is known as the Battese-Harter-Fuller model. The predictors are also exploited in the subject of experience rating in non-life insurance, see Frees et al. (1999) and Bühlmann and Gisler (2005), where the longitudinal data are under consideration. The insurance premium for the next period for every policy in the insurance portfolio is predicted.

A major challenge in this type of prediction is the estimation of the prediction accuracy measure. Most often it is the Root Mean Squared Error (RMSE), which is given in analytical form or can be e.g. estimated using bootstrap. A feature of the distribution of the squared prediction error is usually a very strong positive asymmetry. Because the mean is not recommended as the appropriate measure of the central tendency in such distributions, the alternative prediction accuracy measure called the Quantile of Absolute Prediction Errors (QAPE), proposed by Żądło (2013) and Wolny-Dominiak and Żądło (2020), can be applied.

There is a variety of R packages to calculate the considered predictors together with the accuracy measure of prediction, usually the RMSE. The package **sae**, see Molina and Marhuenda (2015), provides EBLUPs based on Fay-Herriot and Battese-Harter-Fuller models. In turn, the multivariate EBLUP for Fay-Herriot models is implemented in **msae**, see Permatasari and Ubaidillah (2021). Several EBLUPs introduced in Rao and Yu (1994) are implemented in package **saery** introduced by Lefler et al. (2014), likewise in **JoSAE**, see Breidenbach (2018), but with additional heteroscedasticity analysis. The EBP is provided in the package **emdi** described in Kreutzmann et al. (2019).

A new package in this area is our proposed package **qape**. It allows the prediction of flexibly defined characteristics of the response variable using the above three predictors, assuming an appropriate LMM. A novel feature of the package **qape**, compared to those already in place, is the ability of bootstrap estimation of the prediction accuracy measures, both the RMSE and QAPE. Three types of bootstrap procedures are provided: parametric, residual and double.

There are three groups of functions in this package: predictors values calculation, bootstrap

estimation of RMSE and QAPE measures, and Monte Carlo (MC) analysis of properties of predictors and prediction accuracy estimators. The prediction is based on a LMM model defined by the user and allows to predict the population characteristics of the response variable, which can be defined by a linear combination (in the case of EBLUP), by any R function (e.g. `sum`) or any function defined by the user (in the case of the EBP and PLUG-IN predictors). The package allows for full flexibility in defining: the model, the predicted characteristic, and the transformation of the response variable.

This paper is organized as follows. Firstly, the background of the LMM is presented together with the theoretical foundations of the prediction including prediction accuracy measures. Then, the package functionality in the area of prediction is presented and illustrated. A short application based on `radon` data, a cross-sectional dataset available in **HLMdiag** package, to predict three subpopulation characteristics is shown. Subsequently, the theoretical background of the prediction accuracy measures estimation based on bootstrap is presented. Implementations of bootstrap algorithms in **qape** are briefly introduced. Finally, the procedure of the model-based Monte Carlo simulation study is discussed. The paper ends with a conclusion.

2 Prediction accuracy measures

We consider the problem of prediction of any given function of the population vector $\mathbf{Y}$ of the response variable:

$$\theta = f_\theta(\mathbf{Y}) \tag{1}$$

under the LMM. It covers linear combinations of $\mathbf{Y}$ (such as one future realization of the response variable or population and subpopulation means and totals) but also other population and subpopulation characteristics such quantiles and variability measures.

To assess the accuracy of the particular predictor $\hat{\theta}$, firstly, the prediction error is defined as $U = \hat{\theta} - \theta$. Therefore, the well-known RMSE has the following formula:

$$RMSE(\hat{\theta}) = \sqrt{E(\hat{\theta} - \theta)^2} = \sqrt{E(U^2)}. \tag{2}$$

The alternative to the RMSE based on the mean could be the QAPE based on quantiles. It represents the pth quantile of the absolute prediction error $|U|$, see Żądło (2013) and Wolny-Dominiak and Żądło (2020), and it is given by:

$$QAPE_p(\hat{\theta}) = \inf\left\{x : P\left(\left|\hat{\theta} - \theta\right| \leq x\right) \geq p\right\} = \inf\left\{x : P\left(|U| \leq x\right) \geq p\right\} \tag{3}$$

This measure informs that at least $p100\%$ of observed absolute prediction errors are smaller than or equal to $QAPE_p(\hat{\theta})$, while at least $(1-p)100\%$ of them are higher than or equal to $QAPE_p(\hat{\theta})$. Quantiles reflect the relation between the magnitude of the error and the probability of its realization. It means that using the QAPE, it is possible to make a full description of the distribution of prediction errors instead of using the average (reflected by the RMSE). Furthermore, the MSE is the mean of positively (usually very strongly) skewed squared prediction errors, where the mean should not be used as a measure of the central tendency of positively skewed distributions.

The above described accuracy prediction measures RMSE and QAPE can be estimated using the bootstrap techniques. Their estimators as well as the bootstrap distributions of the prediction errors based on any (assumed or misspecified) model are provided in **qape** package, including algorithms where the parallel computing is used.

In the **qape** package, the whole prediction process has its own specific procedure, which can be presented in the following steps.

Procedure 1 The process of prediction, accuracy measures estimation and Monte Carlo simulation analyses in **qape**

1. Define the characteristics of the response variable to predict,
2. provide the information on sample and population values,
3. define the LMM,
4. estimate parameters of the LMM,
5. predict the random variable θ using the chosen class of predictors,
6. estimate the prediction accuracy measures RMSE and QAPE using one of the developed bootstrap algorithms,
7. conduct simulation analyses of properties of predictors and accuracy measures estimators under any (also misspecified) LMM model.

3 The prediction under LMM

The main functions of the **qape** package provide the bootstrap estimation of prediction accuracy measures. However, it must be preceded by the prediction process, including the choice of the LMM and the predictor.

3.1 The model

Let $\mathbf{Y}$ denote the vector of response variables $Y_1, Y_2, ..., Y_N$. Assuming, without a loss of generality, that only the first n realizations of Y_i are observed, $\mathbf{Y}$ can be decomposed as $\mathbf{Y} = \begin{bmatrix}\mathbf{Y}_s^T & \mathbf{Y}_r^T\end{bmatrix}^T$, where $\mathbf{Y}_s$ and $\mathbf{Y}_r$ are of dimension $n \times 1$ and $(N-n) \times 1$, respectively. In all notations, the subscript "s" is used for observed realizations of the variable of interest and "r" for the unobserved ones. Two known matrices of auxiliary variables are also considered, denoted by $\mathbf{X}$ and $\mathbf{Z}$, which are associated with fixed and random effects, respectively. The $\mathbf{X}$ matrix is of dimension $N \times p$, and it consists of p regression variables. It can be decomposed like $\mathbf{Y}$ as follows: $\mathbf{X} = \begin{bmatrix}\mathbf{X}_s^T & \mathbf{X}_r^T\end{bmatrix}^T$, where matrices $\mathbf{X}_s$ and $\mathbf{X}_r$, both known, are of dimension $n \times p$ and $(N-n) \times p$, respectively. Similarly, the $\mathbf{Z}$ matrix of dimension $N \times h$ can be written as follows: $\mathbf{Z} = \begin{bmatrix}\mathbf{Z}_s^T & \mathbf{Z}_r^T\end{bmatrix}^T$, where matrices $\mathbf{Z}_s$ and $\mathbf{Z}_r$, both known, are of dimension $n \times h$ and $(N-n) \times h$, respectively.

Then, let $LMM(\mathbf{X}, \mathbf{Z}, \boldsymbol{\psi})$ denotes the LMM of the following form (e.g. Rao and Molina (2015), p. 98):

$$\begin{cases} \mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{v} + \mathbf{e} \\ E(\mathbf{e}) = \mathbf{0}, E(\mathbf{v}) = \mathbf{0} \\ Var(\mathbf{e}) = \mathbf{R}(\boldsymbol{\delta}), Var(\mathbf{v}) = \mathbf{G}(\boldsymbol{\delta}) \end{cases} \tag{4}$$

The vector of parameters in model (4) is then $\boldsymbol{\psi} = \begin{bmatrix}\boldsymbol{\beta}^T & \boldsymbol{\delta}^T\end{bmatrix}^T$, where $\boldsymbol{\beta}$ is a vector of fixed effects of dimension $p \times 1$ and $\boldsymbol{\delta}$ is a vector of variance components. The random part of the model is described by the known matrix $\mathbf{Z}$, a vector $\mathbf{v}$ of random effects of dimension $h \times 1$ and a vector $\mathbf{e}$ of random components of dimension $N \times 1$, where $\mathbf{e}$ and $\mathbf{v}$ are assumed to be independent. The vector of random components $\mathbf{e}$ will be decomposed similarly to the vector $\mathbf{Y}$, i.e. $\mathbf{e} = \begin{bmatrix}\mathbf{e}_s^T & \mathbf{e}_r^T\end{bmatrix}^T$.

In the residual bootstrap implemented in **qape**, there is a need to re-write the LMM model to take account of the specific structure of data, i.e. the grouping variables taken into account in the random part of the model. In this case, without a loss of the generality, the LMM model can be written as follows:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}_1\mathbf{v}_1 + ... + \mathbf{Z}_l\mathbf{v}_l + ... + \mathbf{Z}_L\mathbf{v}_L + \mathbf{e}, \tag{5}$$

where $\mathbf{v}_1, \ldots, \mathbf{v}_l, \ldots, \mathbf{v}_L$ are independent vectors of random effects assumed for different divisions of the $\mathbf{Y}$ vector (under different grouping of the data) and $\mathbf{Z}_1, \ldots, \mathbf{Z}_l, \ldots, \mathbf{Z}_L$ are known matrices of auxiliary variables associated with random effects. Writing in (5): $\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_1 & \ldots & \mathbf{0} & \ldots & \mathbf{0} \\ \vdots & \ddots & & & \vdots \\ \mathbf{0} & \ldots & \mathbf{Z}_l & \ldots & \mathbf{0} \\ \vdots & & & \ddots & \vdots \\ \mathbf{0} & \ldots & \mathbf{0} & \ldots & \mathbf{Z}_L \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix}\mathbf{v}_1^T & \ldots & \mathbf{v}_l^T & \ldots & \mathbf{v}_L^T\end{bmatrix}^T$ the LMM model is obtained. Let

$$\mathbf{v}_l = \left[\mathbf{v}_{l1}^T \ldots \mathbf{v}_{lk}^T \ldots \mathbf{v}_{lK_l}^T\right]^T \tag{6}$$

be of dimension $K_l J_l \times 1$, where $\mathbf{v}_{lk}$ is of dimension $J_l \times 1$ for all $k = 1, ..., K_l$ and K_l is the number of random effects at the lth level of grouping. Hence, $\mathbf{Z}_l$ is $N \times K_l J_l$. For example, if the random regression coefficient model is considered with two random coefficients where both random effects are subpopulation-specific, where D is the number of subpopulations, then $L = 1$, $K_1 = 2$ and $J_1 = D$.

3.2 Predictors

In the **qape** package, in the general case the predicted characteristic is given by any function of response variables:

$$\theta = f_\theta(\mathbf{Y}). \tag{7}$$

Under the $LMM(\mathbf{X}, \mathbf{Z}, \boldsymbol{\psi})$ model it could be predicted using one of three predictors:

1. Empirical Best Linear Unbiased Predictor (EBLUP),
2. Empirical Best Predictor (EBP) under nested error LMM,

3. PLUG-IN predictor under the LMM.

The first predictor (EBLUP) allows to predict the linear combination of the response variables:

$$\theta = f_\theta(\mathbf{Y}) = \boldsymbol{\gamma}^T\mathbf{Y} = \boldsymbol{\gamma}_s^T\mathbf{Y}_s + \boldsymbol{\gamma}_r^T\mathbf{Y}_r, \tag{8}$$

where $\boldsymbol{\gamma}$ is a vector of weights. In this case, the predicted characteristic θ is basically the linear combination of the response variable. For example, if one of the elements of $\boldsymbol{\gamma}$ equals 1 and the rest of the elements equals 0, then one realization of the response variable is predicted. If all elements in $\boldsymbol{\gamma}$ vector equal 1, then θ becomes the sum of all Y_i's in the whole considered population dataset. The two-stage EBLUP corresponds to the Best Linear Unbiased Predictor (BLUP) introduced in Henderson (1950) and Royall (1976) as:

$$\hat{\theta}^{BLUP}(\boldsymbol{\delta}) = \boldsymbol{\gamma}_s^T\mathbf{Y}_s + \hat{\theta}_r(\boldsymbol{\delta}), \tag{9}$$

where the predictor of the linear combination $\boldsymbol{\gamma}_r^T\mathbf{Y}_r$ of unobserved random variables is given by $\hat{\theta}_r(\boldsymbol{\delta}) = \boldsymbol{\gamma}_r^T\mathbf{X}_r\tilde{\boldsymbol{\beta}}(\boldsymbol{\delta}) + \boldsymbol{\gamma}_r^T\mathbf{Z}_r\tilde{\mathbf{v}}(\boldsymbol{\delta})$, where $\tilde{\boldsymbol{\beta}}(\boldsymbol{\delta})$ is the Best Linear Unbiased Estimator of $\boldsymbol{\beta}$ and $\tilde{\mathbf{v}}(\boldsymbol{\delta})$ is the Best Linear Unbiased Predictor of $\mathbf{v}$, both presented in (4). As shown by Żądło (2017) p. 8094, if $Cov(\mathbf{e}_r, \mathbf{e}_s) = \mathbf{0}$, then the predictor (9) is the BLUP of θ defined as the linear combination (8). Even if $Cov(\mathbf{e}_r, \mathbf{e}_s) \neq \mathbf{0}$, the predictor $\hat{\theta}_r(\boldsymbol{\delta})$ is the Best Linear Unbiased Predictor of the following linear combination of $\boldsymbol{\beta}$ and $\mathbf{v}$: $\boldsymbol{\gamma}_r^T\mathbf{X}_r\boldsymbol{\beta} + \boldsymbol{\gamma}_r^T\mathbf{Z}_r\mathbf{v}$. The EBLUP $\hat{\theta}^{EBLUP}$ is obtained by replacing the vector of variance components $\boldsymbol{\delta}$ in BLUP (9) with the estimator $\hat{\boldsymbol{\delta}}$. If (a) the expectation of the predictor is finite, (b) $\hat{\boldsymbol{\delta}}$ is any even, translation-invariant estimator of $\boldsymbol{\delta}$, (c) the distributions of both random effects and random components are symmetric around $\mathbf{0}$ (not necessarily normal), the EBLUP remains unbiased, as proved by Kackar and Harville (1981).

To introduce the second predictor, called EBP, considered e.g. by Molina and Rao (2010), firstly, the Best Predictor (BP) $\hat{\theta}^{BP}$ of characteristic $\theta(\mathbf{Y})$ has to be defined. It is computed by minimizing the Mean Squared Error $MSE(\hat{\theta}) = E(\hat{\theta} - \theta)^2$ and can be written as $\hat{\theta}^{BP} = E(\theta|\mathbf{Y}_s)$. It means that the conditional distribution of $\mathbf{Y}_r|\mathbf{Y}_s$ must be known to compute its value while at least the parameters of this distribution, denoted by $\boldsymbol{\psi}$ in (4), are unknown. The EBP $\hat{\theta}^{EBP}$ is obtained by replacing these parameters with estimators $\hat{\boldsymbol{\psi}}$. Its value can be computed according to the Monte Carlo procedure presented in the supplementary document for this paper.

The last predictor is the PLUG-IN predictor defined as (e.g. Chwila and Żądło (2019)):

$$\hat{\theta}^{PLUG-IN} = \theta(\begin{bmatrix}\mathbf{Y}_s^T & \hat{\mathbf{Y}}_r^T\end{bmatrix}^T), \tag{10}$$

where $\hat{\mathbf{Y}}_r$ is the vector of fitted values of unobserved random variables under the assumed model (any model specified by the statistician). Under the LMM and if the linear combination of $\mathbf{Y}$ is predicted, the PLUG-IN predictor is the EBLUP, but generally, it is not optimal. However, it was shown in simulation studies that it can have similar or even higher accuracy compared to empirical (estimated) best predictors, where the best predictors minimize the prediction mean squared errors (cf. e.g. Boubeta et al. (2016), Chwila and Żądło (2019), Hobza and Morales (2016)). Moreover, the PLUG-IN predictor is less computationally demanding than the EBP.

3.3 Predictors in qape

To deal with the LMM model, the **qape** package uses the `lmer()` function from the **lme4** package, see Bates et al. (2015). Assuming (4) and based on $\mathbf{Y}_s$, the vector of model parameters $\boldsymbol{\psi} = [\boldsymbol{\beta}^T, \boldsymbol{\delta}^T]^T$ is estimated using the Restricted Maximum Likelihood Method (REML), known to be robust on non-normality, see e.g Jiang (1996), and $\hat{\boldsymbol{\psi}}$ is obtained.

In order to obtain the predictor of θ, one of the three **qape** functions can be applied: `EBLUP()`, `ebpLMMne()` or `plugInLMM()`. Firstly, the characteristic of response variables of interest has to be defined. It is actually obvious for EBLUP, which can be used only to predict the population/subpopulation linear combination (e.g. the sum) by using the argument `gamma` equivalent to the population vector of weights $\boldsymbol{\gamma}$ in (8). For other two predictors, the EBP and the PLUG-IN, the input argument called `thetaFun` has to be given (see $f_\theta(.)$ in (7)). Function `thetaFun` could define one characteristic or a vector of characteristics, for example:

```
> thetaFun1 <- function(x) median(x)
> thetaFun2 <- function(x) c(sum(x), mean(x), sd(x))
```

Secondly, two groups of input arguments, common to all three predictors, has to be provided:

- group 1 - arguments defining the sample and the population
  - `YS` - values of the dependent variable in the sample ($\mathbf{Y}_s$),

  - `reg` - the population matrix of auxiliary variables named in `fixed.part`, `random.part` and `division`,
  - `con` - the population $0-1$ vector with 1s for elements in the sample and 0s for elements which are not in the sample,
- group 2 - arguments defining the model
  - `fixed.part` - fixed-effects terms declared as in `lm4::lmer` function,
  - `random.part` - random-effects terms declared as in `lm4::lmer` function,
  - `weights` - the population vector of weights.

The weights make it possible to include heteroscedasticity of random components in the LMM.

In `EBLUP()` and `plugInLMM()` the random-effects terms of the LMM have to be declared as the input argument `random.part`. The form of the `ebpLMMne` predictor, in turn, requires defining in the `ebpLMMne()` function the so-called `division` argument instead of `random.part`. This input represents the variable dividing the population dataset into subsets, which are taken into account in the nested error linear mixed model with '`division`'-specific random components (presented in supplementary document for this paper).

In the process of prediction, it is often necessary to perform data transformation before estimating the model parameters. An example is the logarithmic scaling of the variable of interest. The **qape** package offers the possibility for declaring the argument `backTrans` to conduct the data back-transformation. Hence, a very flexible solution is used which allows to use any transformation of the response variable such that the back-transformation can be defined. This argument (available in R or defined by the user function) should be the back-transformation function of the already transformed dependent variable used to define the model, e.g. for log-transformed `YS` used as the response variable:

```
> backTrans <- function(x) exp(x)
```

The main output is the value of predictor `thetaP`. For each class of predictors, there are two S3 methods registered for existing generic functions `print` and `summary`. The full list of output arguments is presented in detail in the `qape-manual` file, cf. Wolny-Dominiak and Żądło (2023).

3.4 Radon data and the model

In order to demonstrate the functionality of the package's main functions, in the following examples the `radon` dataset available in **HLMdiag** package (Loy and Hofmann (2014)) is analyzed. It contains the results of a survey measuring radon concentrations in 919 owner-occupied homes in 85 counties of Minnesota (see Figure 1). A study was conducted in 1987-1988 by the Minnesota Department of Health, showing that indoor radon levels are higher in Minnesota compared to typical levels in the U.S. In the data, the response variable `log.radon` (denoted in (11) by $log(Y_{ic})$) is the radon measurement in logarithms of picoCurie per liter. The independent variables, on the other hand, are: `uranium` (x_{1ic}) the average county-level soil uranium content, `basement` (x_{2ic}) the 0-1 variable indicating the level of the home at which the radon measurement was taken - 0 for basement, 1 for the first floor, and `county` (denoted by subscript c in (11)) is county ID.

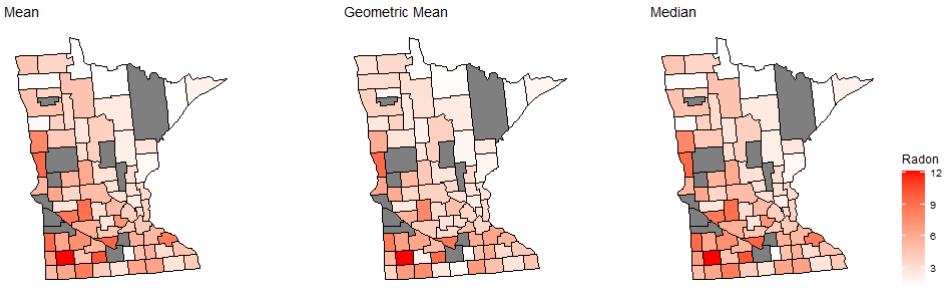


Figure 1: The maps of characteristics of radon concentration in counties in picoCurie per liter. The gray colour means that the value is NA (Not Available)

In all considered examples, the prediction for the county no. 26 (`county == 26`) is conducted and it is assumed that the observations in this county from the first floor (`basement == 1`) are not available (see Figure 2).

The `radon` dataset is widely discussed in the literature. In the paper Nero et al. (1994), the Authors used an ordinary regression model to predict county geometric means of radon concentration using

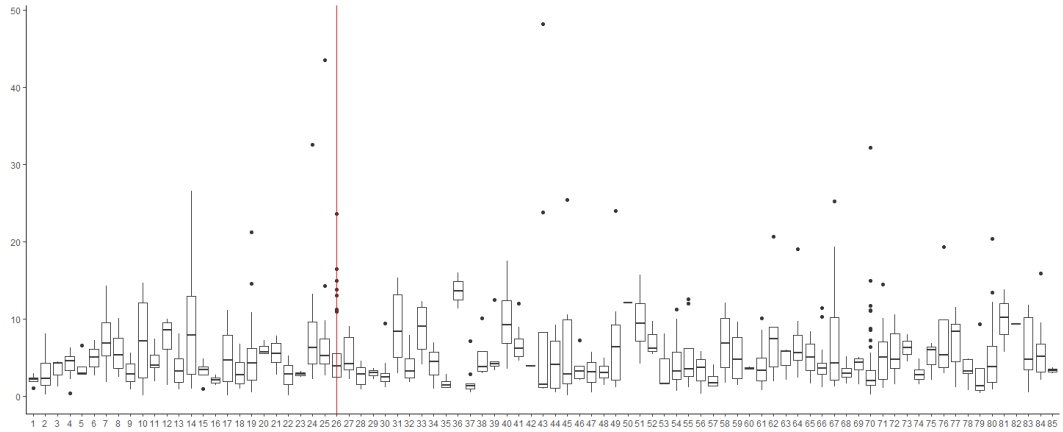


Figure 2: The distributions of radon concentration in picoCurie per liter in counties. The red line indicates county no. 26

surficial soil radium data from the National Uranium Resource Evaluation. In turn, the paper Price et al. (1996) focuses on the prediction of the geometric mean of radon for each county, but using a Bayesian approach. For the `radon` data we use the following model

$$log(Y_{ic}) = \beta_1 x_{1ic} + (\beta_2 + v_{1c})x_{2ic} + \beta_0 + v_{2c} + e_{ic}, \tag{11}$$

where $i = 1, 2, \ldots, N$, $c = 1, 2, \ldots, C$, $N = 919$ observations, $C = 85$ counties, β_1, β_2 and β_0 are unknown fixed effects, v_{1c} and v_{2c} are random effects, e_{ic} are random components, v_{1c}, and e_{ic} are mutually independent, v_{2c} and e_{ic} are mutually independent too, $Cor(v_{1c}, v_{2c}) = \rho$, $v_{1c} \sim (0, \sigma^2_{v_1})$, $v_{2c} \sim (0, \sigma^2_{v_2})$ and $e_{ic} \sim (0, \sigma^2_e)$. As can easily be seen, the considered model is the random coefficient model with two correlated `county`-specific random effects. Its syntax written using the package **lme4** notation is as follows:

```
radon.model <- lmer(log.radon ~ basement + uranium + (basement | county), data = radon)
```

This and similar LMMs are considered, analyzed, and used for the considered dataset in many publications, with a good overview presented in Gelman and Hill (2006). In Gelman and Pardoe (2006), based on their preceding research Price et al. (1996), Lin et al. (1999), Price and Gelman (2005), a very similar model but with additional multivariate normality assumptions is studied, verified and chosen as fitting well to the data within a Bayesian framework. The same model as in Gelman and Pardoe (2006) with its special cases is considered in Cantoni et al. (2021) but within the frequentist approach. Based on 25 measures of explained variation and model selection, the Authors conclude that the same model as considered in our paper (with additional normality assumption, however, which is not used in all cases considered in that paper), "seems the best" (Cantoni et al., 2021, p. 10) for the `radon` data. Further tests of the model are presented by Loy (2013), Loy and Hofmann (2015) and Loy et al. (2017) (see also Cook et al. (2007) for the introduction of the methodology) showing among others: the normality and homescedasticity of random components, the normality of the distribution of the random slope but – what is important for our further considerations – the lack of the normality of the random intercept. Since the problem of choosing and verifying a model for the considered dataset is widely discussed in the literature, we will focus on the issues that are new in this case, namely the problem of prediction and estimation of the prediction accuracy as well as the Monte Carlo analysis of predictors' properties.

3.5 Example 1

This example shows the prediction procedure in the package **qape**. In the first step, it is needed to define all the input arguments that will then be passed to the prediction functions.

```
> Ypop <- radon$log.radon # the population vector of the dependent variable
> # It is assumed that observations from the first floor
> # in county no. 26 are not available:
> con <- rep(1, nrow(radon))
> con[radon$county == 26 & radon$basement == 1] <- 0
> YS <- Ypop[con == 1] # sample vector of the dependent variable
> reg <- dplyr::select(radon, -log.radon) # the population matrix of auxiliary variables
```

```
> fixed.part <- 'basement + uranium' # the fixed part of the considered model
> random.part <- '(basement|county)' # the random part of the considered model
> # The vector of weights to define
> # the predicted linear combination -  the mean for county == 26:
> gamma <-
+   (1 / sum((radon$county == 26))) * ifelse((radon$county == 26), 1, 0)
> estMSE <- TRUE # to include the naive MSE estimator of the EBLUP in the output
```

Then the functions corresponding to each predictor can be used. First, the EBLUP prediction in the package **qape** is presented. As the EBLUP is limited to the linear combination of random variables, the predicted characteristic is simply the arithmetic mean. To be precise, it is the mean of logarithms of measurements (instead of the mean of measurements), because the EBLUP can be used only under the linear (linearized) models. As in the LMM the homescedasticity of random components is assumed, the input argument `weights = NULL` is set up.

```
> myeblup <- EBLUP(YS, fixed.part, random.part, reg, con, gamma,  weights = NULL, estMSE)
> # the value of the predictor of the arithmetic mean
> # of logarithms of radon measurements:
> myeblup$thetaP
[1] 1.306916
> myeblup$neMSE # the value of the naive MSE estimator
[1] 0.002292732
```

Hence, the predicted value of the arithmetic mean of logarithms of radon measurements equals 1.306916 log picoCurie per liter. The estimated root of prediction MSE equals $\sqrt{0.002292732} \approx 0.048$ log picoCurie per liter, but – what is important – it is the value of the naive RMSE estimator (as defined by Rao and Molina, 2015, p. 106), which means that it ignores the decrease of accuracy due to the estimation of model parameters.

The second part of this example shows the prediction of the arithmetic mean, geometric mean and median of radon measurements (not logarithm of radon measurements) in county no. 26 with the use of the PLUG-IN predictor. It requires the setting of two input arguments: `thetaFun` and `backTrans`.

```
> thetaFun <- function(x) {
+   c(mean(x[radon$county == 26]), psych::geometric.mean(x[radon$county == 26]),
+     median(x[radon$county == 26]))
+   }
> backTransExp <- function(x) exp(x) # back-transformation
> myplugin <- plugInLMM(YS, fixed.part, random.part, reg, con, weights = NULL,
+                       backTrans = backTransExp, thetaFun)
> # values of the predictor of arithmetic mean, geometric mean
> # and median of radon measurements:
> myplugin$thetaP
[1] 3.694761 4.553745 3.900000
```

In this case we can conclude that the predicted values of the aritmethmic mean, geometric mean and median in county no. 26 equal: 3.694761, 4.553745 and 3.9 picoCurie per liter, respectively. The problem of prediction accuracy estimation will be discussed in the next sections of the paper.

The **qape** package allows to use the Empirical Best Predictor (EBP) (see the supplementary document for this paper) as well. It provides predicted values of any function of the variable of interest, as the PLUG-IN predictor. However, this requires stronger assumptions to be met. The EBP procedure available in **qape** package is prepared under the assumption of the normality of the variable of interest after any transformation. However, in the case of the considered model for logarithms of radon measurements, the assumption is not met as we mentioned before based on the results presented in the literature. It can also be verified using `normCholTest` function (available in **qape** package) as follows:

```
> normCholTest(radon.model, shapiro.test)$p.value
[1] 2.589407e-08
```

Moreover, due to the fact of very time-consuming iterative procedure used to compute the EBP for the general case, in the **qape** package the function `ebpLMMne` uses a very fast procedure working only for nested error Linear Mixed Models (see Molina and Rao (2010)).

The prediction of any function of the random variables based on cross-sectional data has been considered. Its special case, not presented above but widely discussed in the econometric literature,

is the prediction of one random variable, in this case a radon measurement for one non-observed owner-occupied home. Furthermore, the **qape** package is also designed for prediction based on longitudinal data for current or future periods as shown in examples for the `EBLUP`, `plugInLMM` and `ebpLMMne` functions in the `qape-manual` file, cf. Wolny-Dominiak and Żądło (2023).

4 Bootstrap procedures

The **qape** package provides three main types of bootstrap algorithms: the parametric bootstrap, the residual bootstrap and the double-bootstrap.

The parametric bootstrap procedure is implemented according to González-Manteiga et al. (2007) and González-Manteiga et al. (2008) and could be described in the following steps:

1. based on n observations of the dependent and independent variables ($\mathbf{Y}_s$, $\mathbf{X}_s$ and $\mathbf{Z}_s$) estimate $\boldsymbol{\psi}$ to obtain the vector of estimates $\hat{\boldsymbol{\psi}}$,
2. generate B realizations $y_i^{*(b)}$ of Y_i, under the $LMM(\mathbf{X}, \mathbf{Z}, \hat{\boldsymbol{\psi}})$ and multivariate normality of random effects and random components obtaining $\mathbf{y}^{*(b)} = \begin{bmatrix} y_1^{*(b)} & ... & y_i^{*(b)} & ... & y_N^{*(b)} \end{bmatrix}^T$, where $i = 1, 2, ..., N$ and $b = 1, 2, ..., B$,
3. decompose the vector $\mathbf{y}^{*(b)}$ as follows $\begin{bmatrix} \mathbf{y}_s^{*(b)T} & \mathbf{y}_r^{*(b)T} \end{bmatrix}^T$,
4. in the bth iteration ($b = 1, 2, ..., B$)
   (a) compute the bootstrap realization $\theta^{*(b)} = \theta^{*(b)}(\mathbf{y}^{*(b)}, \hat{\boldsymbol{\psi}})$ of random variable θ,
   (b) obtain the vector of estimates $\hat{\boldsymbol{\psi}}^{*(b)}$ using $\mathbf{y}_s^{*(b)}$ and compute the bootstrap realization of predictor $\hat{\theta}$ denoted by $\hat{\theta}^{*(b)}(\mathbf{y}_s^{*(b)}, \hat{\boldsymbol{\psi}}^{*(b)})$ based on $LMM(\mathbf{X}, \mathbf{Z}, \hat{\boldsymbol{\psi}}^{*(b)})$,
   (c) compute bootstrap realizations of prediction error U^* denoted by u^* and for the bth iteration given by:

$$u^{*(b)} = \hat{\theta}^{*(b)}(\mathbf{y}_s^{*(b)}, \hat{\boldsymbol{\psi}}^{*(b)}) - \theta^{*(b)}(\mathbf{y}^{*(b)}, \hat{\boldsymbol{\psi}}) = \hat{\theta}^{*(b)} - \theta^{*(b)}, \tag{12}$$

5. compute the parametric bootstrap estimators of prediction accuracy measures: RMSE and QAPE replacing prediction errors U in (2) and (3) by their bootstrap realizations.

Another possible method to estimate the prediction accuracy measures is the residual bootstrap. In what follows, we use the notation $srswr(\mathbf{A}, m)$ to indicate the outcome of taking a simple random sample with replacement of size m of rows of matrix $\mathbf{A}$. If $\mathbf{A}$ is a vector, it simplifies to a simple random sample with replacement of size m of elements of $\mathbf{A}$.

To obtain the algorithm of the residual bootstrap, it is enough to replace step 2 of the parametric bootstrap procedure presented above with the following procedure of the population data generation based on (5):

- generate B population vectors of the variable of interest, denoted by $\mathbf{y}^{*(b)}$ as

$$\mathbf{y}^{*(b)} = \mathbf{X}\hat{\boldsymbol{\beta}} + \mathbf{Z}_1\mathbf{v}_1^{*(b)} + ... + \mathbf{Z}_l\mathbf{v}_l^{*(b)} + ... + \mathbf{Z}_L\mathbf{v}_L^{*(b)} + \mathbf{e}^{*(b)}, \tag{13}$$

  where $\hat{\boldsymbol{\beta}}$ is an estimator (e.g. REML) of $\boldsymbol{\beta}$, $\mathbf{e}^{*(b)}$ is a vector of dimension $N \times 1$ defined as $srswr(col_{1 \le i \le n}\hat{e}_i, N)$, where $\hat{e}_i$ ($i = 1, 2, ..., n$) are residuals, $\mathbf{v}_l^{*(b)}$ (for $1, 2, ..., L$) is the vector of dimension $K_l J_l \times 1$ built from the columns of the matrix: $srswr\left(\begin{bmatrix} \hat{\mathbf{v}}_{l1} & \dots & \hat{\mathbf{v}}_{lk} & \dots & \hat{\mathbf{v}}_{lK_l} \end{bmatrix}, J_l\right)$ of dimension $J_l \times K_l$, where $\hat{\mathbf{v}}_{lk}$ are estimates of elements of random effects vector (6).

The next 3–5 steps in this procedure are analogous to steps in the parametric bootstrap procedure.

In the above-described step, it can be seen that if more than one vector of random effect is assumed at the lth level of grouping, then the elements are not sampled with replacement independently. In this case, rows of the matrix formed by these vectors are sampled with replacement.

The residual bootstrap algorithm can also be performed with so-called "correction procedure". This procedure, which can improve the properties of the residual bootstrap estimators due to the underdispersion of the uncorrected residual bootstrap distributions, is presented in the supplementary document for this paper.

5 Bootstrap in qape

Two bootstrap procedures are implemented in separate functions: `bootPar()` (the parametric bootstrap) and `bootRes()` (the residual bootstrap). According to the general Procedure 1, the step preceding the bootstrap procedure in both functions is the definition of the predictor object. It must be one of the following: `EBLUP`, `ebpLMMne` or `plugInLMM`. This object has to be passed to `bootPar()` or `bootRes()` as the input parameter `predictor`. The other input parameters are intuitive: `B` - the number of bootstrap iterations and `p` - order of quantiles in the estimated QAPEs.

The additional input parameter in `bootRes()` is a logical condition called `correction`, which makes it possible to include an additional correction term for both random effects and random components, presented in the supplementary document for this paper, to avoid the problem of underdispersion of residual bootstrap distributions.

The main output values in both functions are basically the measures: `estRMSE` and `estQAPE` computed based on (2) and (3), respectively, where prediction errors are replaced by their bootstrap realizations. There is also the output `error` being the vector of bootstrap realizations of prediction errors, which is useful e.g. in in-depth analysis of the prediction accuracy and for graphical presentation of results. To estimate these accuracy measures, we use below the residual bootstrap with the correction procedure.

As previously stated, our package utilizes the `lmer()` function from the **lme4** package for estimating model parameters. However, this function has been known to generate convergence warnings in certain situations, listed for example by Bates et al. (2015) p. 25, when the estimated variances of random effects are close to zero. Such scenarios may occur when models are estimated for smaller or medium-sized datasets, when complex variance-covariance structures are assumed, or when the grouping variable considered for random effects has only a few levels. Although we have not observed such issues estimating model parameters based on the original dataset required to compute values of the predictors in previous sections, bootstrapping or Monte Carlo simulations are more complex cases. This is because, based on the estimates of model parameters, the values of the dependent variables are generated B times, and then model parameters are estimated in each out of B iterations. Therefore, in at least some iterations, dependent variable values may be randomly generated giving realizations, where the variance of the random effect is relatively close to zero. As a result, estimates of model parameters can be obtained; however, convergence issues implying warnings may occur. In such cases, there are at least two possible solutions. The first option is to discard iterations with warnings, which would imply that the dependent variable would not follow the assumed model as required, but instead only its conditional version with relatively high values of variances of random effects. It will imply overdispersed bootstrap distribution of random effects, which will affect the bias of the bootstrap estimators of accuracy measures. The second option is to consider all generated realizations, despite convergence warnings, as long as the parameters can be estimated for all iterations. We opted for the latter solution, as argued in Bates et al. (2015) p. 25, who noted that "being able to fit a singular model is an advantage: when the best fitting model lies on the boundary of a constrained space".

5.1 Example 2

The analyses presented in Example 1 are continued. We extend the previous results to include the issue of estimating the prediction accuracy of the considered predictors. The use of functions for this estimation primarily requires an object of class predictor, here "myplugin".

```
> class(myplugin)
[1] "plugInLMM"
```

The short chunk of the R code presents the residual bootstrap estimators of the RMSE (`estRMSE`) and the QAPE (`estQAPE`) of the PLUG-IN predictors (`plugin`) of previously analyzed three characteristics of radon measurements in county no. 26: the arithmetic mean, geometric mean and median. In this and subsequent examples we make the computations for relatively high number of iterations allowing, in our opinion, to get reliable results. These results are also used to prepare Figure 3. However, the computations are time-consuming. The supplementary R file contains the same chunks of the code but the number of iterations applied is smaller in order to execute the code swiftly.

```
> # accuracy measures estimates based on
> # the residual bootstrap with the correction:
> B <- 500 # number of bootstrap iterations
> p <- c(0.75, 0.9) # orders of Quantiles of Absolute Prediction Error
> set.seed(1056)
> residBoot <- bootRes(myplugin, B, p, correction = TRUE)
```

```
> # values of estimated RMSEs of the predictor of three characteristics:
> # the arithmetic mean, geometric mean and median of radon measurements, respectively:
> residBoot$estRMSE
[1] 0.1848028 0.2003681 0.2824359
> # values of estimated QAPEs
> # (of order 0.75 in the first row, and of order 0.9 in the second row)
> # of the predictor of three characteristics:
> # the arithmetic mean, geometric mean and median of radon measurements,
> # in the 1st, 2nd and 3rd column, respectively:
> residBoot$estQAPE
         [,1]      [,2]      [,3]
75% 0.1533405 0.2135476 0.2908988
90% 0.2813886 0.3397411 0.4374534
```

Let us concentrate on interpretations of estimators of accuracy measures for the predictor of the geometric mean, i.e. the second value of `residBoot$estRMSE`, and values in the second column of `residBoot$estQAPE`. It is estimated that the average difference between predicted values of the geometric mean and their unknown realizations equals 0.2003681 picoCurie per liter. Furthermore, it is estimated that at least 75% of absolute prediction errors of the predictor of the geometric mean are smaller or equal to 0.2135476 picoCurie per liter and at least 25% of absolute prediction errors of the predictor are higher or equal to 0.2135476 picoCurie per liter. Finally, it is estimated that at least 90% of absolute prediction errors of the predictor of the geometric mean are smaller or equal to 0.3397411 picoCurie per liter and at least 10% of absolute prediction errors of the predictor are higher or equal to 0.3397411 picoCurie per liter. The distributions of bootstrap absolute prediction errors with values of estimated RMSEs and QAPEs for the considered three prediction problems are presented in Figure 3.

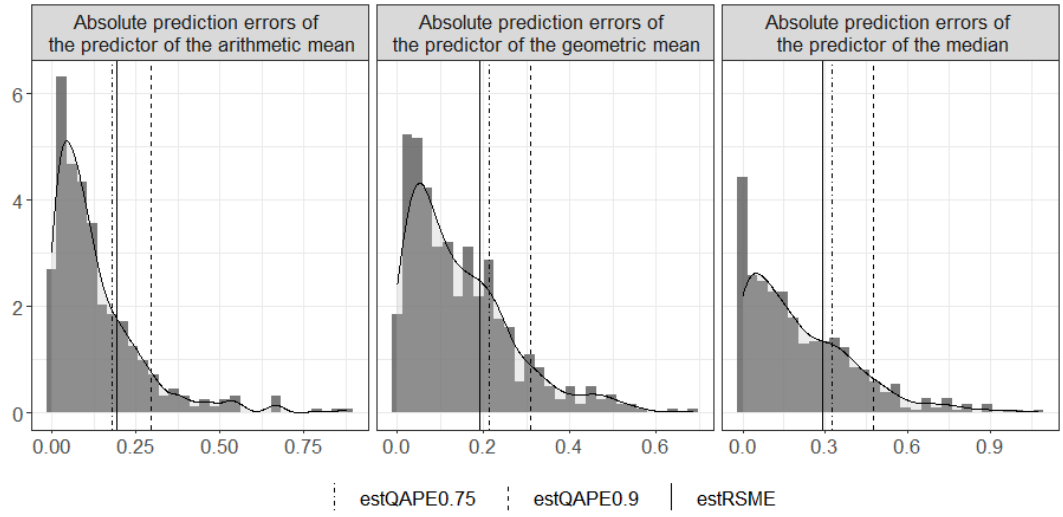


Figure 3: The histograms of bootstrap absolute prediction errors for `myplugin` (for PLUG-IN predictors of the arithmetic mean, geometric mean and median) for $B = 500$

Since the assumption of normality is not met, the parametric bootstrap should not be used in this case. For this reason, we do not present the results for this method below, although – but for illustrative purposes only – they are presented in the supplementary R file. Moreover, these analyses can also be conducted using `bootParFuture()` and `bootResFuture()` functions where parallel computing algorithms are applied. The input arguments and the output of these functions are the same as in `bootPar()` and `bootRes()`. Examples based on these functions are also included in the supplementary R file.

6 Bootstrap under the misspecified model in qape

The qape package also allows to use predictors under a model different from the assumed one (e.g. a simpler or more robust model), but estimate its accuracy under the assumed model. In this case, the parametric and residual bootstrap procedures are implemented in `bootParMis()` and `bootResMis()` functions. These functions allow to estimate the accuracy of two predictors under the model correctly specified for the first of them. Of course, it is expected that the estimated accuracy of the first predictor

will be better than of the second one, but the key issue can be the difference between estimates of accuracy measures. A small difference, even to the second predictor's disadvantage, may be treated by the user as an argument for using the second predictor due to its properties, such as robustness or simplicity.

The considered functions allow to estimate the accuracy of two predictors, which belong to the class `plugInLMM`, under the model used to define the first of them. The remaining arguments are the same as in `bootPar()` and `bootRes()` functions: `B` - the number of bootstrap iterations, and `p` - orders of QAPE estimates to be taken into account.

The output results of `bootParMis()` and `bootResMis()` include – similarly to `bootPar()` and `bootRes()` functions – estimates of the RMSEs and QAPEs of both predictors (denoted here by: `estRMSElmm`, `estRMSElmmMis`, `estQAPElmm` and `estQAPElmmMis`), and boostrap realizations of their prediction errors (`errorLMM` and `errorLMMmis`).

6.1 Example 3

In this example, we study the same accuracy measures as in Example 2, but the aim is to compare the predictor `myplugin` and other predictor defined under the misspecified LMM. First, the misspecified model has to be defined, and a relevant predictor has to be computed.

```
> fixed.part.mis <- '1'
> random.part.mis <- '(1|county)'
> myplugin.mis <- plugInLMM(YS, fixed.part.mis, random.part.mis, reg, con,
+                           weights = NULL, backTrans = backTransExp, thetaFun)
```

Having two objects: `myplugin` and `myplugin.mis`, one can proceed to a comparison by estimating bootstrap prediction accuracy performed using the residual bootstrap with correction procedure. In this case, we estimate the prediction accuracy of these two predictors under the model used to define the first of them.

```
> set.seed(1056)
> residBootMis <- bootResMis(myplugin, myplugin.mis, B, p, correction = TRUE)
> # residual bootstrap with the correction RMSE estimators
> # of 'plugin' of: arithmetic mean, geometric mean and median
> # of radon measurements in county 26:
> residBootMis$estRMSElmm
[1] 0.1848028 0.2003681 0.2824359
> # residual bootstrap with the correction RMSE estimators
> # of 'plugin.mis' of: arithmetic mean, geometric mean and median
> # of radon measurements in county 26:
> residBootMis$estRMSElmmMis
[1] 0.1919184 0.3192304 0.2762137
> # residual bootstrap with the correction QAPE estimators of order 0.75 and 0.9
> # of 'plugin' of: arithmetic mean, geometric mean and median
> # of radon measurements in county 26:
> residBootMis$estQAPElmm
         [,1]      [,2]      [,3]
75% 0.1533405 0.2135476 0.2908988
90% 0.2813886 0.3397411 0.4374534
> # residual bootstrap with the correction QAPE estimators of order 0.75 and 0.9
> # of 'plugin.mis' of: arithmetic mean, geometric mean and median
> # of radon measurements in county 26:
> residBootMis$estQAPElmmMis
         [,1]      [,2]      [,3]
75% 0.2267062 0.3802836 0.3255197
90% 0.2813787 0.4970726 0.4489399
```

The results, presented above, were obtained for the same number of bootstrap iterations as in Example 2 ($B = 500$). If we compare, under the model defined in `plugin`, estimated RMSEs of `plugin` and `plugin.mis` predictors of the geometric mean given by 0.2003681 and 0.3192304 picoCurie per liter, respectively, we can state that the estimated accuracy (measured by RMSE estimators) of the first predictor is better comparing with the second one. If we are not interested in the average accuracy measures but in the right tail of the distribution of prediction errors, we can use estimates of QAPE of order 0.9 to compare the accuracy. The result for the `plugin.mis` of the geometric mean equals to

0.4970726 picoCurie per liter, and it is higher comparing with 0.3397411 picoCurie per liter obtained for `plugin` for the same prediction problem. Hence, in this case, the accuracy comparison based both on the RMSE and QAPE leads to the same finding.

In the previous paragraph, we have focused on the results for the case of prediction of the geometric mean. If the comparison is made for the case of prediction of the arithmetic mean (the first column of output results) or the median (the third column of output results), we will come to the same conclusion regarding the estimated accuracy of `plugin` and `plugin.mis` as in the case of prediction of the geometric mean.

Similarly to the residual bootstrap, the parametric bootstrap procedure `paramBootMis` available in **qape** package can be performed. However, in the considered case the normality assumption is not met (as discussed above) and the procedure is not recommended. The appropriate chunk of the R code is presented in the supplementary R file, but it is solely intended for illustrative purposes.

7 Monte Carlo simulation analyses

In the previous section, our aim was to estimate the prediction accuracy under correctly specified or misspecified model. In this section, we do not estimate the accuracy, but we approximate the true prediction accuracy under the specified model in the Monte Carlo simulation study. The crucial difference is that in this case, the model parameters used are obtained based on the whole population dataset, not the sample. If the number of iterations is large enough, we can treat the computed values of the measures as their true values, which are unknown in practice.

The last step of the analysis in **qape** package presented in Procedure 1 is the Monte Carlo (MC) simulation analysis of:

- properties of predictors
- and properties of parametric, residual and double bootstrap estimators of accuracy measures.

The whole Monte Carlo procedure is as follows.

Procedure 2 Model-based Monte Carlo simulation analyses in **qape**

1. define the population vector of the dependent variable and the population matrix of auxiliary variables,
2. provide the information on the division of the population into the sampled and non-sampled part,
3. define θ - the characteristics of the response variable to be predicted,
4. define the predictors $\hat{\theta}$ and accuracy measures estimators which properties are to be assessed,
5. define the model to be used to generate realizations of the values of the dependent variable and estimate its parameters based on population data,
6. For k=1, 2, ..., K
   - 6.1. generate the population vector of the response variable based on the assumed model,
   - 6.2. based on population data, compute the characteristics θ, denoted by θ_k,
   - 6.3. based on sample data, estimate the parameters of the LMM,
   - 6.4. based on sample data, compute values of predictors $\hat{\theta}$, denoted by $\hat{\theta}_k$,
   - 6.5. based on sample data, estimate the accuracy of $\hat{\theta}$ using bootstrap methods,
7. End For
8. compute accuracy measures of predictors using $\hat{\theta}_k$ and θ_k (for $k = 1, 2, ..., K$),
9. compute accuracy measures of estimators of prediction accuracy measures.

8 Monte Carlo analyses in qape

In order to perform a Monte Carlo (MC) analysis on the properties of predictors, it is necessary to have access to the entire population data for both dependent and independent variables. The function `mcLMMmis()` can be used with the following arguments. Firstly, the population values of the dependent variable (after a necessary transformation) should be declared as `Ypop`. By using the `Ypop` values, we can estimate the model parameters based on the entire population data (assuming that they are known). This allows us to generate values of the dependent variable in the simulation

study that can mimic its distribution in the entire population, not just in the sample. This approach ensures that our simulation study can be an accurate representation of the random process in the entire population, resembling the real-world scenario. Secondly, three predictors: `predictorLMMmis`, `predictorLMM`, `predictorLMM2`, which belong to the class `plugInLMM`, are to be defined. The first one is used only to define the (possibly misspecified) model used to generate population values of the response variables. Accuracy of `predictorLMM` and `predictorLMM2` is assessed in the simulation study. The next two arguments include the number of MC iterations `K` and orders `p` of QAPEs used to assess the prediction accuracy. Finally, it should be noted that it is possible to modify covariance matrices of random components and random effects based on the model defined in `predictorLMMmis`, which are used tThiso generate values of the dependent variable. It is possible by declaring values of `ratioR` and `ratioG` arguments, which the diagonal elements of covariance matrices of random components and random effects, respectively, are divided by.

The output of this function covers the following statistics of both predictors computed in the simulation study: relative biases (`rBlmm` and `rBlmm2`), relative RMSEs (`rRMSElmm` and `rRMSElmm2`) and QAPEs (`QAPElmm` and `QAPElmm2`). Simulation-based prediction errors of both predictors (`errorLMM` and `errorLMM2`) are also taken into account.

8.1 Example 4

In the example, an MC simulation is carried out assuming the `myplugin` predictor. The goal is to approximate the true accuracy of the prediction assuming model (11). Hence, in the package **qape**, all input predictor objects in the function `mcLMMmis` have to be defined as `myplugin`.

```
> # input arguments:
predictorLMMmis <- myplugin # to define the model
predictorLMM <- myplugin # which properties are assessed in the simulation study
predictorLMM2 <- myplugin  # which properties are assessed in the sim. study
```

Except that no modification of covariance matrices has to be used.

```
# diag. elements of the covariance matrix of random components are divided by:
ratioR <- 1
# diag. elements of the covariance matrix of random effects are divided by:
ratioG <- 1
```

We specify the number of Monte Carlo iterations.

```
K <- 500 # the number of MC iterations
```

The analysis is conducted in the object `MC`.

```
> set.seed(1086)
> MC <- mcLMMmis(Ypop, predictorLMMmis, predictorLMM, predictorLMM2,
+                        K, p, ratioR, ratioG)
> # relative bias of 'predictorLMM'
> # of the arithmetic mean, geometric mean and median in county 26 (in %):
> MC$rBlmm
[1] -1.73208393 -0.04053178 -5.22355236
```

Results of the relative biases are obtained. It is seen, that under the assumed model the values of the considered predictor of the geometric mean (the second value of `MC$rBlmm`) are smaller than possible realizations of the geometric mean on average by 0.04053178%. In turn, the relative RMSEs are as follows.

```
> # relative RMSE of 'predictorLMM'
> # of the arithmetic mean, geometric mean and median in county 26 (in %):
> MC$rRMSElmm
[1] 3.429465 4.665810 7.146678
```

In the considered case, the average difference between predicted values of the geometric mean and its possible realizations (the second value of `MC$rRMSElmm`) equals 4.665810%. It should be noted that this value can be treated as the true value of the relative RMSE (if the number of iterations is large enough), not the estimated value obtained in Examples 2 and 3.

Finally, QAPEs of orders 0.75 and 0.9 are considered.

```
> # QAPE of order 0.75 and 0.9 of 'predictorLMM'
> # of the arithmetic mean, geometric mean and median in county 26:
> MC$QAPElmm
         [,1]      [,2]      [,3]
75% 0.1491262 0.1989504 0.2919221
90% 0.2895684 0.2959457 0.4728064
```

Let us interpret the results presented in the second column of `MC$QAPElmm`. At least 75% (90%) of absolute prediction errors of the predictor of the geometric mean are smaller or equal to 0.1989504 (0.2959457) picoCurie per liter and at least 25% (10%) of absolute prediction errors of the predictor are higher or equal to 0.1989504 (0.2959457) picoCurie per liter. Similar to the values of the rRMSEs in the previous code chunk, the values can be considered to be true QAPE values, not the estimates presented in Examples 2 and 3.

In Example 4, the accuracy of one predictor under the model used to define this predictor was presented. A more complex version of the simulation study, where the properties of two predictors are studied under the model defined by the third predictor, is presented in the supplementary R file. What is more, the **qape** package also allows to use `mcBootMis()` function to conduct MC analyses of properties of accuracy measure estimators (estimators of MSEs and QAPEs) of two predictors (which belong to the class `plugInLMM`) declared as arguments. The model used in the simulation study is declared in the first predictor, but the properties of accuracy measures estimators of both predictors are studied. Output results of `mcBootMis()` covers simulation results on properties of different accuracy measures estimators, including the relative biases and relative RMSEs of the parametric bootstrap MSE estimators of both predictors. The same simulation-based statistics but for parametric bootstrap QAPE estimators are also included. Other bootstrap methods, including the residual bootstrap with and without the correction procedure, are also taken into account. The full list of output arguments of `mcBootMis()` function are presented in `qape-manual` file, cf. Wolny-Dominiak and Żądło (2023).

9 Conclusions

The package enables R users to make predictions and assess the accuracy under linear mixed models based on different methods in a fast and intuitive manner – not only based on the RMSE but also based on Quantiles of Absolute Prediction Errors. It also covers functions which allow to conduct Monte Carlo simulation analyses of properties of the methods of users interest. Its main advantage, compared to other packages, is the considerable flexibility in terms of defining the model (as in the **lme4** package) and the predicted characteristic, but also the transformation of the response variable.

In our opinion, the package is useful for scientists, practitioners and decision-makers in all areas of research where accurate estimates and forecasts for different types of data (including cross-sectional and longitudinal data) and for different characteristics play the crucial role. We believe that it will be of special interest to survey statisticians interested in the prediction for subpopulations with small or even zero sample sizes, called small areas.

References

D. Bates, M. Mächler, B. Bolker, and S. Walker. Fitting linear mixed-effects models using lme4. *Journal of Statistical Software*, 67(1):1–48, 2015. doi: 10.18637/jss.v067.i01. [p70, 75]

G. E. Battese, R. M. Harter, and W. A. Fuller. An error-components model for prediction of county crop areas using survey and satellite data. *Journal of the American Statistical Association*, 83(401):28–36, 1988. [p67]

M. Boubeta, M. J. Lombardía, and D. Morales. Empirical best prediction under area-level poisson mixed models. *Test*, 25(3):548–569, 2016. [p67, 70]

J. Breidenbach. *JoSAE: Unit-Level and Area-Level Small Area Estimation*, 2018. URL `https://CRAN.R-project.org/package=JoSAE`. R package version 0.3.0. [p67]

H. Bühlmann and A. Gisler. *A course in credibility theory and its applications*. Springer, 2005. [p67]

E. Cantoni, N. Jacot, and P. Ghisletta. Review and comparison of measures of explained variation and model selection in linear mixed-effects models. *Econometrics and Statistics*, 2021. [p72]

L. Christiaensen, P. Lanjouw, J. Luoto, and D. Stifel. Small area estimation-based prediction methods to track poverty: validation and applications. *The Journal of Economic Inequality*, 10(2):267–297, 2012. [p67]

A. Chwila and T. Żądło. On properties of empirical best predictors. *Communications in Statistics-Simulation and Computation*, pages 1–34, 2019. [p67, 70]

D. Cook, D. F. Swayne, and A. Buja. *Interactive and dynamic graphics for data analysis: with R and GGobi*, volume 1. Springer, 2007. [p72]

R. E. Fay III and R. A. Herriot. Estimates of income for small places: an application of james-stein procedures to census data. *Journal of the American Statistical Association*, 74(366a):269–277, 1979. [p67]

E. W. Frees, V. R. Young, and Y. Luo. A longitudinal data analysis interpretation of credibility models. *Insurance: Mathematics and Economics*, 24(3):229–247, 1999. [p67]

A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press, Cambridge ; New York, 1st edition edition, Dec. 2006. ISBN 978-0-521-68689-1. [p72]

A. Gelman and I. Pardoe. Bayesian measures of explained variance and pooling in multilevel (hierarchical) models. *Technometrics*, 48(2):241–251, 2006. [p72]

W. González-Manteiga, M. J. Lombardía, I. Molina, D. Morales, and L. Santamaría. Estimation of the mean squared error of predictors of small area linear parameters under a logistic mixed model. *Computational Statistics & Data Analysis*, 51:2720–2733, 2007. [p74]

W. González-Manteiga, M. J. Lombardía, I. Molina, D. Morales, and L. Santamaría. Bootstrap mean squared error of small-area eblup. *Journal of Statistical Computation and Simulation*, 78:443–462, 2008. [p74]

C. R. Henderson. Estimation of genetic parameters. *Biometrics*, 6(2):186–187, 1950. [p67, 70]

T. Hobza and D. Morales. Empirical best prediction under unit-level logit mixed models. *Journal of official statistics*, 32(3):661–692, 2016. [p67, 70]

J. Jiang. Reml estimation: asymptotic behavior and related topics. *The Annals of Statistics*, 24(1):255–286, 1996. [p70]

R. N. Kackar and D. A. Harville. Unbiasedness of two-stage estimation and prediction procedures for mixed linear models. *Communications in statistics-theory and methods*, 10(13):1249–1261, 1981. [p70]

A.-K. Kreutzmann, S. Pannier, N. Rojas-Perilla, T. Schmid, M. Templ, and N. Tzavidis. The r package emdi for estimating and mapping regionally disaggregated indicators. *Journal of Statistical Software*, 91, 2019. [p67]

M. D. E. Lefler, D. M. Gonzalez, and A. P. Martin. *saery: Small Area Estimation for Rao and Yu Model*, 2014. URL https://CRAN.R-project.org/package=saery. R package version 1.0. [p67]

C. Lin, A. Gelman, P. N. Price, and D. H. Krantz. Analysis of local decisions using hierarchical modeling, applied to home radon measurement and remediation. *Statistical Science*, 14(3):305–337, 1999. [p72]

A. Loy. *Diagnostics for mixed/hierarchical linear models*. PhD thesis, Iowa State University, 2013. [p72]

A. Loy and H. Hofmann. HLMdiag: A suite of diagnostics for hierarchical linear models in R. *Journal of Statistical Software*, 56(5):1–28, 2014. URL https://www.jstatsoft.org/article/view/v056i05. [p71]

A. Loy and H. Hofmann. Are you normal? the problem of confounded residual structures in hierarchical linear models. *Journal of Computational and Graphical Statistics*, 24(4):1191–1209, 2015. [p72]

A. Loy, H. Hofmann, and D. Cook. Model choice and diagnostics for linear mixed-effects models using statistics on street corners. *Journal of Computational and Graphical Statistics*, 26(3):478–492, 2017. [p72]

I. Molina and Y. Marhuenda. sae: An R package for small area estimation. *The R Journal*, 7(1):81–98, jun 2015. URL https://journal.r-project.org/archive/2015/RJ-2015-007/RJ-2015-007.pdf. [p67]

I. Molina and J. Rao. Small area estimation of poverty indicators. *Canadian Journal of Statistics*, 38(3): 369–385, 2010. [p67, 70, 73]

A. Nero, S. Leiden, D. Nolan, P. Price, S. Rein, K. Revzan, H. Woolenberg, and A. Gadgil. Statistically based methodologies for mapping of radon'actual'concentrations: the case of minnesota. *Radiation Protection Dosimetry*, 56(1-4):215–219, 1994. [p71]

N. Permatasari and A. Ubaidillah. *msae: Multivariate Fay Herriot Models for Small Area Estimation*, 2021. URL https://CRAN.R-project.org/package=msae. R package version 0.1.4. [p67]

P. N. Price and A. Gelman. Should you measure the radon concentration in your home? In *Statistics: A Guide to the Unknown*, pages 149–170. Duxbury Press, Belmont, CA, 4th edition edition, Mar. 2005. ISBN 978-0-534-37282-8. [p72]

P. N. Price, A. V. Nero, and A. Gelman. Bayesian prediction of mean indoor radon concentrations for minnesota counties. *Health Physics*, 71(6):922–936, 1996. [p72]

J. N. Rao and I. Molina. *Small area estimation*. John Wiley & Sons, 2015. [p69, 73]

J. N. Rao and M. Yu. Small-area estimation by combining time-series and cross-sectional data. *Canadian Journal of Statistics*, 22(4):511–528, 1994. [p67]

R. M. Royall. The linear least-squares prediction approach to two-stage sampling. *Journal of the American Statistical Association*, 71(355):657–664, 1976. [p67, 70]

A. Wolny-Dominiak and T. Żądło. On bootstrap estimators of some prediction accuracy measures of loss reserves in a non-life insurance company. *Communications in Statistics-Simulation and Computation*, pages 1–16, 2020. [p67, 68]

A. Wolny-Dominiak and T. Żądło. *qape: Quantile of Absolute Prediction Errors*, 2023. URL https://CRAN.R-project.org/package=qape. R package version 2.0. [p71, 74, 80]

T. Żądło. On parametric bootstrap and alternatives of mse. In *Proceedings of 31st International Conference Mathematical Methods in Economics*, pages 1081–1086, 2013. [p67, 68]

T. Żądło. On prediction of population and subpopulation characteristics for future periods. *Communications in Statistics-Simulation and Computation*, 461(10):8086–8104, 2017. [p70]

Alicja Wolny–Dominiak
Department of Statistical and Mathematical Methods in Economics
University of Economics in Katowice
50, 1 Maja Street
40–287 Katowice
Poland
alicja.wolny-dominiak@uekat.pl
web.ue.katowice.pl/woali/

Tomasz Żądło
Department of Statistics, Econometrics and Mathematics
University of Economics in Katowice
50, 1 Maja Street
40–287 Katowice
Poland
tomasz.zadlo@uekat.pl
web.ue.katowice.pl/zadlo/

text2sdg: An R Package to Monitor Sustainable Development Goals from Text

by Dominik S. Meier, Rui Mata, and Dirk U. Wulff

Abstract Monitoring progress on the United Nations Sustainable Development Goals (SDGs) is important for both academic and non-academic organizations. Existing approaches to monitoring SDGs have focused on specific data types; namely, publications listed in proprietary research databases. We present the text2sdg package for the R language, a user-friendly, open-source package that detects SDGs in text data using different individual query systems, an ensemble of query systems, or custom-made ones. The text2sdg package thereby facilitates the monitoring of SDGs for a wide array of text sources and provides a much-needed basis for validating and improving extant methods to detect SDGs from text.

1 Introduction

The United Nations Sustainable Development Goals (SDGs) have become an important guideline for both governmental and non-governmental organizations to monitor and plan their contributions to social, economic, and environmental transformations. The 17 SDGs cover large areas of application, from ending poverty and improving health, to fostering economic growth and preserving natural resources. As the latest UN report (UN, 2022) attests, the availability of high-quality data is still lacking in many of these areas and progress is needed in identifying data sources that can help monitor work on these goals. Monitoring of SDGs has typically been based on economic and health data, which are often difficult and costly to gather (e.g., `https://sdg-tracker.org/`; `https://www.sdgindex.org/`). One attractive alternative that has emerged from recent scientometric efforts is to detect SDGs from text, such as academic publications. Digitized text represents an attractive resource for monitoring SDGs across a large number of domains because it is becoming widely available in various types of documents, such as news articles, websites, corporate reports, and social media posts. In light of this promise, we developed **text2sdg**, a freely available, open-source tool to enable the SDG-labeling of digitized text and facilitate methodological development in this area. In what follows, we first present some background on existing labeling systems developed to identify SDGs from text, and then provide an overview of the **text2sdg** package, showcase its use in a representative case study, and discuss the promise and limitations of the approach.

2 An overview of SDG labeling systems

The **text2sdg** package provides a user-friendly way to use any existing or custom-made labeling system developed to monitor the 17 SDGs in text sources. The package implements six different labeling systems utilizing different keywords and keyword combination rules, as well as an ensemble model based on the six systems that was trained on labeled data. In the following, we will first introduce the six existing labeling systems, namely the Elsevier, Aurora, Auckland, SIRIS, SDGO, and SDSN systems, before discussing how these systems are combined within the ensemble approach. See table 1 for overview of these labeling systems. We address custom-made labeling systems in a dedicated section below.

2.1 Individual labeling systems

The most prominent SDG labeling system has been developed by *Elsevier*. The Elsevier labeling system was integrated into the Times Higher Education Impact Rankings in 2019, which at the time compared 1,118 universities in their efforts to address the SDGs as measured by the frequency of SDG-related terms in their academic output. The Elsevier queries consist of a list of expert-vetted keywords that are combined using logical AND operators, implying that multiple keywords must be met to label a document as containing a certain SDG. The development of the queries started with an original list of keywords for each SDG that were iteratively fine tuned to maximize the number of identified papers closely reflecting the different SDGs. This involved cropping or combining keywords to reduce the number of irrelevant hits. A detailed report on the initial development of the Elsevier query system is provided by Jayabalasingham et al. (2019). Since the first version, the Elsevier labeling system has been iteratively improved, with the latest versions including additional information specific to academic publications and the Scopus database, such as identifiers of journal names or research areas.

text2sdg implements the latest version without such additional identifiers to broaden the package's applicability beyond the Scopus database (Jayabalasingham et al., 2019).

The Aurora Universities Network's "Societal Impact and Relevance of Research" working group started to develop a labeling system in 2017 to increase the visibility of research into the SDGs. Aurora's queries were developed with the goal of identifying SDG-related academic publications included in the Scopus database. Consequently, the syntax of Aurora queries is similar to the Scopus query language and the Elsevier system. However, in contrast to the Elsevier system, the queries combine keywords in a more complex fashion, recruiting Boolean (AND, OR) and proximity operators (e.g., w/3, implying within 3 words). As a result, Aurora's keywords are more specific, possibly leading to a smaller number of false positives. The initial version of the Aurora system only included terms that appear in the SDG policy text of the targets and indicators defined by the United Nations. Subsequent versions expanded on this by including additional keywords that reflect academic terminology. **text2sdg** implements version 5.0 of the Aurora labeling system (Vanderfeesten et al., 2020a). This version represents an improvement on previous versions based on a survey study (Vanderfeesten et al., 2020b) and modifications inspired in other efforts, namely those from Elsevier (above) and SIRIS (introduced below).

The Auckland labeling system (Wang et al., 2023) was developed by the University of Auckland to better understand how their research output contributes to the SDGs. To construct the queries, they used text-mining techniques to extract global and local SDG keywords from publication metadata. These keywords were then sorted according to the number of publications that include the terms and according to the keywords' term frequency–inverse document frequency. The top-ranked keywords were then manually reviewed to only retain keywords that are relevant. The selected keywords were then combined with those of SDSN and Elsevier as well as UN SDG Indicators to form the final SDG keyword list. These queries formed the basis for the Auckland queries, which make use of Boolean (AND, OR) operators and wildcards (e.g., "*").

The SIRIS labeling system (Duran-Silva et al., 2019) was created by SIRIS Academic as part of the "science4sdgs" project to better understand how science, innovation efforts, and technology related to the SDGs. The SIRIS queries were constructed in a five-step procedure. First, an initial list of keywords was extracted from the United Nations official list of goals, targets and indicators. Second, the list was manually enriched on a basis of a review of SDG relevant literature. Third, a word2vec model that was trained on a text corpus created from the enriched keyword list was used to identify keywords that were semantically related to the initial list. Fourth, using the DBpedia API, keywords were added that, according to the Wikipedia corpus, had a categorical relationship with the initial list. Fifth, and finally, the keyword list was manually revised. The queries of the SIRIS labeling system primarily consist of individual keywords that occasionally are combined with a logical AND. **text2sdg** implements the only currently available version of the SIRIS labeling system (Duran-Silva et al., 2019) .

The Open Source SDG (OSDG) project combines data from multiple sources to detect SDGs in text. Instead of developing yet another query system, OSDG's aim was to re-use and integrate existing knowledge by combining multiple SDG "ontologies" (i.e., query systems). OSDG has also made use of Microsoft Academic Graph to improve their results but because our query-based system cannot implement this procedure, we adopt the simpler ontology initially proposed by OSDG, which we refer to as "SDGO" in the package. The labeling system was based on central keywords in the SDG United Nations description (e.g."sanitation" was classified into "SDG6") and then manually expanded with additional relevant keywords identified from a corpus of already labeled documents. The resulting keyword list only makes use of the OR operator. **text2sdg** implements the only currently available version of these queries (Bautista, 2019).

Finally, the Sustainable Development Solutions Network (SDSN, Sustainable Development Solutions Network (SDSN), 2021) labeling system contains SDG-specific keywords compiled in a collaborative effort by several universities from the Sustainable Development Solutions Network (SDSN) Australia, New Zealand & Pacific Network. This query system was developed to detect SDGs in large sets of university-related text data, such as course listings or research publications. The authors used United Nations documents, Google searches, and personal communications as sources for the keywords. This query system combines keywords with OR operators and does not make use of AND operators.

All in all, as can be seen in Table 1, the latter systems differ from the former four in the complexity of their queries: the Elsevier, Aurora, Auckland, and SIRIS systems make use of keyword-combination queries and other criteria, such as proximity operators, whereas SDGO and SDSN only make use of keywords.

Labeling system	SDGs covered	Query operators	Unique keywords per SDG (mean & SD)	Example query (SDG-01)
Elsevier	SDG 1 - SDG 16	OR, AND, wildcards	74.9 (21.7)	"extreme poverty"
Aurora	SDG 1 - SDG 17	OR, AND, wildcards, proximity search	89.6 (31.6)	("poverty") W/3 ("chronic*" OR "extreme")
Auckland	SDG 1 - SDG 16	OR, AND, wildcards	183 (46.5)	"poverty eradication"
SIRIS	SDG 1 - SDG 16	OR, AND	262 (148)	("anti-poverty") AND ("poverty" OR "vulnerability")
SDGO	SDG 1 - SDG 17	OR	245 (236)	"absolute poverty"
SDSN	SDG 1 - SDG 17	OR	62.6 (16.8)	"End poverty"

Table 1: Overview of the labeling systems implemented in **text2sdg**. Legend: OR—keywords are combined using logical ORs, implying that only the keywords must be matched to assign an SDG label; AND—keywords are combined using logical ANDs, implying that multiple keywords must be matched to assign an SDG label; wildcards—keywords are matched considering different keyword parts; proximity search—keywords must co-occur within a certain word window to assign an SDG label.

2.2 The ensemble labeling system

In another publication (Wulff et al., 2023), we evaluated the accuracy of the six labeling systems implemented by **text2sdg** and a rival approach (i.e., OSDG Pukelis et al., 2020) using expert-labeled data sets. These analyses lead to three critical observations. First, the accuracy of SDG classifications was reasonable for all systems, but varied considerably as a function of the data set. This is because the systems differ in how liberal or conservative they assign SDGs to texts due to differences in the types of query operators they employ. Specifically, employing only OR-operators, SDGO and SDSN were considerably more liberal, whereas the other four systems employing additional operators were more conservative. In other words, the systems implement different trade-offs between sensitivity (i.e., true-positive rate) and specificity (i.e., true-negative rate). As a result, SDGO and SDSN outperformed the other systems for SDG-rich documents and vice versa. In addition to these differences in accuracy, we observed critical biases in SDG profiles, with the systems overemphasizing different sets of SDGs, and strong dependencies between SDG predictions and document length. To address these limitations, we developed an ensemble model approach that uses the the predictions of the six systems and document length as inputs to a random forest model. After training with expert-labeled and synthetic data, the ensemble model showed better out-of-sample accuracy, lower false alarm rates, and smaller biases than any individual labeling system Wulff et al. (2023). As a result, this ensemble model is also made available through **text2sdg** using a dedicated function.

In the following sections, we provide an overview over the **text2sdg** R package and demonstrate how its functions can be used to run to detect and analyze SDGs in text.

3 The text2sdg package

3.1 Motivation for text2sdg

Despite the effort put into developing various labeling systems and their great promise in addressing the SDG-related data scarcity, extant implementations of these approaches are not without shortcomings. First, the labeling systems were mostly developed to be used within academic citation databases (e.g., Scopus) and are not easily applied to other text sources. Second, existing implementations lack transparent ways to communicate which features are matched to which documents or how they compare between a choice of labeling systems. We alleviate these shortcomings by providing an open-source solution, **text2sdg**, that lets users detect SDGs in any kind of text using any of the above-mentioned systems, and ensemble of systems, or even customized, user-made labeling systems. The package provides a common framework for implementing the different extant or novel approaches and makes it easy to quantitatively compare and visualize their results.

3.2 Overview of text2sdg package

At the heart of the **text2sdg** package are the Lucene-style queries that are used to detect SDGs in text and the ensemble models that build on these queries. The queries map text features (i.e., words or a combination of words) to SDGs. For example, a text that contains the words "fisheries" and

Function Name	Description
`detect_sdg`	identifies SDGs in text using an ensemble model that draws on the six labeling systems (Elsevier, Aurora, Auckland, SIRIS, SDGO, SDSN).
`detect_sdg_systems`	identifies SDGs in text by using labeling systems (Elsevier, Aurora, Auckland, SIRIS, SDGO, SDSN).
detect_any	similar to `detect_sdg` but identifies SDGs in text using user-defined queries.
`crosstab_sdg`	crosstab_sdg takes the output of detect_sdg, detect_sdg_systems, or detect_any as input and determines correlations between either query systems or SDGs.
`plot_sdg`	takes the output of detect_sdg, detect_sdg_systems, or detect_any as input and produces adjustable barplots illustrating the hit frequencies produced by the different query systems.

Table 2: Overview of package functions

"marine" would be mapped to SDG 14 (i.e., conserve and sustainably use the oceans, seas and marine resources for sustainable development) by the Aurora system. To enable the use of such queries in R, the **text2sdg** package recruits the **corpustools** package (Welbers and van Atteveldt, 2021). **corpustools** has been built to implement complex search queries and execute them efficiently for large amounts of text. Based on this, **text2sdg** provides several functions that implement extant labeling systems, facilitate the specification of new labeling systems, and analyze and visualize search results. Table 2 gives an overview of the **text2sdg** core functions.

The main functions of **text2sdg** are `detect_sdg` and `detect_sdg_systems`, which implement the ensemble model approach (Wulff et al., 2023) and the implemented labeling systems, respectively, to identify SDGs in texts. The texts are provided to these functions via the `text` argument as either a character vector or an object of class `"tCorpus"` from **corpustools**. All other arguments are optional. By default, the `detect_sdg_systems` function runs only the Aurora, Auckland, Elsevier, and SIRIS systems, but the set systems can be extended to all six systems using the `system` argument. The functions further allow customization of the set of SDGs using the `sdgs` argument and return a `tibble` with one row per hit that has the following columns (and types) (italic column names only present in the tibble returned by `detect_sdg_systems`):

- document (factor) - index of element in the character vector or corpus supply for text
- sdg (character) - labels indicating the matched SDGs
- system (character) - the query or ensemble system that produced the match
- *query_id* (integer) - identifier of query in the query system
- *features* (character) - words in the document that were matched by the query
- hit (numeric) - running index of matches for each system

Further details on the `detect_sdg` and `detect_sdg_systems` functions and their output will be presented in the next section.

The `detect_any` function implements the same functionality as `detect_sdg_systems`, but permits the user to specify customized or self-defined queries. These queries are specified via the `queries` argument and must follow the syntax of the **corpustools** package (see Practical Considerations section for more details).

To support the interpretation of SDG labels generated by `detect_sdg`, `detect_sdg_systems` and `detect_any`, **text2sdg** further provides the `plot_sdg` and `crosstab_sdg` functions. The `plot_sdg` function visualizes the distribution of SDG labels identified in documents by means of a customizable barplot showing SDG frequencies for the different labeling systems. The `crosstab_sdg` function helps reveal patterns of label co-occurrences either across SDGs or systems, which can be controlled using the `compare` argument.

4 Demonstrating the functionality of text2sdg

To showcase the functionalities of the **text2sdg** package we analyze the publicly available p3 dataset of the Swiss National Science Foundation (SNSF) that lists research projects funded by the SNSF. In addition to demonstrating **text2sdg**, the case study will permit us to discuss practical issues concerning the labeling of SDGs, including relevant differences between labeling systems. The data to reproduce the analyses presented below can be found at `https://doi.org/10.5281/zenodo.11060662` (Meier, 2024).

4.1 Preparing the SNSF projects data

The SNSF projects data was downloaded from https://data.snf.ch/datasets. As of March 2022, the p3 database included information on 81,237 research projects. From the data, we removed 54,288 projects where the abstract was absent or not written in English. This left us with a total of 26,949 projects. To ready this data for analysis, we read it using the `readr` function of the **readr** package (Wickham et al., 2021), producing a `tibble` named `projects`. A reduced version of this `tibble` is included in the **text2sdg** package and available through the `projects` object after **text2sdg** has been loaded.

4.2 Using `detect_sdg` and `detect_sdg_systems` to detect SDGs

To label the abstracts in `projects` using `detect_sdg`, we only have to supply the character vector that includes the abstracts to the `text` argument of the `detect_sdg` function. In addition the example below makes use of the `synthetic` argument to implement the `"equal"` (default) and `"triple"` version of the ensemble model. As a result, two versions of the ensemble model are run that were trained on an equal amount of synthetic (non-SDG related) and expert-labeled data and three times the amount of synthetic than labeled data, respectively. A larger amount of synthetic data in training lowers the false-positive rate, but also compromises accuracy (cf. Wulff et al., 2023, for more details).

```
# detect SDGs
> sdgs_ensemble <- detect_sdg(text = projects,
+                             synthetic = c("equal","triple"))
Running systems
Obtaining text lengths
Building features
Running ensemble

> head(sdgs_ensemble)
# A tibble: 6 × 4
  document sdg    system            hit
  <fct>    <chr>  <chr>           <int>
1 22       SDG-06 Ensemble equal   2539
2 39       SDG-03 Ensemble equal    498
3 39       SDG-07 Ensemble equal   2953
4 39       SDG-08 Ensemble equal   4080
5 41       SDG-13 Ensemble equal   5690
6 41       SDG-13 Ensemble triple  3684
```

The first two columns of the `tibble` returned by `detect_sdg` show the document and SDGs identified by the model. Further columns show the system producing the hit and a running hit index for a given system. As the predictions of the six individual labeling systems are used as input for the ensemble models, they will be computed in the background. The user can access these predictions by calling `attr(sdgs_ensemble,"system_hits")`. Alternatively, the user can use the `detect_sdg_systems` function, which provides additional options for customization.

As with the `detect_sdg` function, the `detect_sdg_systems` function requires a character vector as input to the `text` argument. In addition, the example below specifies two optional arguments. First, to indicate that all six systems should be run, rather than the default of only Aurora, Auckland, Elsevier, and SIRIS, we supply a character vector of all six systems' names to the `systems` argument. Second, we explicitly set the `output` argument to "`features`", which in contrast to `output =` "`documents`" delivers more detailed information about which keywords that triggered the SDG labels.

```
# detect SDGs
> sdgs <- detect_sdg_systems(text = projects,
+                            systems = c("Aurora", "Elsevier", "Auckland", "SIRIS", "SDSN", "SDGO"),
+                            output = "features")
Running Aurora
Running Elsevier
Running Auckland
Running SIRIS
Running SDSN
Running SDGO
```

```
> head(sdgs)
# A tibble: 6 × 6
  document sdg    system query_id features      hit
  <fct>    <chr>  <chr>     <dbl> <chr>       <int>
1 1        SDG-01 SDSN        392 sustainable     4
2 1        SDG-02 SDSN        376 maize           3
3 1        SDG-02 SDSN        629 sustainable     8
4 1        SDG-08 SDGO       3968 work            1
5 1        SDG-08 SDSN        812 work           11
6 1        SDG-09 SDSN        483 research        6
```

The above `tibble` produced by **text2sdg** contains for every combination of document, SDG, system, and query (columns 1 to 4), the query feature (keyword) that triggered the label (column 5), and a hit index for a given system (column 6). The first row of the `tibble` thus shows that the query 392 within SDSN labeled document number 1 with SDG-01, because the document included the feature *sustainable*, and that this was the fourth hit produced by the SDSN system. It is important to note that, in other cases, multiple features of a query might be matched, which will result in multiple rows per combination of document, SDG, system, and query. This can be avoided by setting the `output` argument to "`documents`", in which case all features' hits of such combinations will be grouped into a single row.

4.3 Analyzing the SDG labels

To visualize the distribution of SDG labels across SDGs and systems in the `sdgs tibble`, we apply the `plot_sdg` function. By default, `plot_sdg` shows a barplot of the number of documents labeled by each of the SDGs, with the frequencies associated with the different systems stacked on top of each other. The function counts a maximum of one hit per document-system-SDG combination. Duplicate combinations resulting from hits by multiple queries or keywords in queries will be suppressed by default and the function returns a message reporting the number of cases affected.

```
> plot_sdg(sdgs)
139048 duplicate hits removed. Set remove_duplicates = FALSE to retain duplicates.
```

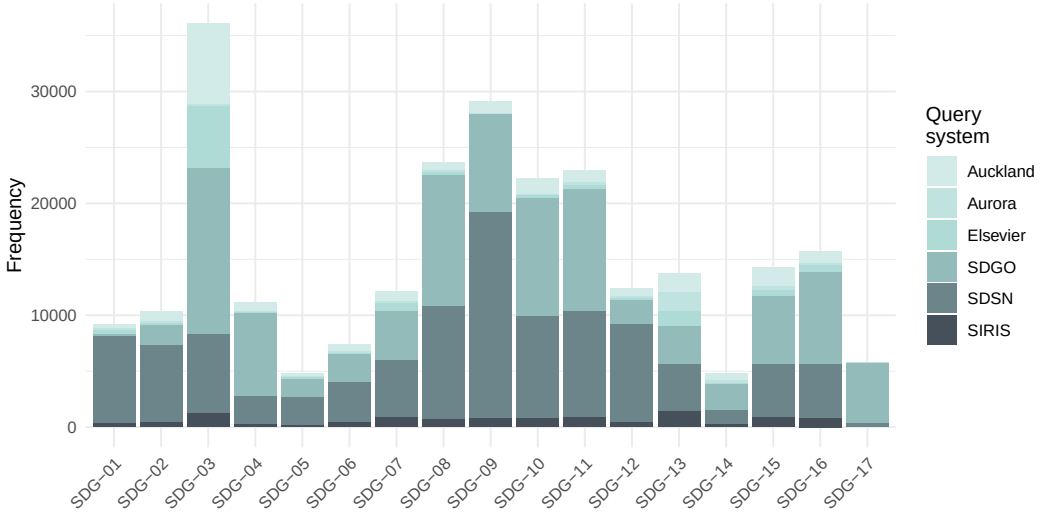


Figure 1: Default plot of distribution of detected SDGs.

The plot produced by `plot_sdg` (Figure 1) shows considerable differences in the frequency of different SDGs, with SDGs 3 ("Good Health and Well-Being") and 9 ("Industry, Innovation And Infrastructure") being most frequent and SDGs 5 ("Gender Equality") and 14 ("Life Below Water") being least frequent. Furthermore, there are substantial differences in the number of labels produced by different systems, with SDSN and SDGO having produced many more labels than the other three systems.

To customize the visualization of SDG frequencies, the `plot_sdg` function provides several additional arguments. For instance, by setting `sdg_titles` to `TRUE`, the SDG titles will be added to the annotation of the plot. Other arguments are `normalize` to show probabilities instead of frequencies, `color` to change the filling of bars, and `remove_duplicates` to eliminate duplicate document-system-SDG combinations. Furthermore, as `plot_sdg` is built on **ggplot2** (Wickham, 2016), the function can easily be extended by functions from the **ggplot2** universe. To illustrate these points, the code below generates a plot (Figure 2) that includes SDG titles and separates the results of the different SDG systems using facets.

```
> plot_sdg(sdgs,
+          sdg_titles = TRUE) +
+   ggplot2::facet_wrap(~system, ncol= 1, scales = "free_y")
139048 duplicate hits removed. Set remove_duplicates = FALSE to retain duplicates.
```

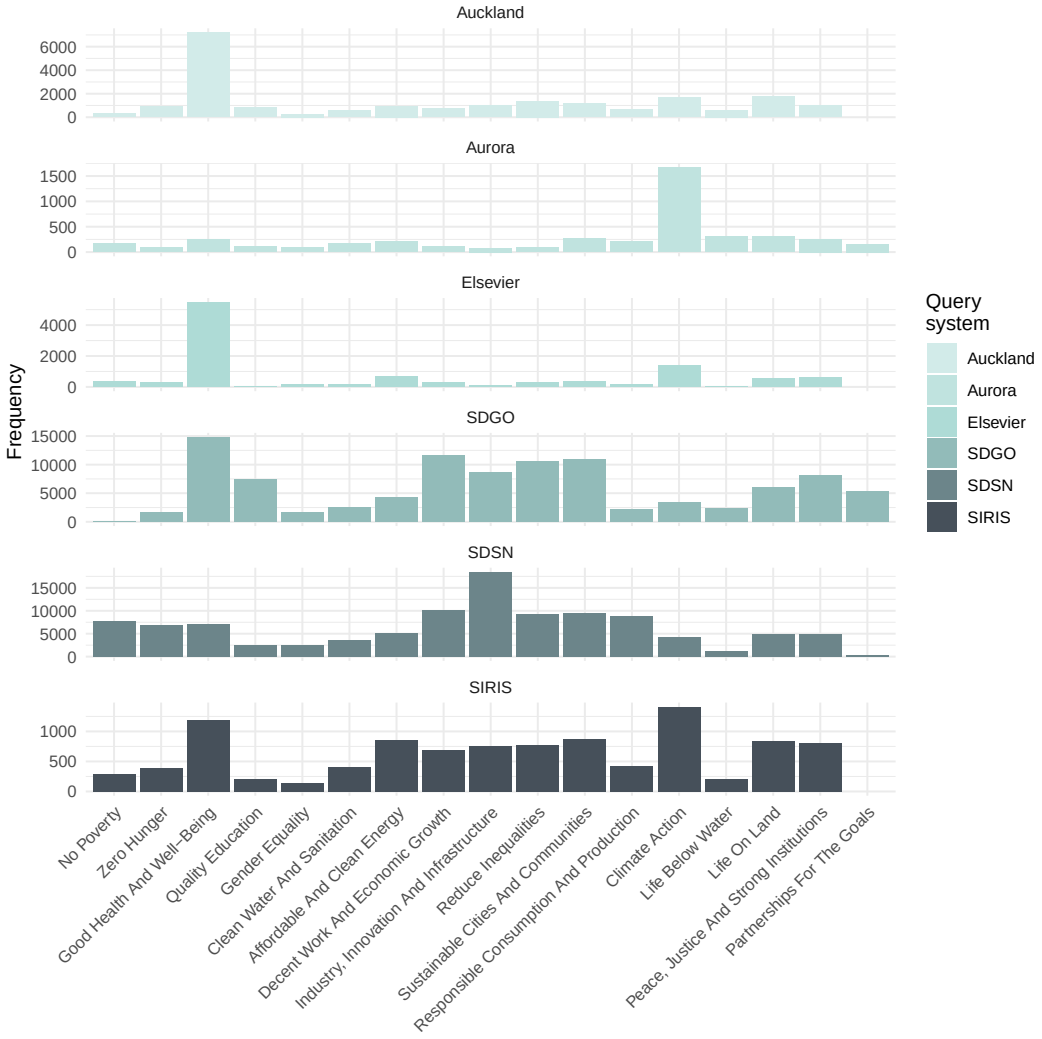


Figure 2: Distribution of detected SDGs facetted by system.

The separation of systems better illustrates the results of systems that produce fewer hits and helps compare the results across systems. This reveals, for instance, that in the Elsevier system SDG 3 ("Good Health and Well-Being") was most prominent, whereas in the Aurora system this was SDG 13 ("Climate Action"). These results highlight that the different labeling systems do not necessarily agree concerning the assignment of SDGs to documents.

To quantify the commonalities and differences between labeling systems, **text2sdg** provides the `crosstab_sdg` function. The function evaluates the level of alignment across either systems (the default) or SDGs by calculating ϕ coefficients between the vectors of labels. We supply the `hits` argument of the function with the `sdgs tibble` containing the labels produced by `detect_sdg`. Note that the function only considers distinct combinations of documents, systems and SDGs, irrespective of whether the `detect_sdg` function was run using `output = "documents"` or `output = "features"`.

```
> crosstab_sdg(sdgs)
          Auckland    Aurora  Elsevier      SDGO      SDSN     SIRIS
Auckland 1.0000000 0.3345247 0.6676524 0.3314806 0.2896650 0.4115387
Aurora   0.3345247 1.0000000 0.3256877 0.1614586 0.1569791 0.3703457
Elsevier 0.6676524 0.3256877 1.0000000 0.2642918 0.2192051 0.3538272
SDGO     0.3314806 0.1614586 0.2642918 1.0000000 0.3722997 0.2244774
SDSN     0.2896650 0.1569791 0.2192051 0.3722997 1.0000000 0.2330684
SIRIS    0.4115387 0.3703457 0.3538272 0.2244774 0.2330684 1.0000000
```

The output of `crosstab_sdg()` for the SNSF projects reveals two noteworthy insights. First, the correspondence between the labels of different systems is rather small, as indicated by ϕ coefficients that are mostly smaller than 0.4. Second, there are two groups of systems that are more similar to one another. On the one hand, Elsevier, Auckland, Aurora, and SIRIS, and, on the other hand, SDGO and SDSN. These groups correspond to differences in query operators, with the former four including AND operators in their queries, whereas the latter two do not. `crosstab_sdg()` can also be called with the output from the ensemble models.

```
> crosstab_sdg(sdgs_ensemble)
                Ensemble equal Ensemble triple
Ensemble equal       1.0000000       0.8127837
Ensemble triple      0.8127837       1.0000000
```

It can further be informative to analyze the correlations between SDGs. To do this, we set the `compare` argument in `crosstab_sdg()` to `"sdgs"`. The output below shows the result for the first six SDGs by setting `sdgs = 1:6`. It can be seen that certain pairs of SDGs—in particular, SDG-01 and SDG-02—co-occur more frequently. These results may provide insights into the co-occurrence structure of SDGs in the data at hand. However, these results can also highlight the importance of considering similarities between queries targeting different SDGs.

```
> crosstab_sdg(sdgs, compare = "sdgs", sdgs = 1:6)
           SDG-01     SDG-02     SDG-03     SDG-04     SDG-05     SDG-06
SDG-01 1.00000000 0.47455139 0.04811778 0.07928418 0.14252372 0.16622948
SDG-02 0.47455139 1.00000000 0.10611662 0.06751253 0.09338952 0.17504027
SDG-03 0.04811778 0.10611662 1.00000000 0.18092227 0.10936179 0.04882173
SDG-04 0.07928418 0.06751253 0.18092227 1.00000000 0.11791600 0.07887042
SDG-05 0.14252372 0.09338952 0.10936179 0.11791600 1.00000000 0.04603253
SDG-06 0.16622948 0.17504027 0.04882173 0.07887042 0.04603253 1.00000000
```

5 Practical considerations

5.1 Specifying user-defined labeling systems

The query systems implemented in **text2sdg** represent important efforts to systematize the monitoring of SDGs from text. Nevertheless, these efforts are still relatively young and validations of the systems are largely missing, creating a need for continued development. **text2sdg** supports the further development of new SDG labeling systems by providing the `detect_any` function. In this section, we provide additional detail on using this feature of **text2sdg**.

The `detect_any` function also uses **corpustools** as the back-end. This implies that new queries must be specified to match the syntax of **corpustools**. The syntax supports standard Boolean operators (AND, OR, and NOT), wildcard operators, and proximity search. Boolean operators control how different keywords are combined in a query. For instance, the query "marine OR fisheries" matches text that contains either of these two words whereas the query "marine AND fisheries" only matches text that contains both words. Corpustools also allows to specify common query wildcard operators [1]. The wildcard operators ? and $*$ allow the specification of variable word parts. For instance, the question mark operator ? matches one unknown character or no character at all, e.g., "?ish" would match "fish", "dish", or "ish". The asterisk operator $*$, by contrast, matches any number of unknown characters, e.g., "*ish" would match "fish" but also "Swedish". Both wildcards can be used at the start, within or end

[1] Note that the meaning of these wildcards differs from regex wildcards.

of a term. Proximity search extends a Boolean AND, by requiring that two keywords have no more than defined distances to one another. For instance, "climate change"∼3 specifies matches in which "climate" and "change" both occur no more than three words apart. A complete description of the **corpustools** syntax is presented in the **corpustools** vignette and documentation.

To supply a user-defined labeling system to `detect_any`, the queries must be placed in a `data.frame` or `tibble` that additionally includes a column specifying the labeling system's name and a column of SDG labels corresponding to the queries.

- system (character) - name of the labeling systems.
- queries (character) - user-defined queries.
- sdg (integer) - SDGs labels assigned by queries.

The example below illustrates the application of a user-defined labeling system using `detect_any`. First, a `tibble` is defined that includes three rows, one for each of three different queries stored in the `query` column. The system is called `"my_example_system"` in the `system` column and each of the queries is assigned SDG-14 in the `sdg` column. Note that specification of the labeling system need not be made in R, but can easily be outsourced to a spreadsheet that is then processed into a `tibble`. Second, the system is supplied to the `system` argument of the `detect_any` function, along with the texts (here, the SNSF abstracts). The output is analogous to the output of the `detect_sdg_systems` function (for brevity, we only show the first three lines of the output).

```
> # definition of query set
> my_example_system <- tibble::tibble(system = "my_example_system",
+                              query = c("marine AND fisheries",
+                                        "('marine fisheries') AND sea",
+                                        "?ish"),
+                              sdg = c(14,14,14))
> detect_any(text = projects,
+            system = my_example_system)
# A tibble: 591 × 6
   document sdg    system            query_id features   hit
   <fct>    <chr>  <chr>                <dbl> <chr>    <int>
 1 6        SDG-14 my_example_system        3 wish       122
 2 134      SDG-14 my_example_system        3 wish        18
 3 241      SDG-14 my_example_system        3 fish        59
```

5.2 Applying text2sdg to non-English data

The queries of the labeling systems implemented by **text2sdg** are in English, implying that texts in other languages must first be translated to English. We assessed feasibility and whether translation affects the reliability of SDG labels by making use of back translation with one language we are most familiar with (German). To this end, we first translated 1,500 randomly selected SNSF project abstracts from English to German and from German to English and then compared the labels of the original English and back-translated English abstracts. We carried out the translation using the DeepL translation engine (www.deepl.com/translator).

Table 3 shows the results of this analysis. Overall, the correlations as measured by the *phi*-coefficient are very high. The systems showed correlations above or equal to 0.88, with Elsevier and Auckland showing the highest value of 0.93. Considering that our analysis involves not only one, but two translation steps—from German to English and back—these results suggest that **text2sdg** can be applied to non-English text, such as German, with very high accuracy. One should note, however, that the quality of translation may vary across languages and translation engines so additional work is needed to compare performance across different languages.

Aurora	Elsevier	Auckland	SIRIS	SDSN	SDGO
0.91	0.93	0.93	0.88	0.91	0.91

Table 3: *phi*-coefficient between the labels for the original English text and the labels for the back-translated (English-German-English) English text

5.3 Estimating the runtime of text2sdg

The analysis of text data can be computationally intense. To provide some guidance on the expected runtime of **text2sdg** for data with different numbers of documents and different document lengths, we carried out several experiments. For this purpose, we first simulated documents by concatenating 10, 100, 1,000, or 10,000 words drawn randomly according to word frequencies in Wikipedia and combined 1, 10, 100, or 1,000 thus-generated documents into simulated data sets. Then we evaluated the runtime of **text2sdg** separately by system for the simulated data sets.

Figure 3 shows the average runtime in seconds across 7,000 repetitions of each combination of document length and number of documents for each of the labeling systems. The results highlight noteworthy points. First, runtime is primarily a function of the number of words, irrespective of how words are distributed across documents. Second, the runtime per words decreases as the number of words increases, which is due to a constant overhead associated with optimizing the labeling systems' queries. Third, there are considerable differences in the runtime between systems, which is, in part, due to the functions' overhead and, in part, due to differences in number and complexity of queries. The fastest system is Elsevier, processing 10 million words in roughly one minute; the slowest system is SIRIS, processing 10 million words in about 40 minutes. Overall, these experiments highlight that **text2sdg** can efficiently process large amounts of text, but also that some care should be exercised when dealing with extremely large or many texts. In such cases, it may be advisable to rely on more efficient labeling systems, such as Elsevier or SDSN.

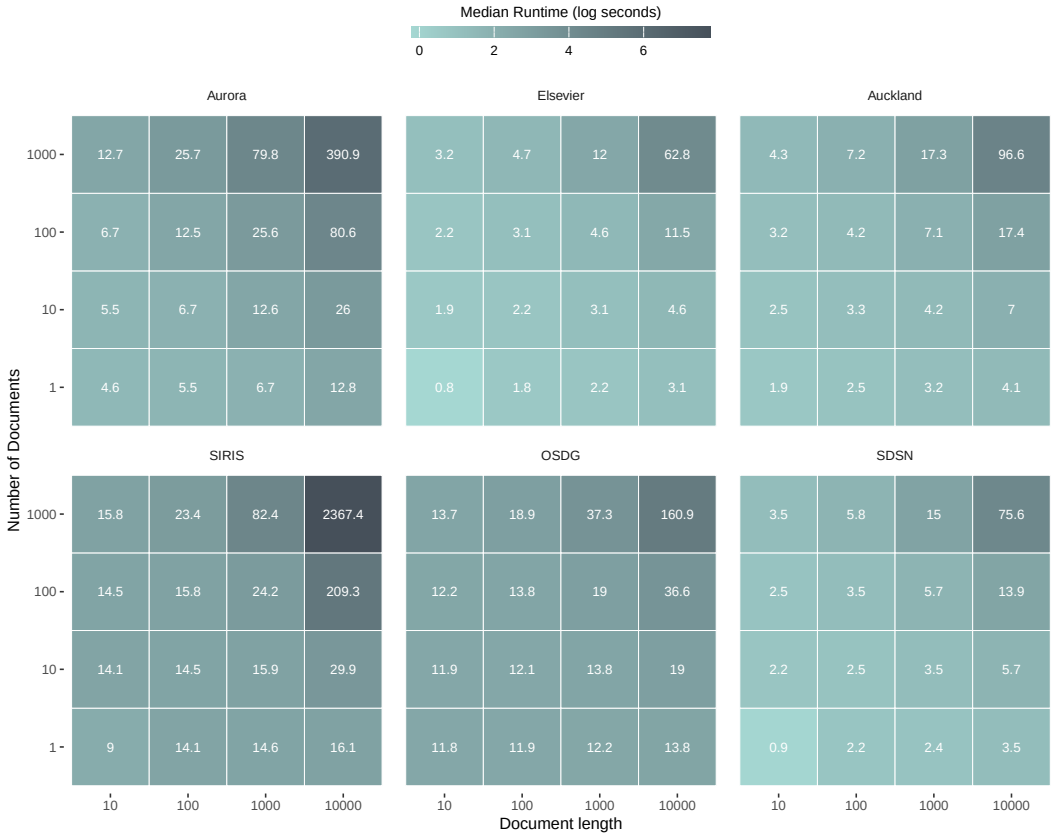


Figure 3: Median runtime as a function of number of documents and document length using 6 different query systems. Each cell reflects the average runtime of 7,000 runs with numbers reflecting the median runtime in seconds and color reflecting the logarithm of the median runtime in seconds.

6 Other approaches to detecting SDGs in text

There are a number of other approaches to detecting SDGs in text. First, there are approaches outside the R ecosystem. One such tool is the European Union's SDG Mapper (`https://knowsdgs.jrc.ec.europa.eu/sdgmapper`) that produces an analysis of SDGs per document using an online interface in which registered users can upload single documents. Another prominent example is the OSDG tool developed by the SDG Ai Lab of the United Nations in collaboration with private partners. It can

detect SDGs in text that is provided through the OSDG website (https://osdg.ai/) or, if granted access, through an API. The OSDG tool builds on the SDG Ontology (SDGO) that is also implemented in **text2sdg**. OSDG additionally leverages a machine learning tool that was trained on expert-labeled data to make the final predictions (Pukelis et al., 2022). One advantage of OSDG relative to **text2sdg** is that it allows to detect SDGs in 15 different languages. This is done by using translation of the input text into English before passing it through the OSDG workflow. While this is convenient to the user, the same outcome can be achieved with our package by making use of translation models through, for example the **deeplr** R package. As our proof-of-concept above has shown, **text2sdg** can be used with non-English text (e.g., German) with very high accuracy by using such an approach.

Second, there are currently, to our knowledge, two other R packages aimed at providing methods for the automated detection of SDGs in text. The **SDGdetector** package is based on a custom query system that was generated by pooling several existing query systems and manual adaptions. The resulting labeling system permits finer-grained predictions on the level of SDG targets [2]. However, the method is computationally taxing and limited to texts that are shorter than 750 characters or approximately 150 words. The **SDGmapR** package builds on publicly available SDG keywords that are assigned weights that indicate the degree to which a keyword reflects a given SDG. The package computes SDG weights for each text by adding up the weights of the keywords that were found in the text. The larger this weight, the larger should be the likelihood that the text is related to a specified SDG. The advantage of this approach is that it permits customization of the decision boundary (i.e., the weight needed to count a text as SDG related). However, the package does not give the user a binary decision regarding whether a text relates to a given SDG. None of the two packages offers an ensemble model that can be used to categorize the presence of SDGs as is the case with **text2sdg**.

7 Discussion

The **text2sdg** package offers an open and easily accessible way of detecting SDGs in text using both individual query systems, a state-of-the-art ensemble model that combines queries from extant systems (Wulff et al., 2023), as well as custom-made queries.

While our package implements several query-based methods to detect SDGs in text as well as a state-of-the-art ensemble model, the field of detecting SDGs in text is rapidly evolving. Our aim is to continuously update **text2sdg** as new open source methods of detecting SDGs in text are released. Bundling many systems in a coherent API is not only convenient for users, but also helps catalyze development of new and hopefully more accurate methods by making it easy to compare the performance of the different systems. We deliberately incorporated functions that allow users to implement and test their own query systems to facilitate this process. We also encourage others to contribute to **text2sdg** by adding new systems or by expanding the existing functionalities to analyse the output of the systems.

Indeed, although the systems implemented by **text2sdg** have been shown to achieve high accuracy (Wulff et al., 2023), it is important to stress that these systems must be further developed to increase their accuracy for a greater number of document types. Two approaches can help in achieving this. First, unsupervised methods such as topic models (Grün and Hornik, 2011) or semantic network analysis (Siew et al., 2019) can help in identifying novel linguistic patterns for the detection of SDGs. One should note, however, that unsupervised methods are no replacement for top-down, rule-based methods as implemented by **text2sdg**, because of the strong requirement to compare results across data sets, analyses, and time, which require a clear set of benchmarks that are not simply data-driven. Second, recent transformer based models (Reimers and Gurevych, 2019) could be leveraged to learn more complex relationships between specific linguistic patterns and SDGs. However, the field will have to work towards producing more balanced training data before the full potential of these approaches can be exploited. Moreover, one should note that transformer models are computationally expensive and often limited to short text due to architecture constraints (Ding et al., 2020). Whether such developments will emerge and can be ultimately integrated into **text2sdg** or will represent alternative approaches remains an open question.

8 Conclusion

In this article, we introduced a new R package, **text2sdg**, designed to help identify SDGs from text. The package promises to help detect SDGs in text sources using different existing or custom-made

[2] Each SDG has several targets that are operationalized with indicators (SDG/targets/indicators). For example the first target of SDG 1 reads as follows: "By 2030, eradicate extreme poverty for all people everywhere, currently measured as people living on less than $1.25 a day".

labeling systems as well as a high-performance ensemble model that builds on these labeling systems. Our case study and additional analyses suggest that the approach can handle both sources in English as well as translations, allows user-friendly use of novel queries, and provides reasonably efficient performance for analysing large corpora.

References

N. Bautista. Sdg ontology. 2019. URL https://doi.org/10.6084/m9.figshare.11106113.v1. [p84]

M. Ding, C. Zhou, H. Yang, and J. Tang. Cogltx: Applying bert to long texts. *Advances in Neural Information Processing Systems*, 33:12792–12804, 2020. [p93]

N. Duran-Silva, E. Fuster, F. A. Massucci, and A. Quinquillà. A controlled vocabulary defining the semantic perimeter of Sustainable Development Goals, Dec. 2019. URL https://doi.org/10.5281/zenodo.3567769. [p84]

B. Grün and K. Hornik. topicmodels: An r package for fitting topic models. *Journal of statistical software*, 40:1–30, 2011. doi: https://doi.org/10.18637/jss.v040.i13. [p93]

B. Jayabalasingham, R. Boverhof, K. Agnew, and L. Klein. Identifying research supporting the united nations sustainable development goals. *Mendeley Data*, 1, 2019. URL https://doi.org/10.17632/87txkw7khs.1. [p83, 84]

D. S. Meier. Descriptions of snsf-funded research projects, Apr. 2024. URL https://doi.org/10.5281/zenodo.11060662. [p86]

L. Pukelis, N. B. Puig, M. Skrynik, and V. Stanciauskas. Osdg–open-source approach to classify text data by un sustainable development goals (sdgs). *arXiv preprint arXiv:2005.14569*, 2020. doi: https://doi.org/10.48550/arXiv.2005.14569. [p85]

L. Pukelis, N. Bautista-Puig, G. Statulevičiūtė, V. Stančiauskas, G. Dikmener, and D. Akylbekova. Osdg 2.0: a multilingual tool for classifying text data by un sustainable development goals (sdgs), 2022. URL https://arxiv.org/abs/2211.11252. [p93]

N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019. doi: https://doi.org/10.48550/arXiv.1908.10084. [p93]

C. S. Siew, D. U. Wulff, N. M. Beckage, and Y. N. Kenett. Cognitive network science: A review of research on cognition through the lens of network representations, processes, and dynamics. *Complexity*, 2019, 2019. doi: https://doi.org/10.1155/2019/2108423. [p93]

Sustainable Development Solutions Network (SDSN). Compiled list of sdg keywords, 2021. URL https://ap-unsdsn.org/regional-initiatives/universities-sdgs/. [p84]

UN. *The Sustainable Development Goals Report 2022*. United Nations, 2022. [p83]

M. Vanderfeesten, R. Otten, and E. Spielberg. Search Queries for "Mapping Research Output to the Sustainable Development Goals (SDGs)" v5.0, July 2020a. URL https://doi.org/10.5281/zenodo.3817445. [p84]

M. Vanderfeesten, E. Spielberg, and Y. Gunes. Survey data of "Mapping Research Output to the Sustainable Development Goals (SDGs)", May 2020b. URL https://doi.org/10.5281/zenodo.3813230. [p84]

W. Wang, W. Kang, and J. Mu. Mapping research to the sustainable development goals (sdgs). 2023. URL https://doi.org/10.21203/rs.3.rs-2544385/v1. [p84]

K. Welbers and W. van Atteveldt. *corpustools: Managing, Querying and Analyzing Tokenized Text*, 2021. URL https://CRAN.R-project.org/package=corpustools. R package version 0.4.8. [p86]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p89]

H. Wickham, J. Hester, and J. Bryan. *readr: Read Rectangular Text Data*, 2021. URL https://CRAN.R-project.org/package=readr. R package version 2.1.1. [p87]

D. U. Wulff, D. S. Meier, and R. Mata. Using novel data and ensemble models to improve automated labeling of sustainable development goals. *arXiv preprint arXiv:2301.11353*, 2023. [p85, 86, 87, 93]

Dominik S. Meier
University of Basel
Steinengraben 22 4051 Basel
Switzerland
(ORCID: 0000-0002-3999-1388)
dominik.meier@unibas.ch

Rui Mata
University of Basel
Missionsstrasse 60-62 4055 Basel
Switzerland
(ORCID: 0000-0002-1679-906X)
rui.mata@unibas.ch

Dirk U. Wulff
University of Basel
Missionsstrasse 60-62 4055 Basel
Switzerland
(ORCID: 0000-0002-4008-8022)
dirk.wulff@unibas.ch

GenMarkov: Modeling Generalized Multivariate Markov Chains in R

by Carolina Vasconcelos and Bruno Damásio

Abstract This article proposes a new generalization of the Multivariate Markov Chains (MMC) model. The future values of a Markov chain commonly depend on only the past values of the chain in an autoregressive fashion. The generalization proposed in this work also considers exogenous variables that can be deterministic or stochastic. Furthermore, the effects of the MMC's past values and the effects of pre-determined or exogenous covariates are considered in our model by considering a non-homogeneous Markov chain. The Monte Carlo simulation study findings showed that our model consistently detected a non-homogeneous Markov chain. Besides, an empirical illustration demonstrated the relevance of this new model by estimating probability transition matrices over the space state of the exogenous variable. An additional and practical contribution of this work is the development of a novel R package with this generalization.

1 Introduction

Multivariate Markov chains (MMC) have a wide range of applications, in various fields. Hence, several studies and generalizations of the MMC models have been made. However, the availability of packages that allow the estimation and application of these models are scarce, and most of these methods use algorithms and software that are not broadly available or can only be applied in particular situations. In the last few years, R software has been gaining importance in the field of statistical computing. This phenomenon might be because it is free and open-source software, which compiles and runs on a wide variety of operating systems. Specifically, in R software, there are some available packages related to Markov chains (MC) and MMC. For example, the **march** package (Maitre and Emery, 2020; Berchtold et al., 2020) allows the computation of various Markovian models for categorical data, including homogeneous Markov chains of any order, MTD models, Hidden Markov models, and Double Chain Markov Models. Ogier Maitre developed this package with contributions from Andre Berchtold, Kevin Emery, Oliver Buschor, and Andre Berchtold maintains it. All the models computed by this package are for univariate categorical data. The **markovchain** package (Spedicato, 2017) contains functions and methods to create and manage discrete-time Markov chains. In addition, it includes functions to perform statistical and probabilistic analysis (analysis of their structural proprieties). Finally, the **DTMCPack** package (Nicholson, 2013) contains a series of functions that aid in both simulating and determining the properties of finite, discrete-time, discrete-state Markov chains. There are two main functions: `DTMC` and `MultDTMC`, which produce n iterations of a Markov Chain(s) based on transition probabilities and an initial distribution given by the user, for the univariate and multivariate case, respectively. This last package is the only one available in R for MMC. In general, the work on MMC models is mostly based on improving the estimation methods and/or making the model more parsimonious. In this work, we aim to develop a new generalization that considers exogenous variables. Specifically, the effects of the MMC's past values and the effects of pre-determined or exogenous covariates are considered in our model by considering a non-homogeneous Markov chain. Additionally, we address statistical inference and implement these methods in an R package. The R package includes three functions: `multimtd`, `multimtd_probit` and `mmcx`. The first two functions estimate the MTD model for multivariate categorical data, with Chings's specification (Ching et al., 2002) and with the Probit specification (Nicolau, 2014), respectively. The last function allows the estimation of our proposed model, the Generalized Multivariate Markov Chain (GMMC) model. The R package, **GenMarkov**, with these three functions is available in the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=GenMarkov`.

2 Multivariate Markov chains

Markov chains can be appropriate for representing dependencies between successive observations of a random variable. However, when the order of the chain or the number of possible values increases, Markov chains have lack parsimony. In this context, Jacobs and Lewis (1978), Pegram (1980) and Logan (1981) proposed several models for HOMC. Notwithstanding these developments, the Mixture Transition Distribution model (Raftery, 1985) proved to be more suitable to model HOMC, which overshadowed the previously proposed models. Several relevant extensions of the MTD model emerged: the Multimatrix MTD (Berchtold, 1995, 1996), which allowed modeling the MTD by using a different $m \times m$ transition matrix for each lag, the Infinite-Lag MTD model that assumes an infinite

lag order ($l = \infty$), which was first considered by Mehran (1989) and later developed by Le et al. (1996) in a more general context. Finally, the MTD with General State Spaces allowed modeling more general processes with an arbitrary space state (Martin and Raftery, 1987; Adke and Deshmukh, 1988; Wong and Li, 2001). Although the MTD model presents a more parsimonious approach to model Markov chains with order higher than one, it has weaknesses. Namely, when considering more than one data sequence, one represents the MMC as a HOMC, by expanding the state-space. This approach could result in a more complex probability transition matrix. Consequently, this can make the estimation unfeasible as the order, states, and the number of data sequences increase. Additionally, the model assumes the same transition matrix for each lag. In this setting, Ching et al. (2002) determined an alternative to handle the unfeasibility of the conventional multivariate Markov chain (MMC) by proposing a model with fewer parameters. The model developed is essentially the same as the MTD. However, it considers a different $m \times m$ transition matrix for each lag and considers more than one data sequence. In the proposed multivariate Markov chain model, Ching et al. (2002) assume the following relationship:

Let $x_t^{(j)}$ be the state vector of the jth sequence at time t. If the jth sequence is in state l at time t then

$$x_{t+1}^{(j)} = \sum_{k=1}^{s} \lambda_{jk} P^{(jk)} x_t^{(k)}, \text{for } j = 1, 2, \ldots, s \tag{1}$$

where $0 \leq \lambda_{jk} \leq 1$ for $j \leq s, k \leq s$ and $\sum_{k=1}^{s} \lambda_{jk} = 1$ for $j = 1, 2, \ldots, s$. The λ_{jk} can be interpreted as the mixing probability of the jth state to the kth state.

The state probability distribution of the kth sequence at time $(t+1)$ depends on the weighted average of $P^{(jk)} x_t^{(k)}$. Here $P^{(jk)}$ is a transition probability matrix from the states in the kth sequence to the states in the jth sequence and $x_t^{(k)}$ is the state probability distribution of the kth sequences at time t. In matrix form:

$$\underline{x}_{t+1}^{(j)} \equiv \begin{bmatrix} x_{t+1}^{(1)} \\ \vdots \\ x_{t+1}^{(s)} \end{bmatrix} = \begin{bmatrix} \lambda_{11} P^{(11)} & \ldots & \lambda_{1s} P^{(1s)} \\ \vdots & \ddots & \vdots \\ \lambda_{s1} P^{(s1)} & \ldots & \lambda_{ss} P^{(ss)} \end{bmatrix} \begin{bmatrix} x_t^{(1)} \\ \vdots \\ x_t^{(s)} \end{bmatrix} \equiv Q \underline{x}_t \tag{2}$$

where Q is an $ms \times ms$ block matrix ($s \times s$ blocks of $m \times m$ matrices) and x_t is a stacked ms column vector (s vectors, each one with m rows).

The matrices $P^{(jk)}$ can be estimated for each data sequence by counting the transition frequency from the states in the kth sequence to those in the jth sequence, obtaining the transition frequency matrix for the data sequence. After normalization, the estimates of the transition probability matrices, i.e., $\widehat{P}^{(jk)}$, are obtained. Regarding the λ_{jk} coefficients, the estimation method proposed by Ching et al. (2002) involves the following optimization problem:

$$min_{\lambda} max_i |[\sum_{k=1}^{m} \lambda_{jk} \widehat{P}^{(jk)} \widehat{\boldsymbol{x}}^{(k)} - \widehat{\boldsymbol{x}}^{(j)}]| \tag{3}$$

$$\text{s.t.} \sum_{k=1}^{s} \lambda_{jk} \text{ and } \lambda_{jk} \geq 0$$

Besides this, different models have been proposed for multiple categorical data sequences. Kijima et al. (2002) proposed a parsimonious MMC model to simulate correlated credit risks. Siu et al. (2005) proposed an easy to implement model; however, its applicability was limited by the number of parameters involved. Ching et al. (2008) proposed a simplified model based on an assumption proposed in Zhang et al. (2006). Zhu and Ching (2010) proposed a method of estimation based on minimizing the prediction error with equality and inequality restrictions and Nicolau and Riedlinger (2014) proposed a new approach to estimate MMC which avoids imposing restrictions on the parameters, based on non-linear least squares estimation, facilitating the model estimation and the statistical inference. Berchtold (2003) proposed a MTD model for heteroscedastic time series. Lastly, Wang et al. (2014) proposed a new multivariate Markov chain model to reduce the number of parameters. Thus, generally, the models used in the published papers were developed by Ching et al. (2002) or were a consequent generalization of them and addressed the MMC as an end in itself. In Damásio (2013) and Damásio and Nicolau (2014), a different and innovative concept was proposed: the usage of MMC as regressors in a certain model. Hence, given that the MMC Granger causes a specific dependent variable, and taking advantage of the information about the past state interactions between the MMC categories, it was possible to forecast the current dependent variable more accurately. Other relevant contributions are related to the optimization algorithm, as in Lèbre and Bourguignon (2008) and Chen and Lio (2009), and to empirical applications (Ching et al., 2003; Ching and Ng, 2006; Damásio, 2018;

Damásio and Mendonça, 2019; Damásio and Mendonça, 2020). Also, Damásio and Nicolau (2020) proposed a new methodology for detecting and testing the presence multiple structural breaks in a Markov chain occurring at unknown dates. In the vast majority of MMC models' studies, a positive correlation between the different data sequences is assumed due to the restrictions imposed. This aspect means it is always considered that at moment t, an increase in a state probability for a data sequence has an increasing impact on another data sequence, for time $t+1$. Thereupon, if one has a negative correlation between series, the parameter estimates are forced to be zero. The solution to this problem is very straightforward; one can relax the assumptions and not assume the constraints. However, that means the results produced by the model will no longer be probabilities. Raftery and Tavaré (1994) presented an alternative, by dropping the positivity condition and imposing another set of restrictions. Ching et al. (2008) also tackled this issue and proposed a method where one splits the Q matrix into the sum of two other matrices and one represents the positive correlations and another the negative correlations. Also, in Nicolau (2014), a specification completely free from constraints, inspired by the MTD model, was proposed, facilitating the estimation procedure and, at the same time, providing a more accurate specification for $P_j(i_0|i_1,\ldots,i_s)$. The model was:

$$P_j(i_0|i_1,\ldots,i_s) = P_j^{\Phi}(i_0|i_1,\ldots,i_s) := \frac{\Phi(\eta_{j0}+\eta_{j1}P(i_0|i_1)+\cdots+\eta_{js}P(i_0|i_s))}{\sum_{k=1}^{m}\Phi(\eta_{j0}+\eta_{j1}P(k|i_1)+\cdots+\eta_{js}P(k|i_s))} \tag{4}$$

where $n_{ji} \in \mathbb{R}(j=1,\ldots,s; i=1,\ldots,m)$ and Φ is the (cumulative) standard normal distribution function.

This specification is denoted as and MTD-Probit model. The log-likelihood is given by:

$$LL = \sum_{i_1,i_2,\ldots,i_{i_s},i_0} n_{i_1,i_2,\ldots,i_{i_s},i_0} log(P_j^{\Phi}(i_0|i_1,\ldots,i_s)) \tag{5}$$

and the maximum likelihood estimator is defined, as usual, as $\widehat{\eta} = \arg\max_{n_{j1},\ldots,n_{js}} LL$. The parameters $P_{jk}(i_0|i_1)$, k =1,...,s can be estimated in advance, through the consistent and unbiased estimators proposed by Ching et al. (2002):

$$\widehat{P}_{jk}(i_0|i_1) = \frac{n_{i_1 i_0}}{\sum_{i_0=1}^{n} n_{i_1 i_0}} \tag{6}$$

This specification can be superior to the MTD because the estimation procedure is easier, and the standard numerical optimization routines can be easily applied in the absence of constraints. However, similarly to the standard MTD, the likelihood is not a strictly concave function on the entire parameter state-space, thus the choice of starting values is still important. Additionally, the model describes a broader range of possible dependencies since the parameters are not constrained. Moreover, this proposed model is more accurate than the MTD model. For more details on this, see Nicolau (2014).

Overall, the published work on MMC models was mostly based on improving the estimation methods and/or making the model more parsimonious. In Damásio (2013) and Damásio and Nicolau (2014), a different approach was used, and the work developed focused on the usage of MMC as regressors in a certain model. Notably, it showed that an MMC can improve the forecast of a dependent variable. In a way, it demonstrated that an MMC can be an end in itself, but it can be an instrument to reach an end or a purpose. In this work, the opposite will be developed: instead of considering an MMC as regressors, a model in which a vector with pre-determined exogenous variables is part of $\mathcal{F}_{t-1}$ is proposed.

3 Covariates in Markov chain models

Regarding the inclusion of covariates in Markov chains models, Regier (1968) proposed a two-state Markov chain model, where the transition matrix probabilities were a function of a parameter, q, that described the tendency of the subject to move from state to state. Kalbfleisch and Lawless (1985) proposed a panel data analysis method under a continuous-time Markov model that could be generalized to handle covariate analysis and the fitting of certain non-homogeneous models. This work overcame the limitations of Bartholomew (1968), Spilerman and Singer (1976) and Wasserman (1980) methodologies, by developing a new algorithm that provided a very efficient way of obtaining maximum likelihood estimates. Also, Muenz and Rubinstein (1985) developed a Markov model for covariates dependence of binary sequences, where the transitions probabilities were estimated through two logistic regressions that depended on a set of covariates. Essentially, Muenz and Rubinstein (1985) modeled a non-homogeneous Markov chain through logistic regression, considering only two states. Islam et al. (2004) developed an extension of this model considering three states, and Islam and Chowdhury (2006) generalized this approach for HOMC. Additionally, Azzalini (1994) proposed a model to study the influence of time-dependent covariates on the marginal distribution of

a binary response in serially correlated binary data, where Markov chains are expressed in terms of transitional probabilities. Jackson (2011) proposed a Markov model for panel data, which allowed for the transitions intensities to vary between individuals or constant time-dependent covariates. Specifically, this work allowed to account for different intensities throughout transitions of states and include individual-specific covariates. The time-inhomogeneos model proposed is restricted to piecewise-constant intensities. The implementation of this work is available in the package **msm**. More recently, Bolano (2020) proposed an MTD-based approach to handle categorical covariates, that considers each covariate separately and combines the effects of the lags of the MTD and the covariates employing a mixture model. Specifically, the model is given by:

$$P(X_t = k \mid X_{t-1} = i, C_1 = c_1, \ldots, C_l = c_l) \approx \theta_0 a_{ik} + \sum_{h=1}^{l} \theta_h d_{c_h k} \tag{7}$$

where a_{ik} is the transition probability from state i to state k, as in a conventional Markov chains and $d_{c_h k}$ is the probability of observing the states k given the modality c_h of the covariate h. Lastly, $\theta_0, \ldots, \theta_l$ are the weights of the explanatory elements of the model.

According to the literature presented, several researchers have proposed methodologies or generalizations to include covariates in Markov chain models. Primarily for social sciences and health applications, where the transition probabilities were generally modeled through logistic regression. However, there has been an increased focus on categorical covariates, opposing continuous covariates and a lack of approaches to multivariate Markov chain models. Thus, with this work, we aim to tackle this research gap.

4 Multivariate Markov chains with covariates

4.1 Theoretical model

In this work, a new generalization of Ching et al. (2002) MMC model is presented: the GMMC model, that is, we will consider exogeneous or pre-determined covariates in the σ - algebra generated by the available information until $t-1$ ($\mathcal{F}_{t-1}$). These variables can be deterministic or stochastic and do not necessarily need to be reported at time t. Broadly, the model is given by:

$$P(S_{jt} = k|\mathcal{F}_{t-1}) = P(S_{jt} = k|S_{1t-1} = i_1, S_{2t-1} = i_2, \ldots, S_{st-1} = i_s, \boldsymbol{x}_t) \tag{8}$$

We can specify this model as proposed by Ching et al. (2002) with Raftery's notation:

$$P(S_{jt} = i_0|S_{1t-1} = i_1, \ldots, S_{st-1} = i_s, \boldsymbol{x}_t) \equiv \\ \lambda_{j1} P(S_{jt} = i_0|S_{1t-1} = i_1, \boldsymbol{x}_t) + \cdots + \lambda_{js} P(S_{jt} = i_0|S_{st-1} = i_s, \boldsymbol{x}_t) \tag{9}$$

subject to the usual constraints.

4.2 Estimation and inference

This proposed model is estimated through MLE, similar to the standard MTD model. The log-likelihood is given by:

$$LL = \sum_{t=1}^{n} log P(S_{jt} = i_0|S_{1t-1} = i_1, \ldots, S_{st-1} = i_s, \boldsymbol{x}_t) \tag{10}$$

Additionally, the probabilities can be estimated through an multinomial logit model. The proof for consistency and asymptotic distribution is available in the Supplementary Material section.

4.3 Monte Carlo simulation study

A Monte Carlo simulation study was designed to evaluate the dimension and power of the test parameters of the proposed model. The R statistical environment was used for all computations. This simulation study was comprised of two parts.

Table 1: Power and dimension of test assessment

	Hypothesis	Test
Power	$H_0 : \lambda_{11} = 0$	$\frac{\widehat{\lambda}_{11}^2}{se(\widehat{\lambda}_{11})^2} \sim \chi^2_{(1)}$
	$H_0 : \lambda_{12} = 1$	$\frac{(\widehat{\lambda}_{12}-1)^2}{se(\widehat{\lambda}_{12})^2} \sim \chi^2_{(1)}$
Dimension	$H_0 : \lambda_{11} = 1$	$\frac{(\widehat{\lambda}_{11}-1)^2}{se(\widehat{\lambda}_{11})^2} \sim \chi^2_{(1)}$
	$H_0 : \lambda_{12} = 0$	$\frac{\widehat{\lambda}_{12}^2}{se(\widehat{\lambda}_{12})^2} \sim \chi^2_{(1)}$

Part I: Detect a non-homogeneous Markov chain

First, we considered two sequences with two and three states. The main goal was to assess if the model detected the presence of a non-homogeneous Markov chain correctly and if the estimate of the parameter would correspond to the expected. So, given two sequences, one generated through a non-homogeneous Markov chain and the other generated through a homogeneous Markov chain, it would be expected that the parameter associated with the transition probabilities of the first sequence would be one and the parameter associated with the transition probabilities of the second sequence would be zero. With this in mind, the transitions probabilities of the first sequence were estimated through a logistic regression, where parameters of this regression were randomly generated in R, and the second sequence was generated through a first-order Markov chain. Hence, for both states cases considered, it was expected that the estimated regression would be:

$$P(S_{1t} = i_0 | S_{1t-1} = i_1, S_{2t-1} = i_2, \boldsymbol{x}_{t-1}) = \\ 1 \times P(S_{1t} = i_0 | S_{1t-1} = i_1, \boldsymbol{x}_{t-1}) + 0 \times P(S_{1t} = i_0 | S_{2t-1} = i_2, \boldsymbol{x}_{t-1}) \quad (11)$$

To assess the test power and dimension, we used the Wald test with the following hypothesis:

The simulation procedure was performed as follows:

1. Generate the values of the coefficients for the probability transition matrix of series S_{1t} randomly;
2. Generate the probability transition matrix of series S_{2t} randomly;
3. Set the initial value of S_{2t} to 1 and simulate the following from the defined probability transition matrix;
4. In each iteration (of 1000 repetitions),
    - Generate $X_t \sim N(2, 25)$;
    - Generate the time-varying probabilities of series S_{1t} through the values of the fixed coefficients and the lagged variable x_t;
    - Set the initial values of the series S_{1t} as 1;
    - For each period t, simulate the next state of S_{1t} from the probabilities simulated for that moment;
    - Estimate the model through the function `mmcx`;
    - Calculate the Wald test and add to the counter if it is rejected.

Considering two states, the test dimension was at 5.7% with a sample size of 100 observations, sightly increased with 500 observations, and returned to the expected values in 1000 and 5000 observations. For a sample size of 100, 500, and 1000 observations, we have low test power. So, when considering two states, the sample must have at least 5000 observations, or, if that is not possible, consider a higher significance level when testing for individual significance.

Considering three states, the test dimension was 9.7% for a sample size of 100 observations, 0.2% for a sample size of 500 observations, and 0.3% for a sample size of 1000. Regarding the test power, we see similar behavior, for a sample of 100 observations, the test power was 90.5%, and from a sample of 500 observations, we reach a test power of 100%. Thus, when considering three states, one may consider a sample of 500 observations without compromising the test power and dimension.

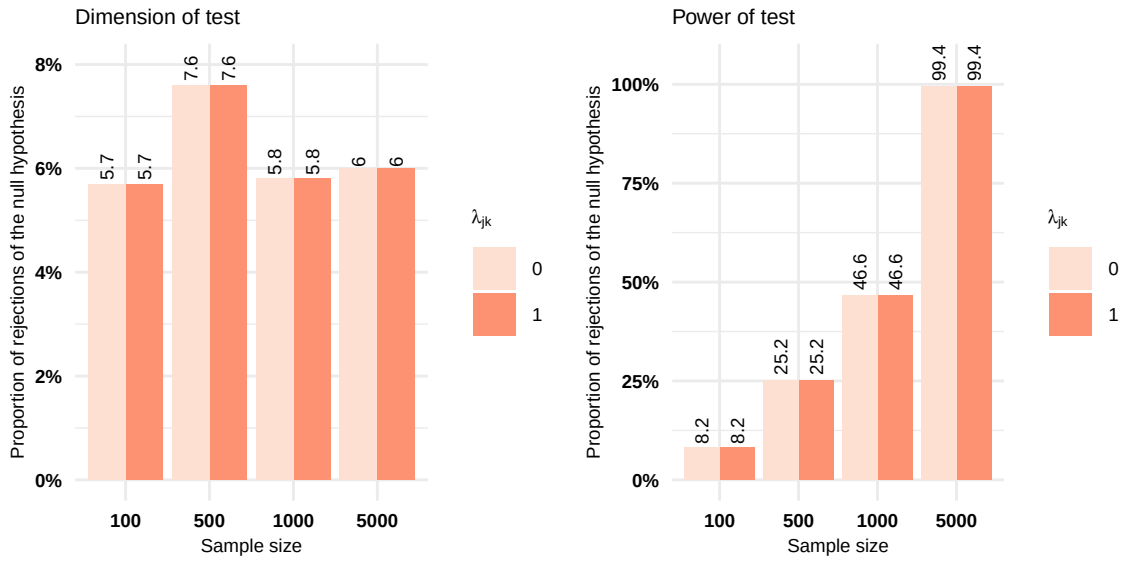


Figure 1: Simulation study results for two-states, displaying the proportion of rejections of the null hypothesis for two parameter values. Dimension of test remains stable regardless sample size. Power of test increases with sample size. The proposed model detects the presence of non-homogenenous Markov Chain.

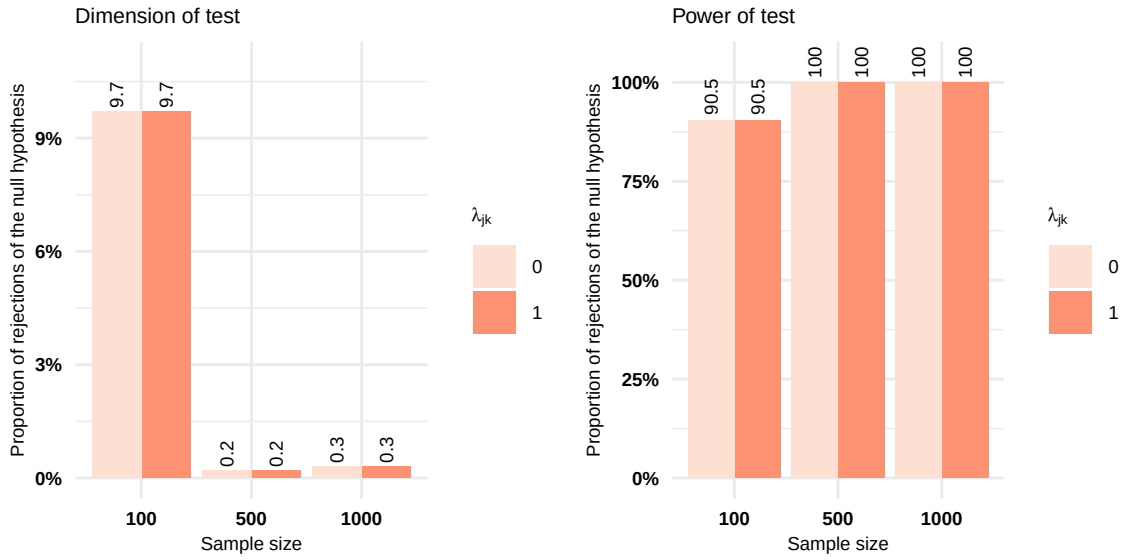


Figure 2: Simulation study results for three-states, displaying the proportion of rejections of the null hypothesis for two parameter values. Dimension of test decreases as sample size increases. Power of test is stable regardless of sample size. The proposed model detects the presence of non-homogenenous Markov Chain.

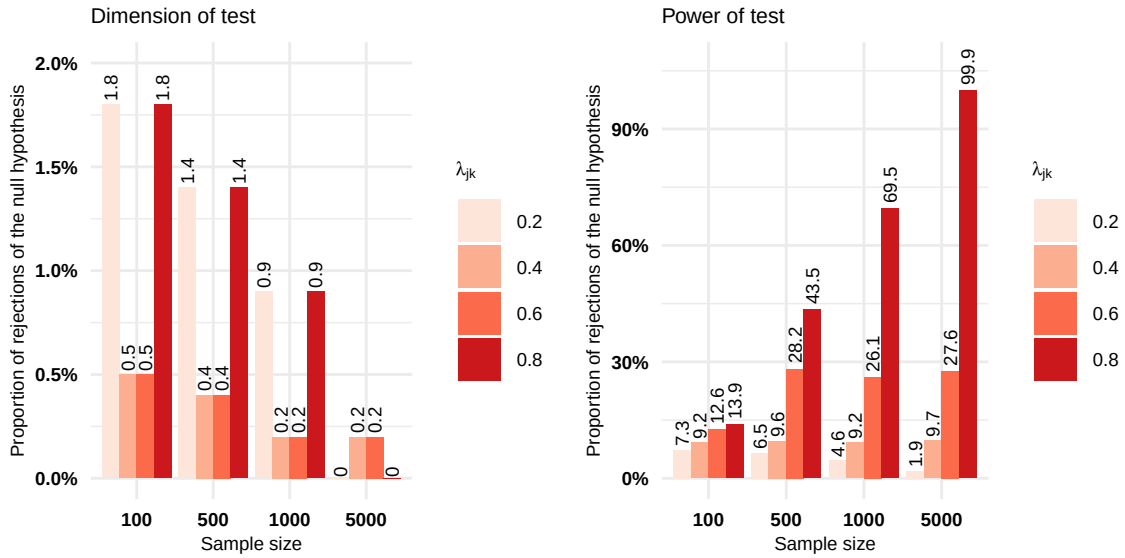


Figure 3: Simulation study results for persistent states on low values of the parameters (case 1), displaying the proportion of rejections of the null hypothesis for four parameter values. Dimension decreases as sample size increases. Power of test increases with sample size. The proposed model has low power of test when low parameter values are associated with persistent states.

Part II: Detecting Parameters Assigned Values

Secondly, we performed a simulation study where we considered two non-homogeneous Markov chain with two states. Here, the main goal was to assess if the model correctly detected the parameters assigned. So, in this case, we started by generating the terms of the model proposed. These terms were estimated through logistic regression, and the parameters of this regression were randomly generated in R. Similarly to Part I, we considered a Wald test to assess the power and dimension of the test. The simulation procedure was performed as follows:

1. Generate the values of the coefficients to calculate the probability transition matrices randomly;
2. In each iteration (of 1000 repetitions),
   - Generate $\{x_t\} \sim N(2, 25)$;
   - Generate the probabilities $P\left(S_{jt}|S_{st-1}, x_{t-1}\right)$, with $j = 1, 2$ and $s = 1, 2$.
   - Set the initial values of the series S_{1t} and S_{2t} as 1;
   - For each period t, calculate the probabilities $P\left(S_{1t}|S_{1t-1}, S_{2t-1}, x_{t-1}\right)$ and $P\left(S_{2t}|S_{1t-1}, S_{2t-1}, x_{t-1}\right)$ through the assigned values of the λ's. Considering the calculated probabilities, simulate the next state for each series, S_{1t} and S_{2t}.
   - Estimate the model through the function `mmcx`;
   - Calculate the Wald test and add to the counter if it is rejected.

The probabilities $P\left(S_{1t}|S_{1t-1}, x_{t-1}\right)$ and $P\left(S_{1t}|S_{2t-1}, x_{t-1}\right)$ presented some differences regarding its values' distributions. Specifically, $P\left(S_{1t}|S_{1t-1}, x_{t-1}\right)$ had more extreme probabilities values, with the minimum value being close to 0 and the maximum value being close to 1. And, the probabilities $P\left(S_{1t}|S_{2t-1}, x_{t-1}\right)$ had more moderate values, with the minimum value being, on average, 0.3 and the maximum value, 0.7. When the probabilities have values close to 1, one says that the states/regimes are persistent. We calculated the power and dimension of test for each value of λ when the estimated probabilities are moderate and when they are extreme. Hence, considering equation 1:

$$P\left(S_{1t} = i_0|S_{1t-1} = i_1, \ldots, S_{2t-1} = i_2, \boldsymbol{x}_{t-1}\right) = \\ \lambda_{11} P\left(S_{1t} = i_0|S_{1t-1} = i_1, \boldsymbol{x}_{t-1}\right) + \lambda_{12} P\left(S_{1t} = i_0|S_{2t-1} = i_s, \boldsymbol{x}_{t-1}\right) \quad (12)$$

The parameter λ_{11} will be associated with more extreme probabilities and λ_{12} will be associated with more moderate probabilities.

When the states are persistent and the parameter's value is low (i.e., 0.2 and 0.4), we have low test power. By increasing this value, the power of test increases as well. When the states are not persistent, we do not have a clear pattern regarding the power of test, for a value of the parameter of 0.2, the

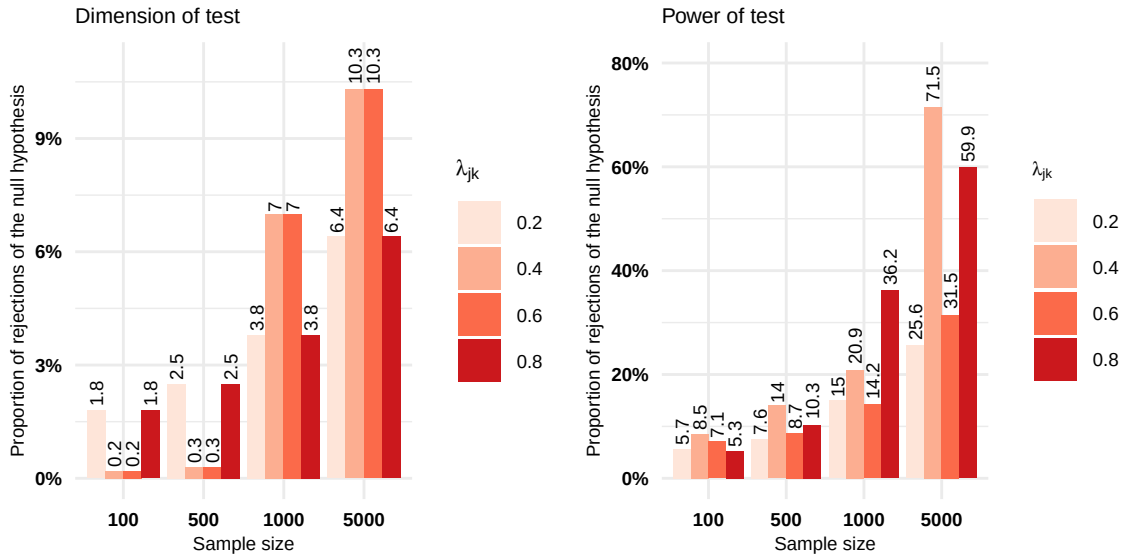


Figure 4: Simulation study results for persistent states on high values of the parameters (case 2), displaying the proportion of rejections of the null hypothesis for four parameter values. Dimension and power of test increase as sample size increases. The results point towards a low test power in this setting.

power of test is still low (although not as low as the first scenario), increases when we have a value of 0.4, decreases when the value is 0.6 and increases again when the value is 0.8. Overall, the estimated standard errors seem high, leading to low test power. Regarding the test dimension, when we have a higher weight associated with the non-persistent states, the test dimension converges to 0. However, when this weight is associated with the persistent states, the test dimension increases with the sample size, reaching a value of 10% in some cases. Hence, one must use a 10% significance level to perform statistical inference on the parameters in this situation.

4.4 Software implementation

Regarding the software implementation for each function, for the `multimtd` function the estimation method was presented in Berchtold (2001) applied to the multivariate case. For `multimtd_probit`, a package for numerical maximization of the log-likelihood, **maxLik** (Henningsen and Toomet, 2011), was used. This package performs Maximum Likelihood estimation through different optimization methods that the user can choose. The optimization methods available are Newton-Raphson, Broyden - Fletcher - Goldfarb - Shanno, BFGS al- algorithm, Berndt - Hall - Hall - Hausman, Simulated AN- Nealing, Conjugate Gradients, and Nelder-Mead. Finally, for the `mmcx` function, a different approach was used. Unlike the MTD- Probit, the model proposed has equality and inequality restrictions in the parameters. The **maxLik** (Henningsen and Toomet, 2011) package only allows one type of restriction for each Maximum Likelihood estimation, so it was not possible to use this package to estimate the proposed model with exogenous variables. Hence, the algorithm used was the Augmented Lagrangian method, available in the **alabama** (Varadhan, 2015) package through the function `auglag`. This estimation method for the proposed model is not very common, however, it has been applied to Markov chain models (Rajarshi, 2013). The GMMC model's probabilities were estimated through a Multinomial Logit using `rmultinom` of the **nnet** package (Venables and Ripley, 2002).

Additionally, the hessian matrices were also computed, which allowed performing statistical inference. The `maxLik` and `auglag` compute the Hessian matrices with the estimates. For the function `multimtd`, since the optimization procedure of Berchtold (2001) was used, the hessian was computed through the second partial derivatives. The function `multi.mtd` requires the following elements:

- `y`, a matrix of the categorical data sequences.
- `deltaStop`, the delta below which the optimization phases of the parameters stop.
- `is_constrained`, flag indicating whether the function will consider the usual set of constraints (usual set: *TRUE*, new set of constraints: *FALSE*).
- `delta`, the amount of change to increase/decrease in the parameters for each iteration of the optimization algorithm.

The last three arguments concern the optimization procedure. For more details see Berchtold (2001). Considering two vectors of two categorical data sequences, `s1` and `s2`, to estimate the model and obtain the results:

```
multi.mtd(y=cbind(s1,s2), deltaStop=0.0001, is_constrained=TRUE, delta=0.1)
```

The function `multi.mtd_probit` requires the following arguments:

- `y`, a matrix of the categorical data sequences.
- `initial`, a vector of the initial values of the parameters.
- `nummethod`, the numerical maximization method, currently either "NR" (for Newton-Raphson), "BFGS" (for Broyden-Fletcher-Goldfarb-Shanno), "BFGSR" (for the BFGS algorithm implemented in R), "BHHH" (for Berndt-Hall-Hall-Hausman), "SANN" (for Simulated ANNealing), "CG" (for Conjugate Gradients), or "NM" (for Nelder-Mead). Lower-case letters (such as "nr" for Newton-Raphson) are allowed. The default method is "BFGS". For more details see **maxLik** (Henningsen and Toomet, 2011) package.

Considering two vectors of two categorical data sequences, `s1` and `s2` again, to estimate the model an obtain the results with BFGS maximization method:

```
multi.mtd_probit(y = cbind(s1,s2), initial=c(1,1,1), nummethod='bfgs')
```

Finally, the function `mmcx` requires the following elements:

- `y`, a matrix of categorical data sequences.
- `x`, a matrix of covariates (exogeneous variables).
- `initial`, a vector of the initial values of the parameters.

Considering two vectors of two categorical data sequences, `s1` and `s2`, and a vector of an exogeneous variables, `x`, to estimate the model and obtain the results:

```
mmcx(y = cbind(s1,s2), x = cbind(x), initial=c(1,1))
```

These functions return a list with the parameter estimates, standard errors, z-statistics, p- values, and the log-likelihood function value for each equation.

The package offers an additional function that allows to obtain the transition probability matrices of `mmcx` considering a specific value of `x` defined by the user. The function is `MMC_tpm` and requires the following elements:

- `s`, a matrix of categorical data sequences.
- `x`, a matrix of covariates (exogeneous variables).
- `value`, a single value of `x`, to condition the probability transition matrices.
- `result`, a list returned by the function `mmcx` containing the model's estimates.

Considering two vectors of two categorical data sequences, `s1` and `s2`, a vector of an exogeneous variables, `x` and `res` the list returned by the function `mmcx`, to obtain the transition probability matrices:

```
MMC_tpm(s = cbind(s1,s2), x = cbind(x), value = max(x), result = res)
```

The function returns an array containing the probability transition matrices, conditioned on a specific value of `x`, for each equation.

5 Illustration

Markov chain models are used in interdisciplinary areas, such as economics, business, biology, and engineering, with applications to predict long-term behavior from traffic flow to stock market movements, among others. Modeling and predicting stock markets returns is particularly relevant for investors and policy makers. Since the stock market is a volatile environment, and the returns are difficult to predict, estimating the set of probabilities that describe these movements, might provide relevant input. Additionally, incorporating the effect of key macroeconomic variables could provide a more accurate picture of this specific environment.

The following empirical illustration aims to model stock returns of two indexes as a function of the interest rate spread, specifically the 10-Year Treasury Constant Maturity Minus 3-Month Treasury Constant Maturity.

Table 2: Summary statistics of *stockreturns* dataset

Variable	Minimum	1^{st} Quantile	Median	Mean	3^{rd} Quantile	Maximum
$spread_t$	-0.52	0.92	1.54	1.454	2.03	2.97
$r_{t;SP500}$	-12.765	-0.32	0.07	0.054	0.518	8.968
$r_{t;DJIA}$	-13.842	-0.327	0.071	0.046	0.508	10.764

The interest rate spread is a key macroeconomic variable and provides valuable information regarding the economy state. Specifically, it has been used to forecast recessions as in Estrella and Mishkin (1996), Dombrosky and Haubrich (1996), Chauvet and Senyuz (2016), Tian and Shen (2019) and McMillan (2021). Generically, short-term yields are lower than long-term yields when the economy is in expansion. On the other hand, short-term yields are higher than long-term yields when the economy is in recession. The difference between these yields (or, more specifically, the yield curve's slope) can be used to forecast the state of the economy. Hence, this indicator might provide relevant input for investors.

We considered the 5-week-day daily stock returns ($r_t = 100 \times \log(P_t/P_{t-1})$, where P_t is the adjusted close price) of two indexes, S&P500 and DJIA, from November 11^{th} 2011 to September 1^{st} 2021 (2581 observations). Additionally, we considered the interest rate spread ($spread_t$), the 10-Year Treasury Constant Maturity Minus 3-Month Treasury Constant Maturity. The data was retrieved from FRED. Below, we have the descriptive statistics of these variables.

Moreover, to apply the model proposed, it is necessary to have a categorical time series, thus we applied the following procedure:

$$S_{st} = \begin{cases} 1, r_t \leq \widehat{q}_{s;0.25} \\ 2, \widehat{q}_{s;0.25} < r_t < \widehat{q}_{s;0.75} \\ 3, r_t \geq \widehat{q}_{s;0.75} \end{cases}$$

where $\widehat{q}_{s;\alpha}$ is the estimated quantile of order α of the marginal distribution of r_t. Considering this illustration and the model proposed, we will have two equations:

$$P(S_{sp500,t}|S_{sp500,t-1}, S_{djia,t-1}, spread_{t-1}) = \lambda_{11}P(S_{sp500,t}|S_{sp500,t-1}, spread_{t-1}) + \lambda_{12}P(S_{sp500,t}|S_{djia,t-1}, spread_{t-1}) \quad (13)$$

$$P(S_{djia,t}|S_{sp500,t-1}, S_{djia,t-1}, spread_{t-1}) = \lambda_{21}P(S_{djia,t}|S_{sp500,t-1}, spread_{t-1}) + \lambda_{22}P(S_{djia,t}|S_{djia,t-1}, spread_{t-1}) \quad (14)$$

In Figures 5 to 8 generate through **ggplot2** (Wickham, 2016) and **gridExtra** (Auguie, 2017), we have the smoothed conditional probabilities of both series, depending on $spread_{t-1}$. The number of observations is high, and the probabilities varied abruptly in a small time frame, making the plots hard to read. To simplify, a moving average model (from **pracma** (Borchers, 2022)) of order 5, due to the frequency of the data, was adjusted to these probabilities to illustrate how they evolve throughout time. These plots represent the probabilities associated with the parameters of the general model proposed, showcasing how these vary throughout time and the main of advantage of this generalization. Instead of having fixed matrices of transition probabilities, we allow for these to vary throughout time, depending on the values of $spread_{t-1}$. Specifically, Figures 5 and 6 correspond to the non-homogeneous Markov chain to build the SP&500's equation and Figures 7 and Figures 8 correspond to the non-homogeneous Markov chain to build DJIA's equation. We see a similar behavior within each series regardless of whether it depends on the previous states of S_{1t} or S_{2t}. Additionally, the scales of the graphs are small, indicating that these probabilities vary around the same set of values.

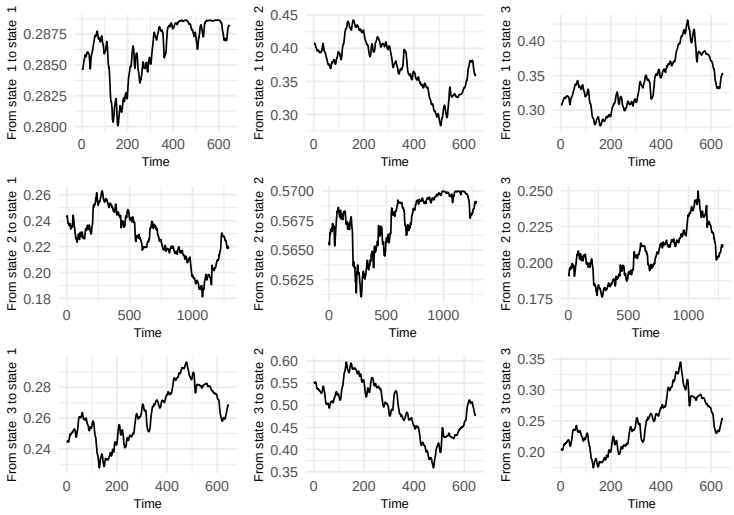


Figure 5: Estimated conditional probabilities of series 1 (SP500) depending on $spread_{t-1}$ and on series 1 (SP500) previous state: $P(S_{sp500,t}|S_{sp500,t-1}, spread_{t-1})$. This figure shows the estimated non-homogeneous Markov chain from which the realized probabilites will be extracted to maximize the log-likelihood function.

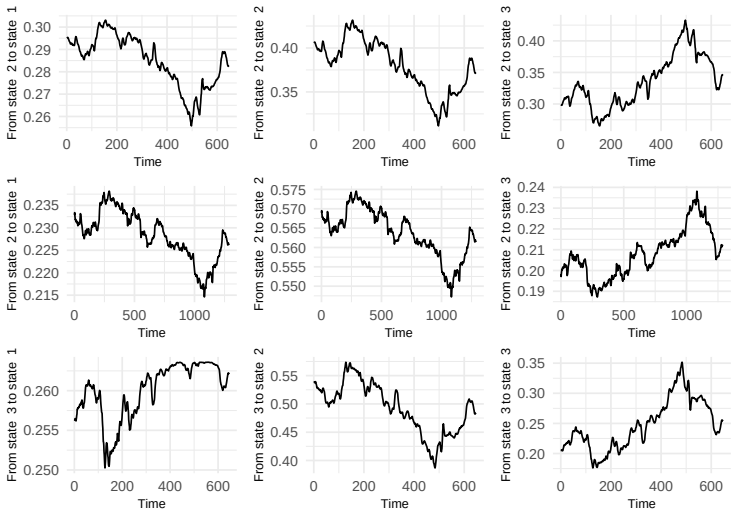


Figure 6: Estimated conditional probabilities of series 1 (SP500) depending on $spread_{t-1}$ and on series 2 (DJIA) previous state: $P(S_{sp500,t}|S_{djia,t-1}, spread_{t-1})$. This figure shows the estimated non-homogeneous Markov chain from which the realized probabilites will be extracted to maximize the log-likelihood function.

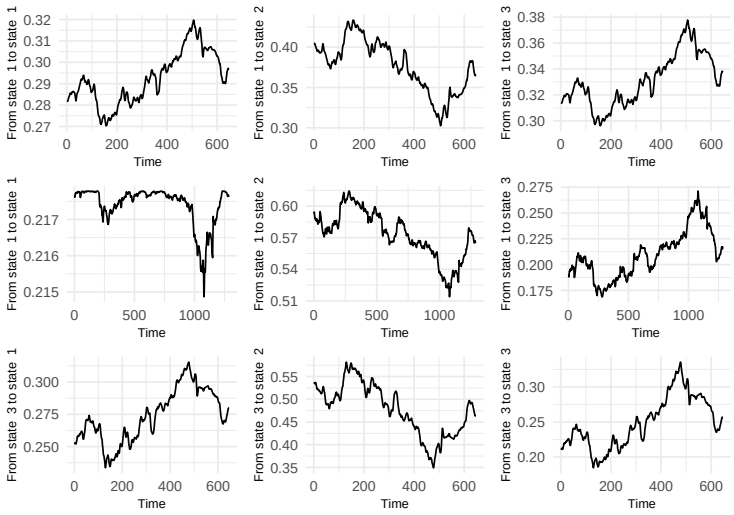


Figure 7: Estimated conditional probabilities of series 2 (DJIA) depending on $spread_{t-1}$ and on series 1 (SP500) previous state: $P(S_{djia,t}|S_{sp500,t-1}, spread_{t-1})$. This figure shows the estimated non-homogeneous Markov chain from which the realized probabilites will be extracted to maximize the log-likelihood function.

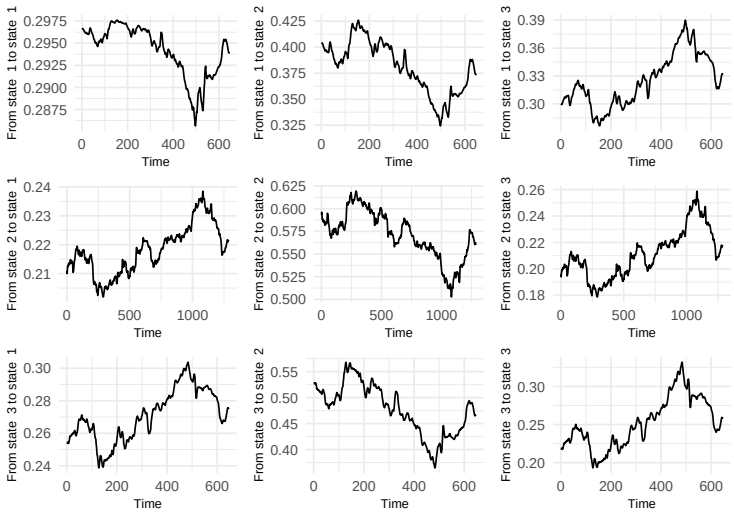


Figure 8: Estimated conditional probabilities of series 2 (DJIA) depending on $spread_{t-1}$ and on series 2 (DJIA) previous state: $P(S_{djia,t}|S_{djia,t-1}, spread_{t-1})$. This figure shows the estimated non-homogeneous Markov chain from which the realized probabilites will be extracted to maximize the log-likelihood function.

The model can be estimated through the `mmcx` function:

```
attach(stockreturns)
res <- mmcx(cbind(sp500, djia), spread_1, initial=c(1,1))

#> ------------------------------------------
#> Equation 1
#>   Estimate Std. Error t value Pr(>|t|)
#> 1 0.685660   0.171346   4.002    0.000 ***
#> 2 0.314340   0.171346   1.835    0.067 *
#>
#> Log-Likelihood: -2636.355
#> ------------------------------------------
#> ------------------------------------------
#> Equation 2
#>   Estimate Std. Error t value Pr(>|t|)
#> 1 0.629993   0.176383   3.572    0.000 ***
#> 2 0.370007   0.176383   2.098    0.036 **
#>
#> Log-Likelihood: -2636.622
#> ------------------------------------------
```

Considering the first equation, the effect of the probabilities depending on S&P500's previous state and the interest rate spread has a higher weight on the overall probability. Also, this estimate is highly significant, presenting a p-value close to zero. The effect of DJIA's previous state in S&P500 is lower but it is also significant for a 10% significance level. In the second equation, the effect of S&P500's previous state is higher than DJIA's and both estimates are highly significant.

One of the advantages of this approach is the possibility to assess the transition probabilities for specific values of x_t, in this case, the interest rate spread. For both series, we calculated the transition probabilities for this variable's minimum and maximum value in the sample, which are -0.52 and 2.97, respectively. To obtain the probability transition matrices for these two cases, the code is the following:

```
tpm_max <- MMC_tpm(cbind(sp500, djia), spread_1,
                   value = max(spread_1), result = res)

tpm_min <- MMC_tpm(cbind(sp500, djia), spread_1,
                   value = min(spread_1), result = res)

library(markovchain)
plot(new('markovchain', transitionMatrix = tpm_max[,,1])) # Generate figure 9
plot(new('markovchain', transitionMatrix = tpm_min[,,1])) # Generate figure 10
plot(new('markovchain', transitionMatrix = tpm_max[,,2])) # Generate figure 11
plot(new('markovchain', transitionMatrix = tpm_min[,,2])) # Generate figure 12
```

In Figures 10 and 9, we have the transition probabilities network for S&P500, corresponding to the minimum and maximum value of the spread. The most noticeable difference between these two networks is regarding the transition probability from the second state to the third state. For the maximum value of $spread_{t-1}$, the transition probability from the second state to the third state is 0.6. So, when the economy is strong, one might expect to have higher returns, when $t-1$ was in the second state. However, this scenario shifts when considering the minimum value of $spread_{t-1}$. The probability of obtaining higher returns, that is, being in state three, becomes almost evenly distributed, regardless of the state in $t-1$. This indicates the instability of the stock market, when the economy is weaker. Another difference in these networks, is regarding the transition probability from the third state to the first state. For the maximum value of $spread_{t-1}$, this probability is 0.27 and for the minimum value increases to 0.44. This is also expected, since when the economy is weaker, the probability of having lower returns is greater.

Considering the second equation (Figures 11 and 12), corresponding to the DJIA's returns, we see a similar behaviour as in S&P500's networks. The transition probability from the second state to the third state is higher for the maximum value of $spread_{t-1}$ and the transition probability from the third state to the first state is higher when we consider the minimum value of $spread_{t-1}$. Although, the difference of this last probability between the minimum and maximum value of $spread_{t-1}$ is not as big as in S&P500. Overall, the rest of the probabilities structure, remains the same.

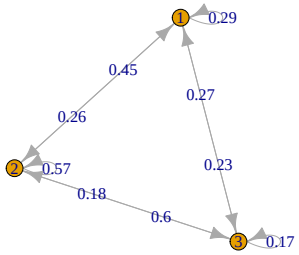


Figure 9: Graphical representation of the transition probability matrix of Series 1: SP500 for the maximum value of spread$_{t-1}$. The highest probability of 0.6 refers to the transition from state 2 to state 3.

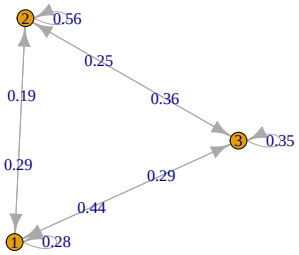


Figure 10: Graphical representation of the transition probability matrix of Series 1: SP500 for the minimum value of spread$_{t-1}$. The highest probability of 0.56 refers to the transition from state 2 to state 2.

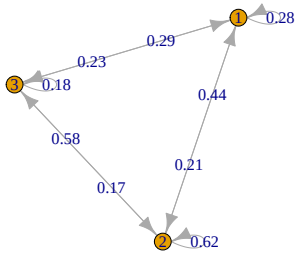


Figure 11: Graphical representation of the transition probability matrix of Series 2: DJIA for the maximum value of spread$_{t-1}$. The probability of 0.58 refers to the transition from state 2 to state 3.

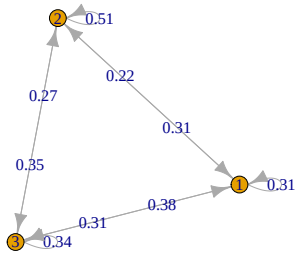


Figure 12: Graphical representation of the transition probability matrix of Series 2: DJIA for the minimum value of spread$_{t-1}$. The highest probability of 0.51 refers to the transition from state 2 to state 2.

6 Conclusions, limitations and further research

Several proposals for including of exogenous variables in MMC models have been presented. The main limitations were associated with the high complexity of the models to be developed and estimated. Additionally, most models considered only categorical exogenous variables, existing a lack of focus on continuous exogenous variables. This work proposes a new approach to include continuous exogenous variables in Ching et al. (2002) model for multivariate Markov chain. This is relevant because it allows studying the effect of previous series and exogenous variables on the transition probabilities. The model is based on Ching et al. (2002) MMC model but considers non-homogeneous Markov chains. Thus, the probabilities that compose the model are dependent on exogenous variables. These probabilities are estimated as a usual non-homogeneous Markov chain through a multinomial logit model. The model parameters are then estimated through MLE, as well as the standard errors. We developed a package with the estimation function of the model proposed. In this, we considered the Augmented Lagrangian optimization method for estimating the parameters through MLE. Additionally, we designed a Monte Carlo simulation study to assess this model's test power and dimension. The results showed that the model detected a non-homogeneous Markov chain. Moreover, an empirical illustration demonstrated the relevance of this new model by estimating the probability transition matrix for different exogenous variable values. Ignoring the effect of exogenous variables in MMC means that we would not detect the probabilities' changes according to the covariates' values. In this setting, one would have a limited view of the studied process. Hence, this approach allows us to understand how a specific variable influences a specific process. The main contributions of this work are the development of a package with functions for multivariate Markov chains, addressing the statistical inference in these models and the inclusion of covariates. The limitations are related to the implementation in R, specifically the optimization algorithm applied is not common for MMC models, in that sense, it would be beneficial to study new approaches to optimizing the maximum likelihood function as further research. Additionally, extending this generalization to the MTD-probit model proposed by Nicolau (2014) would also be relevant, which removes the constraints of the model's parameters and allows the model to detect negative effects.

References

S. Adke and S. Deshmukh. Limit Distribution of a High Order Markov Chain. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(1):105–108, 1988. URL https://www.jstor.org/stable/2345812. [p97]

B. Auguie. *gridExtra: Miscellaneous Functions for "Grid" Graphics*, 2017. URL https://CRAN.R-project.org/package=gridExtra. R package version 2.3. [p105]

A. Azzalini. Logistic regression for autocorrelated data with application to repeated measures. *Biometrika*, 81(4):767–775, 1994. ISSN 00063444. doi: 10.1093/biomet/81.4.767. [p98]

J. Bartholomew. Stochastic Models for Social Processes. *The Australian and New Zealand Journal of Sociology*, 4(2):171–172, 1968. doi: https://doi.org/10.1177/144078336800400215. [p98]

A. Berchtold. Autoregressive Modelling of Markov Chains. *Proc. 10th International Workshop on Statistical Modelling*, 104:19–26, 1995. doi: 10.1007/978-1-4612-0789-4_3. [p96]

A. Berchtold. Modélisation autorégressive des chaines de Markov : utilisation d'une matrice différente pour chaque retard. *Revue de Statistique Appliquée*, 44(3):5–25, 1996. URL http://www.numdam.org/item/RSA_1996__44_3_5_0/. [p96]

A. Berchtold. Estimation in the mixture transition distribution model. *Journal of Time Series Analysis*, 22(4):379–397, 2001. doi: https://doi.org/10.1111/1467-9892.00231. [p103, 104]

A. Berchtold. Mixture transition distribution (MTD) modeling of heteroscedastic time series. *Computational Statistics and Data Analysis*, 41(3-4):399–411, 2003. ISSN 01679473. doi: 10.1016/S0167-9473(02)00191-3. [p97]

A. Berchtold, O. Maitre, and K. Emery. Optimization of the mixture transition distribution model using the march package for R. *Symmetry*, 12(12):1–14, 2020. ISSN 20738994. doi: 10.3390/sym12122031. [p96]

D. Bolano. Handling covariates in markovian models with a mixture transition distribution based approach. *Symmetry*, 12(4), 2020. ISSN 20738994. doi: 10.3390/SYM12040558. [p99]

H. W. Borchers. *pracma: Practical Numerical Math Functions*, 2022. URL https://CRAN.R-project.org/package=pracma. R package version 2.4.2. [p105]

M. Chauvet and Z. Senyuz. A dynamic factor model of the yield curve components as a predictor of the economy. *International Journal of Forecasting*, 32(2):324–343, 2016. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2015.05.007. [p105]

D. G. Chen and Y. L. Lio. A Novel Estimation Approach for Mixture Transition Distribution Model in High-Order Markov Chains. *Communications in Statistics - Simulation and Computation*, 38(5): 990–1003, 2009. doi: 10.1080/03610910802715009. [p97]

W. K. Ching and M. K. Ng. *Markov Chains: Models, Algorithms and Applications*. Springer, 2006. ISBN 9780387293370. doi: 10.1007/0-387-29337-X. [p97]

W. K. Ching, E. S. Fung, and M. K. Ng. A multivariate markov chain model for categorical data sequences and its applications in demand predictions. *IMA Journal of Management Mathematics*, 13 (3):187–199, 2002. doi: 10.1093/imaman/13.3.187. [p96, 97, 98, 99, 110]

W. K. Ching, E. S. Fung, and M. K. Ng. A higher-order markov model for the newsboy's problem. *The Journal of the Operational Research Society*, 54(3):291–298, 2003. [p97]

W. K. Ching, M. K. Ng, and E. S. Fung. Higher-order multivariate Markov chains and their applications. *Linear Algebra and its Applications*, 428(2-3):492–507, 2008. doi: 10.1016/j.laa.2007.05.021. [p97, 98]

B. Damásio. *Multivariate Markov Chains - Estimation, Inference and Forecast. A New Approach: What If We Use Them As Stochastic Covariates?* Master dissertation, Universidade de Lisboa, Instituto Superior de Economia e Gestão, 2013. URL http://hdl.handle.net/10400.5/6397. [p97, 98]

B. Damásio. *Essays on Econometrics: Multivariate Markov Chains*. PhD dissertation, Universidade de Lisboa, Instituto Superior de Economia e Gestão, 2018. URL https://www.repository.utl.pt/bitstream/10400.5/18128/1/TD-BD-2019.pdf. [p97]

B. Damásio and S. Mendonça. Modelling insurgent-incumbent dynamics: Vector autoregressions, multivariate Markov chains, and the nature of technological competition. *Applied Economics Letters*, 26(10):843–849, 2019. doi: 10.1080/13504851.2018.1502863. [p98]

B. Damásio and S. Mendonça. Leader-follower dynamics in real historical time: A markovian test of non-linear causality between sail and steam (co-)development, mimeo, 2020. [p98]

B. Damásio and J. Nicolau. Combining a regression model with a multivariate Markov chain in a forecasting problem. *Statistics & Probability Letters*, 90:108–113, 2014. ISSN 0167-7152. doi: https://doi.org/10.1016/j.spl.2014.03.026. [p97, 98]

B. Damásio and J. Nicolau. Time inhomogeneous multivariate Markov chains : detecting and testing multiple structural breaks occurring at unknown dates. REM Working Papers 0136–2020, Instituto Superior de Economia e Gestão, 2020. URL http://hdl.handle.net/10400.5/20164. [p98]

A. M. Dombrosky and J. Haubrich. Predicting real growth using the yield curve. *Economic Review*, I (Q):26–35, 1996. URL https://EconPapers.repec.org/RePEc:fip:fedcer:y:1996:i:qi:p:26-35. [p105]

A. Estrella and F. S. Mishkin. The yield curve as a predictor of U.S. recessions. *Current Issues in Economics and Finance*, 2(Jun), 1996. URL https://www.newyorkfed.org/research/current_issues/ci2-7.html. [p105]

A. Henningsen and O. Toomet. maxlik: A package for maximum likelihood estimation in R. *Computational Statistics*, 26(3):443–458, 2011. doi: 10.1007/s00180-010-0217-1. URL http://dx.doi.org/10.1007/s00180-010-0217-1. [p103, 104]

M. A. Islam and R. I. Chowdhury. A higher order Markov model for analyzing covariate dependence. *Applied Mathematical Modelling*, 30(6):477–488, 2006. ISSN 0307904X. doi: 10.1016/j.apm.2005.05.006. [p98]

M. A. Islam, S. Arabia, and R. I. Chowdhury. A Three State Markov Model for Analyzing Covariate Dependence. *International Journal of Statistical Sciences*, 3(i):241–249, 2004. URL http://www.ru.ac.bd/stat/wp-content/uploads/sites/25/2019/01/P21.V3s.pdf. [p98]

C. Jackson. Multi-state models for panel data: the msm package for r. *Journal of statistical software*, 38: 1–28, 2011. doi: 10.18637/jss.v038.i0810.18637/jss.v038.i08. [p99]

P. Jacobs and A. Lewis. Discrete Time Series Generated by Mixtures II : Asymptotic Properties. *Journal of the Royal Statistical Society: Series B (Methodological)*, 40(2):222–228, 1978. URL https://www.jstor.org/stable/2984759. [p96]

J. D. Kalbfleisch and J. F. Lawless. The analysis of panel data under a Markov assumption. *Journal of the American Statistical Association*, 80(392):863–871, 1985. ISSN 1537274X. doi: 10.1080/01621459.1985.10478195. [p98]

M. Kijima, K. Komoribayashi, and E. Suzuki. A multivariate Markov model for simulating correlated defaults. *Journal of Risk*, 4, 07 2002. doi: 10.21314/JOR.2002.066. [p97]

N. D. Le, R. D. Martin, and A. Raftery. Modeling Flat Stretches, Brusts, and Outliers in Time Series Using Mixture Transition Distribution Models. *Journal of the American Statistical Association*, 91(436): 1504–1515, 1996. doi: 10.1111/j.2517-6161.1985.tb01383.x. [p97]

J. Logan. A structural model of the higher-order Markov process incorporating reversion effects. *The Journal of Mathematical Sociology*, 8(1):75–89, 1981. doi: 10.1080/0022250X.1981.9989916. [p96]

S. Lèbre and P. Y. Bourguignon. An EM algorithm for estimation in the mixture transition distribution model. *Journal of Statistical Computation and Simulation*, 78(8):713–729, 2008. doi: 10.1080/00949650701266666. [p97]

O. Maitre and K. Emery. *march: Markov Chains*, 2020. URL https://CRAN.R-project.org/package=march. R package version 3.3.2. [p96]

R. D. Martin and A. Raftery. Non-Gaussian State-Space Modeling of Nonstationary Time Series: Comment: Robustness, Computation, and Non-Euclidean Models. *Journal of the American Statistical Association*, 82(400):1044–1050, 1987. doi: 10.2307/2289377. [p97]

D. G. McMillan. Predicting gdp growth with stock and bond markets: Do they contain different information? *International Journal of Finance & Economics*, 26(3):3651–3675, 2021. doi: https://doi.org/10.1002/ijfe.1980. [p105]

F. Mehran. Analysis of Discrete Longitudinal Data: Infinite-Lag Markov Models. In *Statistical Data Analysis and Inference*, pages 533–541. North-Holland, Amsterdam, 1989. ISBN 978-0-444-88029-1. doi: https://doi.org/10.1016/B978-0-444-88029-1.50053-8. [p97]

L. R. Muenz and L. V. Rubinstein. Markov Models for Covariate Dependence of Binary Sequences . *Biometrics*, 41(1):91–101, 1985. URL http://www.jstor.org/stable/2530646. [p98]

W. Nicholson. *DTMCPack: Suite of functions related to discrete-time discrete-state Markov Chains*, 2013. URL https://CRAN.R-project.org/package=DTMCPack. R package version 0.1-2. [p96]

J. Nicolau. A new model for multivariate markov chains. *Scandinavian Journal of Statistics*, 41(4): 1124–1135, 2014. ISSN 14679469. doi: 10.1111/sjos.12087. [p96, 98, 110]

J. Nicolau and F. I. Riedlinger. Estimation and inference in multivariate Markov chains. *Statistical Papers*, 56(4):1163–1173, 2014. ISSN 09325026. doi: 10.1007/s00362-014-0630-6. [p97]

G. Pegram. An Autoregressive Model for Multilag Markov Chains. *Journal of Applied Probability*, 17(2): 350–362, 1980. doi: 10.2307/3213025. [p96]

A. Raftery. A Model for High-Order Markov Chains. *Journal of the Royal Statistical Society: Series B (Methodological)*, 47(3):528–539, 1985. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1985.tb01383.x. [p96]

A. Raftery and S. Tavaré. Estimation and Modelling Repeated Patterns in High Order Markov Chains with the Mixture Transition Distribution Model. *Applied Statistics*, 43(1):179–199, 1994. doi: 10.2307/2986120. [p98]

M. Rajarshi. *Statistical Inference for Discrete Time Stochastic Processes*. SpringerBriefs in Statistics, 2013. ISBN 9783642179792. URL http://www.springer.com/978-81-322-0762-7. [p103]

M. H. Regier. A Two-State Markov Model for Behavioral Change. *Journal of the American Statistical Association*, 63(323):993–999, 1968. doi: 10.1080/01621459.1968.11009325. [p98]

T. K. Siu, W. K. Ching, E. S. Fung, and M. K. Ng. On a multivariate Markov chain model for credit risk measurement. *Quantitative Finance*, 5(6):543–556, 2005. ISSN 14697688. doi: 10.1080/14697680500383714. [p97]

G. A. Spedicato. Discrete time markov chains with r. *The R Journal*, 07 2017. URL https://journal.r-project.org/archive/2017/RJ-2017-036/index.html. R package version 0.6.9.7. [p96]

S. Spilerman and B. Singer. The Representation of Social Processes by Markov Models. *American Journal of Sociology*, 82(1):1–54, 1976. URL https://www.jstor.org/stable/2777460. [p98]

R. Tian and G. Shen. Predictive power of markovian models: Evidence from us recession forecasting. *Journal of Forecasting*, 38(6):525–551, 2019. doi: https://doi.org/10.1002/for.2579. [p105]

R. Varadhan. *alabama: Constrained Nonlinear Optimization*, 2015. URL https://CRAN.R-project.org/package=alabama. R package version 2015.3-1. [p103]

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer, New York, fourth edition, 2002. URL https://www.stats.ox.ac.uk/pub/MASS4/. ISBN 0-387-95457-0. [p103]

C. Wang, T. Z. Huang, and W. K. Ching. A new multivariate Markov chain model for adding a new categorical data sequence. *Mathematical Problems in Engineering*, 2014, 2014. doi: 10.1155/2014/502808. [p97]

S. Wasserman. Analyzing social networks as stochastic processes. *Journal of the American Statistical Association*, 75(370):280–294, 1980. doi: 10.1080/01621459.1980.10477465. [p98]

H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. ISBN 978-3-319-24277-4. URL https://ggplot2.tidyverse.org. [p105]

C. S. Wong and W. K. Li. On a mixture autoregressive conditional heteroscedastic model. *Journal of the American Statistical Association*, 96(455):982–995, 2001. doi: 10.1198/016214501753208645. [p97]

X. Zhang, M. L. King, and R. J. Hyndman. A Bayesian approach to bandwidth selection for multivariate kernel density estimation. *Computational Statistics and Data Analysis*, 50(11):3009–3031, 2006. doi: 10.1016/j.csda.2005.06.019. [p97]

D. M. Zhu and W. K. Ching. A new estimation method for multivariate Markov chain model with application in demand predictions. *Proceedings - 3rd International Conference on Business Intelligence and Financial Engineering, BIFE 2010*, pages 126–130, 2010. doi: 10.1109/BIFE.2010.39. [p97]

Carolina Vasconcelos
NOVA Information Management School (NOVA IMS)
Campus de Campolide, 1070-312 Lisboa, Portugal
cvasconcelos@novaims.unl.pt

Bruno Damásio
NOVA Information Management School (NOVA IMS)
Campus de Campolide, 1070-312 Lisboa, Portugal
bdamasio@novaims.unl.pt

Fitting a Quantile Regression Model for Residual Life with the R Package qris

Kyu Hyun Kim, Sangwook Kang, and Sy Han Chiou

Abstract In survival analysis, regression modeling has traditionally focused on assessing covariate effects on survival times, which is defined as the elapsed time between a baseline and event time. Nevertheless, focusing on residual life can provide a more dynamic assessment of covariate effects, as it offers more updated information at specific time points between the baseline and event occurrence. Statistical methods for fitting quantile regression models have recently been proposed, providing favorable alternatives to modeling the mean of residual lifetimes. Despite these progresses, the lack of computer software that implements these methods remains an obstacle for researchers analyzing data in practice. In this paper, we introduce an R package **qris** (Kim et al., 2022), which implements methods for fitting semiparametric quantile regression models on residual life subject to right censoring. We demonstrate the effectiveness and versatility of this package through comprehensive simulation studies and a real-world data example, showcasing its valuable contributions to survival analysis research.

1 Introduction

In the analysis of time-to-event data, standard statistical inference procedures often focus on quantities based on failure time and its relationship with covariates measured at baseline. However, throughout the follow-up process, inference procedures based on residual life become increasingly intuitive for assessing the survival of subjects and can offer insights into the effectiveness of treatments in prolonging the remaining lifetime. As covariates can substantially change over time and models based solely on baseline covariates have limited potential for long-term prognosis, there is a growing interest in modeling the remaining lifetime of a surviving subject with updated patient information. Many efforts have been made to model the mean residual life including proportional mean residual life models (Maguluri and Zhang, 1994; Oakes and Dasu, 1990, 2003; Chen et al., 2005), additive mean residual life models (Chen and Cheng, 2006; Chen, 2007; Zhang et al., 2010), and proportional scaled mean residual life models (Liu and Ghosh, 2008). Given that failure times are usually right-skewed and heavy-tailed, the mean of the residual life might not be identifiable if the follow-up time is not sufficiently long. For this reason, quantiles, which are robust under skewed distribution, have traditionally been used more frequently as alternative summary measures. For example, the approach on the semiparametric quantile regression model for continuous responses (Koenker and Bassett Jr, 1978) has been extended to uncensored failure time data (Jung, 1996; Portnoy and Koenker, 1997; Wei et al., 2006) and censored failure times data (Ying et al., 1995; Portnoy, 2003; Peng and Huang, 2008; Huang, 2010).

When the outcome variable is the residual life, semiparametric quantile models that apply the inverse probability of censoring weighting (IPCW) principle to address right-censored observations have been explored (Jung et al., 2009; Kim et al., 2012; Li et al., 2016). These approaches are based on non-smooth estimating functions with respect to regression parameters, and the estimates of the regression parameters are obtained either through zero-crossing of non-smooth estimating functions using grid search techniques (Jung et al., 2009) or by optimizing non-smooth objective functions with L_1-minimization algorithms (Kim et al., 2012; Li et al., 2016). While these methods are relatively straightforward to implement, an additional challenge lies in standard error estimation, which necessitates the computationally intensive use of a multiplier bootstrap method (Li et al., 2016). Alternatively, Jung et al. (2009) and Kim et al. (2012) utilized the minimum dispersion statistic and the empirical likelihood method, respectively, to bypass the need to directly estimate the variance of the regression parameter estimator for hypothesis testing and constructing confidence intervals. The non-smooth nature of the estimating functions in these approaches precludes the estimation of variance using the robust sandwich-type variance estimator typically employed in equation-based estimation methods. To lessen the associated computational burden, an induced smoothing was proposed (Brown and Wang, 2005), which modifies the non-smooth estimating equations into smooth ones. Leveraging the asymptotic normality of the non-smooth estimator, the smooth estimating functions are constructed by averaging out the random perturbations inherent in the non-smooth estimating functions. The resulting estimating functions become smooth with respect to the regression parameters, allowing for the straightforward application of standard numerical algorithms, such as the Newton-Raphson method. Furthermore, these smoothed estimating functions facilitate the straightforward computation of variances using the robust sandwich-type estimator. The induced smoothing approach has been employed in fitting semiparametric accelerated failure

time (AFT) models via the rank-based approach (Johnson and Strawderman, 2009; Chiou et al., 2021, 2015b; Kang, 2017). Regarding quantile regression, Choi et al. (2018) considered the induced smoothing approach under a competing-risks setting. All of these methods are based on modeling event times. Recently, Kim et al. (2023) proposed an induced smoothing estimator for fitting a semiparametric quantile regression model for residual life.

The availability of published R packages for fitting quantile regression models is somewhat limited. The `rq()`, `nlrq()`, `rqss()`, and `crq()` functions in the package **quantreg** (Koenker, 2022) are predominantly used and provide various features for fitting linear, nonlinear, non-parametric, and censored quantile regression models, respectively. The `rq()` function minimizes non-smooth objective functions to obtain point estimates of regression coefficients and can accommodate right-censored survival times by incorporating weights. By redefining survival times as the remaining lifetime at time t_0, one can also obtain a non-smoothed estimator for quantile regression models for residual life (Kim et al., 2012). On the other hand, the `nlrq()` function is designed to fit a nonlinear quantile regression model, while the `rqss()` function fits additive quantile regression models with nonparametric terms, including univariate components and bivariate components, using smoothing splines and total variation regularization techniques (Koenker et al., 1994; Koenker and Mizera, 2004). Furthermore, the `crq()` function fits a quantile regression model for censored data on the τ-th conditional quantile function of the response variable. Overall, the **quantreg** implements three methods for handling right-censored survival times: Powell (1986)'s estimator, Portnoy (2003)'s estimator and Peng and Huang (2008)'s estimator. However, none of the implemented methods in the `nlrq()`, `rqss()`, or `crq()` functions are applicable for handling censored residual life using the induced smoothing methods. The only function that implements the induced smoothing method is the `aftsrr()` function in the package **aftgee** (Chiou et al., 2021), but it is specifically designed for fitting semiparametric AFT models, which are not directly applicable to fitting quantile regression models.

Other R packages that can be used to fit quantile regression models for survival data include the package **ctqr** (Frumento, 2021), package **Brq** (Alhamzawi, 2020), package **brms** (Bürkner, 2018), and package **cmprskQR** (Dlugosz et al., 2019). The `ctqr()` function in the package **ctqr** implements the methods proposed in Frumento (2021) for right or interval-censored failure times with left-truncation. The `Bqr()` function in the package **Brq** implements Bayesian methods based on the asymmetric Laplace distribution. In the package **brms**, the `brm()` function with the `family=asym_laplace()` option enables the implementation of full Bayesian inference. The `crrQR()` function in the package **cmprskQR** allows fitting quantile regression models with competing risks. All of these R packages are designed for fitting quantile regression models for failure times defined from a baseline and are not applicable to the residual life setting.

The recently developed R package **qris** (Kim et al., 2022) provides an efficient tool for fitting semiparametric quantile regression models for residual life subject to right censoring. The **qris** package offers three methods for estimating the regression parameters: L_1-minimization of non-smooth objective functions, induced smoothing with a non-iterative approach, and an iterative procedure. For standard error estimation, the **qris** package provides two resampling-based approaches: the partial multiplier bootstrap and the full multiplier bootstrap methods. The partial multiplier bootstrap method utilizes the robust sandwich-type estimator by incorporating the sample variance of perturbed estimating functions, while the full multiplier bootstrap method is obtained by considering the sample variance from the solutions of perturbed estimating functions. To enhance the interpretability of results, the **qris** package incorporates graphical visualizations of covariate effects at different quantiles and base times, utilizing the plotting environment similar to that in the **ggplot2** package (Wickham et al., 2022), thereby allowing for extensive flexibility and customization. The ultimate goal of creating the **qris** package is to facilitate the easy incorporation of quantile regression for residual life into daily routines. The package **qris** is available on the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=qris`.

The rest of the article is organized as follows: Section Semiparametric quantile regression for residual life introduces a semiparametric regression model for quantiles of residual life and the estimation methods implemented in the package. Section Package implementation provides details about computing algorithms. Illustrations of the package using a simulated dataset and the real data from the North Central Cancer Treatment Group are presented in Section Illustration. Finally, in Section Conclusion, concluding remarks are provided along with some discussions.

2 Semiparametric quantile regression for residual life

Define T as the potential failure time that is subject to right censoring by C and $\mathbf{X}$ as a $p \times 1$ vector of covariates, where p is the number of covariates, including an intercept. The observed data consists of n independent copies of $(Z, \delta, \mathbf{X})$, where $Z = \min(T, C)$, $\delta = I(T \leq C)$, and $I(\cdot)$ is an indicator

function. We also assume T and C are marginally independent. Define the τ-th quantile of the residual life at $t_0 > 0$ as $\theta_\tau(t_0)$ that satisfies $P(T_i - t_0 \geq \theta_\tau(t_0) \mid T_i > t_0) = 1 - \tau$. We consider the semiparametric quantile regression model for the residual life (Kim et al., 2012, 2023). Given $T_i > t_0$,

$$\log(T_i - t_0) = \mathbf{X}_i^\top \boldsymbol{\beta}_0(\tau, t_0) + \epsilon_i, i = 1, \ldots, n, \tag{1}$$

where $\boldsymbol{\beta}_0(\tau, t_0)$ is a $p \times 1$ vector of regression coefficients, and ϵ_i is a random error having zero τ-th quantile. The quantile regression model for a continuous response (Koenker and Bassett Jr, 1978) is a special case of Equation (1) when $t_0 = 0$. For ease of notation, we omit τ and t_0 in $\boldsymbol{\beta}_0(\tau, t_0)$ and $\theta_\tau(t_0)$ and write $\boldsymbol{\beta}_0$ and θ. We present different estimation procedures to estimate $\boldsymbol{\beta}_0$ given τ and t_0 in the following.

2.1 Estimation using non-smooth functions

When there is no censoring, an estimator for β_0 in Equation (1) can be obtained by solving the estimating equation (Kim et al., 2012), where

$$\frac{1}{n}\sum_{i=0}^{n} I[T_i \geq t_0]\mathbf{X}_i \left\{ I\left[\log(T_i - t_0) \leq \mathbf{X}_i^\top \boldsymbol{\beta}\right] - \tau \right\} = 0. \tag{2}$$

However, Equation (2) cannot be directly used when $T_i - t_0$ is subject to right censoring. The IPCW technique can be incorporated into Equation (2) to account for the right censoring (Li et al., 2016). Specifically, in the presence of right censoring, the estimator for $\boldsymbol{\beta}_0$ in Equation (1) can be obtained as the root of the following weighted estimating equations:

$$U_{t_0}(\boldsymbol{\beta}, \tau) = \frac{1}{n}\sum_{i=1}^{n} I[Z_i \geq t_0]\mathbf{X}_i \left\{ I\left[\log(Z_i - t_0) \leq \mathbf{X}_i^\top \boldsymbol{\beta}\right] \frac{\delta_i}{\widehat{G}(Z_i)/\widehat{G}(t_0)} - \tau \right\}, \tag{3}$$

where $\widehat{G}(\cdot)$ is the Kaplan-Meier estimate of the survival function $G(\cdot)$ of the censoring time C and $\widehat{G}(t) = \prod_{i:t_i \leq t}(1 - \sum_{j=1}^{n}(1-\delta_j)I(Z_j \leq t_i)/\sum_{j=1}^{n} I(Z_j \geq t_i))$. A computational challenge arises because the exact solution to Equation (3) might not exist due to the non-smoothness in β caused by the involvement of indicator functions. When the exact solutions do not exist, the root of Equation (3) can be approximated by minimizing the L_1-objective function $L_{t_0}(\boldsymbol{\beta}, \tau)$ (Li et al., 2016),

$$\begin{aligned} L_{t_0}(\boldsymbol{\beta}, \tau) =& \frac{1}{n}\sum_{i=1}^{n} \frac{\delta_i I[Z_i > t_0]}{\widehat{G}(Z_i)/\widehat{G}(t_0)} \left|\log(Z_i - t_0) - \mathbf{X}_i^\top \beta\right| + \\ & \left| M - \boldsymbol{\beta}^\top \sum_{l=1}^{n} -\mathbf{X}_l \frac{\delta_l I[Z_l > t_0]}{\widehat{G}(Z_l)/\widehat{G}(t_0)} \right| + \left| M - \boldsymbol{\beta}^\top \sum_{l=1}^{n} 2\tau \mathbf{X}_l I[Z_l > t_0] \right|, \end{aligned}$$

where $M > 0$ bounds $\left|\boldsymbol{\beta}^\top \sum_{i=1}^{n} -\mathbf{X}_i \frac{\delta_i I[Z_i > t_0]}{\widehat{G}(Z_i)/\widehat{G}(t_0)}\right|$ and $\left|\boldsymbol{\beta}^\top \sum_{i=1}^{n} 2\tau \mathbf{X}_i I[Z_i > t_0]\right|$ from above. Numerically, the limit M is set to be an extremely large number, and the `qris()` function uses $M = 10^6$. Denote the resulting estimator to be $\widehat{\boldsymbol{\beta}}_{\text{NS}}$. It has been shown that $\widehat{\boldsymbol{\beta}}_{\text{NS}}$ is consistent for $\boldsymbol{\beta}_0$ and asymptotically normally distributed (Li et al., 2016).

Despite the well-established asymptotic properties, directly estimating the variance of $\widehat{\boldsymbol{\beta}}_{\text{NS}}$ is impractical because it involves the derivative of non-smooth functions. A multiplier bootstrap method has typically been employed (Li et al., 2016) to address this difficulty. The multiplier bootstrap method considers the perturbed version of $U_{t_0}(\beta, \tau)$, defined as

$$U^*_{t_0}(\beta, \tau) = \frac{1}{n}\sum_{i=1}^{n} \eta_i I[Z_i \geq t_0]\mathbf{X}_i \left\{ I\left[\log(Z_i - t_0) \leq \mathbf{X}_i^\top \boldsymbol{\beta}\right] \frac{\delta_i}{\widehat{G}^*(Z_i)/\widehat{G}^*(t_0)} - \tau \right\},$$

where $\eta_i, i = 1, \ldots, n$, are independently and identically (iid) generated from a positive random variable with unity mean and variance, and $\widehat{G}^*(\cdot)$ is a perturbed version of $\widehat{G}(\cdot)$, constructed as $\widehat{G}^*(t) = \prod_{i:t_i \leq t}(1 - \sum_{j=1}^{n}\eta_j(1-\delta_j)I(Z_j \leq t_i)/\sum_{j=1}^{n}\eta_j I(Z_j \geq t_i))$ for a given realization of η_i. On the other hand, a perturbed L_1-objective function, denoted as $L^*_{t_0}(\boldsymbol{\beta}, \tau)$, can be similarly

constructed, where

$$L^*_{t_0}(\boldsymbol{\beta},\tau) = \frac{1}{n}\sum_{i=1}^{n} \frac{\delta_i I[Z_i > t_0]}{\widehat{G}^*(Z_i)/\widehat{G}^*(t_0)} \left|\log(Z_i - t_0) - \mathbf{X}_i^\top \boldsymbol{\beta}\right| + \left|M - \boldsymbol{\beta}^\top \sum_{l=1}^{n} -\mathbf{X}_l \frac{\delta_l I[Z_l > t_0]}{\widehat{G}^*(Z_l)/\widehat{G}^*(t_0)}\right| + \left|M - \beta^\top \sum_{l=1}^{n} 2\tau \mathbf{X}_l \eta_l I[Z_l > t_0]\right|.$$

Solving for $U^*_{t_0}(\boldsymbol{\beta},\tau) = 0$, or equivalently, minimizing $L^*_{t_0}(\boldsymbol{\beta},\tau)$, yields one realization of $\widehat{\boldsymbol{\beta}}_{\text{NS}}$. The multiplier bootstrap variance is computed as the sample variance of a large number of realizations of $\widehat{\boldsymbol{\beta}}_{\text{NS}}$.

2.2 Estimation using induced smoothed functions

The regression coefficient in Equation (1) can be more efficiently obtained through the induced smoothed version of Equation (3). The induced smoothed estimating functions are constructed by taking the expectation with respect to a mean-zero random noise added to the regression parameters in Equation (3). Specifically,

$$\begin{aligned}\widetilde{U}_{t_0}(\boldsymbol{\beta},\tau,H) &= E_w\{U_{t_0}(\boldsymbol{\beta} + \mathbf{H}^{1/2}\mathbf{W},\tau)\} \\ &= \frac{1}{n}\sum_{i=1}^{n} I[Z_i > t_0]\mathbf{X}_i \left\{\Phi\left(\frac{\mathbf{X}_i^\top\boldsymbol{\beta} - \log(Z_i - t_0)}{\sqrt{\mathbf{X}_i^\top \mathbf{H}\mathbf{X}_i}}\right) \frac{\delta_i}{\widehat{G}(Z_i)/\widehat{G}(t_0)} - \tau\right\}, \end{aligned} \tag{4}$$

where $\mathbf{H} = O(n^{-1})$, $\mathbf{W} \sim N(0, \mathbf{I}_p)$ is a standard normal random vector, $\mathbf{I}_p$ is the $p \times p$ identity matrix, and $\Phi(\cdot)$ is the cumulative distribution function of a standard normal random variable. A typical choice for $\mathbf{H}$ is to fix it at $n^{-1}\mathbf{I}_p$, while some alternative choices are explored in Chiou et al. (2015a). Let $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ be the solution to $\widetilde{U}_{t_0}(\boldsymbol{\beta},\tau,\mathbf{H}) = 0$. Since Equation (4) is a smooth function in $\boldsymbol{\beta}$, the estimator can be obtained using standard numerical algorithms such as the Newton-Raphson method. Moreover, the induced smoothed estimator for $\boldsymbol{\beta}_0$ has been shown to be asymptotically equivalent to its non-smooth counterpart (Kim et al., 2023).

Following the idea in Section Estimation using non-smooth functions, the multiplier bootstrap procedure can be similarly employed to estimate the variance of $\widehat{\boldsymbol{\beta}}_{\text{IS}}$. The perturbed version of Equation (4) takes the form of

$$\widetilde{U}^*_{t_0}(\boldsymbol{\beta},\tau,\mathbf{H}) = \frac{1}{n}\sum_{i=1}^{n} \eta_i I[Z_i > t_0]\mathbf{X}_i \left\{\Phi\left(\frac{\mathbf{X}_i^\top\boldsymbol{\beta} - \log(Z_i - t_0)}{\sqrt{\mathbf{X}_i^\top \mathbf{H}\mathbf{X}_i}}\right) \frac{\widehat{G}^*(t_0)\delta_i}{\widehat{G}^*(Z_i)} - \tau\right\}. \tag{5}$$

The multiplier bootstrap procedure estimates the variance of $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ by calculating the sample variance of a large number of realizations of $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ obtained by repeatedly solving Equation (5).

It has been shown that the asymptotic variance $\text{Var}(\boldsymbol{\beta},\tau)$ can be decomposed into $\mathbf{A}(\boldsymbol{\beta})^\top\mathbf{V}(\boldsymbol{\beta})\mathbf{A}(\boldsymbol{\beta})$ (Kim et al., 2023), where the two components, $\mathbf{A}(\boldsymbol{\beta})$ and $\mathbf{V}(\boldsymbol{\beta})$, can be estimated separately. Since Equation (4) is a smooth function in $\boldsymbol{\beta}$, the slope matrix, $\mathbf{A}(\boldsymbol{\beta})$, can be conveniently estimated by differentiating $\widetilde{U}_{t_0}(\boldsymbol{\beta},\tau,\mathbf{H})$ with respect to $\boldsymbol{\beta}$. The explicit form of $\mathbf{A}(\boldsymbol{\beta})$ is as follows:

$$\begin{aligned}\mathbf{A}(\boldsymbol{\beta}) &= \frac{\partial \widetilde{U}_{t_0}(\boldsymbol{\beta},\tau,\mathbf{H})}{\partial\boldsymbol{\beta}} \\ &= \frac{1}{n}\sum_{i=1}^{n} I[Z_i > t_0]\mathbf{X}_i \frac{G(t_0)\delta_i}{G(Z_i)} \phi\left(\frac{\mathbf{X}_i{}^\top\boldsymbol{\beta} - \log(Z_i - t_0)}{\sqrt{\mathbf{X}_i{}^\top \mathbf{H}\mathbf{X}_i}}\right)\left(\frac{-\mathbf{X}_i}{\sqrt{\mathbf{X}_i{}^\top \mathbf{H}\mathbf{X}_i}}\right), \end{aligned} \tag{6}$$

where $\phi(\cdot)$ is the density function of the standard normal random variable.

The slope matrix, $\widehat{\mathbf{A}}(\widehat{\boldsymbol{\beta}}_{\text{IS}})$, can be evaluated directly by plugging in $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ and $\widehat{G}(\cdot)$. On the other hand, the variance of the estimating function, $\widehat{\mathbf{V}}(\boldsymbol{\beta})$, can be obtained by a computationally efficient resampling method motivated by the multiplier bootstrap procedure in Section Estimation using non-smooth functions. Specifically, we propose estimating $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}_{\text{IS}})$ as the simple variance of a large set of realizations of the perturbed version of $\widetilde{U}_{t_0}(\widehat{\boldsymbol{\beta}}_{\text{IS}},\tau,\mathbf{H})$ presented in Equation (5). We refer to this procedure as the partial multiplier bootstrapping approach because it utilizes the perturbed estimating function, similar to the full multiplier bootstrapping approach, but the computation of $\widehat{\mathbf{A}}(\widehat{\boldsymbol{\beta}}_{\text{IS}})$ and $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}_{\text{IS}})$ does not involve the repeated solving of the perturbed estimating equations. Thus, the partial multiplier bootstrapping approach is expected to be computationally more efficient

than the multiplier bootstrap method. A similar procedure and its performance have been studied in modeling failure times with semiparametric AFT models (Chiou et al., 2014, 2021).

2.3 Iterative procedure in induced smoothing estimation

The induced estimator $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ is obtained with a fixed $\mathbf{H}$, as described in Section Estimation using induced smoothed functions, and its variance is estimated separately. This estimation procedure can be viewed as a special case of the following iterative procedure, which updates $\mathbf{H}$ and $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ iteratively. Specifically, the iterative algorithm utilizes the Newton-Raphson method while sequentially updating $\widehat{\boldsymbol{\beta}}_{\text{IS}}$ and $\widehat{\text{Var}}(\widehat{\boldsymbol{\beta}}_{\text{IS}})$ until convergence. Similar iterative algorithms have also been considered previously in the induced smoothing approach for semiparametric AFT models (Johnson and Strawderman, 2009; Chiou et al., 2014, 2015b; Choi et al., 2018). The iterative procedure is summarized as follows:

Step 1: Set the initial values $\widehat{\boldsymbol{\beta}}^{(0)}$, $\widehat{\boldsymbol{\Sigma}}^{(0)} = \mathbf{I}_p$, and $\mathbf{H}^{(0)} = n^{-1}\widehat{\boldsymbol{\Sigma}}^{(0)}$.

Step 2: Given $\widehat{\boldsymbol{\beta}}^{(k)}$ and $\mathbf{H}^{(k)}$ at the k-th step, update $\widehat{\boldsymbol{\beta}}^{(k)}$ by

$$\widehat{\boldsymbol{\beta}}^{(k+1)} = \widehat{\boldsymbol{\beta}}^{(k)} - \widehat{\mathbf{A}}(\widehat{\boldsymbol{\beta}}^{(k)})^{-1}\widetilde{U}_{t_0}(\widehat{\boldsymbol{\beta}}^{(k)}, \tau, \mathbf{H}^{(k)}).$$

Step 3: Given $\widehat{\boldsymbol{\beta}}^{(k+1)}$ and $\widehat{\boldsymbol{\Sigma}}^{(k)}$, update $\widehat{\boldsymbol{\Sigma}}^{(k)}$ by

$$\widehat{\boldsymbol{\Sigma}}^{(k+1)} = \widehat{\mathbf{A}}(\widehat{\boldsymbol{\beta}}^{(k+1)})^{-1}\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}^{(k+1)}, \tau)\widehat{\mathbf{A}}(\widehat{\boldsymbol{\beta}}^{(k+1)})^{-1}.$$

Step 4: Set $\mathbf{H}^{(k+1)} = n^{-1}\widehat{\boldsymbol{\Sigma}}^{(k+1)}$. Repeat Steps 2, 3 and 4 until $\widehat{\boldsymbol{\beta}}^{(k)}$ and $\widehat{\boldsymbol{\Sigma}}^{(k)}$ converge.

The initial value, $\widehat{\boldsymbol{\beta}}^{(0)}$, could be chosen as $\widehat{\boldsymbol{\beta}}_{\text{NS}}$. We define $\widehat{\boldsymbol{\beta}}_{\text{IT}}$ and $\widehat{\boldsymbol{\Sigma}}_{\text{IT}}$ as the values of $\widehat{\boldsymbol{\beta}}^{(k)}$ and $\widehat{\boldsymbol{\Sigma}}^{(k)}$ at convergence, and $\widehat{\text{Var}}(\widehat{\boldsymbol{\beta}}_{\text{IT}}) = n^{-1}\widehat{\boldsymbol{\Sigma}}_{\text{IT}}$. In Step 3, $\widehat{\mathbf{V}}(\widehat{\boldsymbol{\beta}}^{(k+1)}, \tau)$ is obtained using the partial multiplier bootstrap approach. However, the full multiplier bootstrap approach can also be employed but would require longer computation times.

3 Package implementation

The main function in the **qris** package for estimating the regression parameters in the quantile regression model for residual life is the `qris()` function. The `qris()` function is written in C++ and incorporated into R using the **Rcpp** (Eddelbuettel et al., 2022a) and **RcppArmadillo** (Eddelbuettel et al., 2022b) packages. The synopsis of `qris` is:

```
> args(qris)
function (formula, data, t0 = 0, Q = 0.5, nB = 100, method = c("smooth",
"iterative", "nonsmooth"), se = c("fmb",
"pmb"), init = c("rq", "noeffect"), verbose = FALSE,
control = qris.control())
```

The required argument is `formula`, which specifies the quantile regression model to be fitted using the variables in `data`. The `formula` assumes that the response variable is a '`Surv`' object created by the `Surv()` function in the **survival** package (Therneau, 2021). This formula structure is commonly adopted for handling survival data in R, as seen in functions like `survreg()` and `coxph()` in the **survival** package. The argument `t0` specifies the base time used in defining residual life. The default value of `t0` is set to zero, in which case residual life reduces to a failure time. The `Q` argument is used to specify the target quantile of residual life to estimate, with the default value being set to 0.5 (median). The `nB` argument specifies the bootstrapping size used in standard error estimation, with the default value set to 100. The `method` argument specifies one of the three estimation methods: `"nonsmooth"`, `"smooth"`, and `"iterative"`, corresponding to the estimating procedures outlined in Sections Estimation using non-smooth functions, Estimation using induced smoothed functions, and Iterative procedure in induced smoothing estimation, respectively. Given the point estimates of the regression parameters, their standard errors can be estimated using one of two implemented methods: `se = "fmb"` and `se = "pmb"`. The `se = "fmb"` method employs a full-multiplier bootstrapping approach to estimate the variance by the sample variance of large realizations of $\widehat{\beta}$. The `se = "pmb"` method estimates the variance using a robust sandwich variance estimator and employs the computationally efficient partial multiplier bootstrapping approach described in Section Estimation using induced smoothed functions. The `"fmb"` option is available for all three point estimation methods, whereas the `"pmb"` option is not available for the `"nonsmooth"` point estimation method due to the lack of a closed-form sandwich variance estimator. The `init`

argument allows users to specify the initial value for estimating regression parameters by either a p-dimensional numerical vector or a character string. In the latter case, the options `init = "rq"` and `init = "noeffect"` correspond to the point estimate obtained from the `rq()` function in the **quantreg** package and a p-dimensional vector of zeros, respectively. The default value for `init` is `init = "rq"`. Among the three methods implemented for point estimation, `method = "smooth"` and `method = "nonsmooth"` are non-iterative, in the sense that point estimation is performed separately from the estimation of standard errors. On the other hand, `method = "iterative"` calculates point estimates and the corresponding standard error estimates simultaneously through iterative updates. When `method = "iterative"`, users can define specific convergence criteria using `qris.control()`. The available options in `qris.control()` are as follows.

```
> args(qris.control)
function (maxiter = 10, tol = 0.001, trace = FALSE)
```

The `maxiter` argument specifies the maximum number of iterations. The default value for `maxiter` is ten, as the proposed algorithm typically converges within ten steps based on our exploration. The convergence tolerance is controlled using the `tol` argument, which has a default value of `1e-3`. The `trace` argument takes a logical value and is used to determine whether to print the result for each iteration. The default setting is `trace = FALSE`. The '`qris`' object is fully compatible with many of R's generic functions, including `coef()`, `confint()`, `plot()`, `predict()`, `print()`, `residuals()`, `summary()`, and `vcov()`.

Among the available `S3` methods, a unique feature of the **qris** package's `S3 plot` method, when applied to a '`qris`' object, is its ability to automatically update the original object by extending the range of τ or t_0 values. This extension enables the generation of a covariate effect plot over the newly specified values of τ or t_0, providing a comprehensive visualization of the covariate effects across the extended range. The `S3` method for plotting a '`qris`' object is shown below.

```
> argsAnywhere(plot.qris)
function (x, t0s = NULL, Qs = NULL, nB = NULL, vari = NULL, byQs = FALSE,
    ggextra = NULL, ...)
NULL
```

The argument `x` is a '`qris`' object created using the `qris()` function. The `t0s` and `Qs` arguments are numeric vectors that enable users to specify the values of t_0 or τ for plotting the covariate effect. If `t0s` and `Qs` are not specified, the covariate effects are plotted against $\tau = 0.1, 0.2, \ldots, 0.9$ at the base time (t_0) inherited from the '`qris`' object specified in `x`. The `nB` argument is a numerical variable that controls the sample size for bootstrapping, used to compute standard error estimations based on the variance estimation specified in the original '`qris`' object. When `nB` is specified, the function calculates standard errors for all combinations of t_0 and τ specified in `t0s` and `Qs`, computes 95% confidence intervals accordingly, and includes them in the covariate effect plot. The `vari` argument is a character string that allows users to specify the names of the covariates they want to display in the effect plots. When the `vari` argument is not specified, all covariates will be included in the plots by default. The coefficient event plot can be plotted against the specified quantiles by setting `byQs = TRUE` or against the specified base times by setting `byQs = FALSE`. Finally, the `ggextra` argument allows users to pass additional graphical parameters to the **ggplot2** package, offering further customization options for the plots. When the `plot()` function is called, it internally invokes the `qris.extend()` function to compute the covariate effects at additional values. The syntax for the `qris.extend()` function is provided below:

```
> args(qris.extend)
function (x, t0s = NULL, Qs = NULL, nB = NULL, vari = NULL)
NULL
```

The arguments in `qris.extend()` are inherited from the arguments specified in the `plot()` function. To reduce runtime when repeatedly calling the `plot()`, one can calculate the desired covariate effects by applying `qris.extend()` outside of `plot()` first and then supply the results to `plot()`. This approach allows for pre-computation of the covariate effects, making it more efficient when generating multiple plots. Overall, the unique plotting feature in **qris** provides users with a seamless and effortless approach to conducting a comprehensive assessment of the covariate effects across different quantiles or base times.

4 Illustration

4.1 Simulated data

In this subsection, we present a simple simulation example to validate the implementations in the proposed qris package. The simulation involves five covariates, denoted as $X_1, \ldots, X_5$. Among these covariates, X_1 and X_4 follow a standard uniform distribution, X_2 follows a binomial distribution with a success probability of 0.5, X_3 follows a standard normal distribution, and X_5 follows a standard exponential distribution. We assume that X_2, X_3, X_4, and X_5 do not impact the residual life, meaning their corresponding coefficient values β_2, β_3, β_4, and β_5 are zero. The survival time T is generated from a Weibull distribution with the survival function $S(t) = \exp\{-(\rho t)^\kappa\}$ for $t > 0$, where $\kappa = 2$, and ρ is obtained by solving

$$\rho^{-1}\{(\rho t_0)^\kappa - \log(1-\tau)\}^{(1/\kappa)} - t_0 = \exp\{\beta_0 + \beta_1 X_1\}, \tag{7}$$

for a specified t_0 and τ. We set the intercept $\beta_0 = \log(5)$ and $\beta_1 = \log(2)$ at $t_0 = 0$. Given ρ, τ, and X_1, the true values of β_0 and β_1 can be obtained sequentially from Equation 7 for different $t_0 > 0$. In our case, the corresponding true values of β_0 are approximately 1.411 and 1.219 for $t_0 = 1$ and 2, respectively. Similarly, the true values of β_1 are approximately 0.797 and 0.907 for $t_0 = 1$ and 2, respectively. The closed-form expression for generating T is then $\{-\log(1-u)\}^{1/\kappa}/\rho$, where u is a uniform random variable over $(0, 1)$. Given these specifications, we have implemented the `data.gen()` function to generate simulation data. The `data.gen()` function takes four arguments: `n`, `t0`, `cen`, and `Q`, representing the sample size, t_0, censoring proportion, and τ, respectively. We generate censoring times C from an independent uniform distribution over $(0, c)$, where c is chosen to achieve the desired censoring proportions of 10% and 30%. Using the generated dataset, we fit the model using three different estimation methods: induced smoothing, non-smooth, and iterative-induced smoothing. All analyses were conducted on a 4.2 GHz Intel(R) quad Core(TM) i7-7700K central processing unit (CPU) using R 4.3.0 (R Core Team, 2021). The following code demonstrates the implementation of `data.gen()` to generate a simulation dataset.

```
> data.gen <- function(n, t0, cen = .3, Q = .5) {
+   if (!(t0 %in%  0:2))
+     stop("T0 is limited to three specific values: 0, 1, or 2.")
+   if (!(cen %in%  c(0, .1, .3)))
+     stop("Censoring is limited to three specific values: 0%, 10%, or 30%.")
+   if (!(Q %in% c(.25, .5)))
+     stop("Q is limited to two specific values: 0.25, or 0.50.")
+   censoring <- Inf
+   if (t0 == 0) {
+     if (cen == .1) censoring <- runif(n, 0, 125.1)
+     if (cen == .3) censoring <- runif(n, 0, 25.49)
+     beta0 <- log(5); beta1 <- log(2)
+   }
+   if (t0 == 1) {
+     if (cen == .1) censoring <- runif(n, 0, 120.8)
+     if (cen == .3) censoring <- runif(n, 0, 23.41)
+     beta0 <- 1.410748; beta1 <- 0.7974189
+   }
+   if (t0 == 2) {
+     if (cen == .1) censoring <- runif(n, 0, 120.6)
+     if (cen == .3) censoring <- runif(n, 0, 26.20)
+     beta0 <- 1.219403; beta1 <- 0.9070615
+   }
+   dat <- data.frame(censoring,
+                     Time0 = sqrt(-log(1 - runif(n))),
+                     X1 = runif(n),
+                     X2 = rbinom(n, 1, .5),
+                     X3 = rnorm(n),
+                     X4 = runif(n),
+                     X5 = rexp(n, 1))
+   rho <- (-log(1 - Q))^0.5 * (((exp(beta0 + beta1 * dat$X1) + t0)^2 - t0^2)^-0.5)
+   dat$Time0 <- dat$Time0 / rho
+   dat$Time <- pmin(dat$Time0, dat$censoring)
+   dat$status <- 1 * (dat$Time0 < dat$censoring)
```

```
+   subset(dat, select = c(Time, status, X1, X2, X3, X4, X5))
+ }
> set.seed(3)
> head(data.gen(200, 0))

Time status         X1 X2         X3         X4        X5
1  4.283379      0 0.09137221  0  2.1638425 0.33833437 0.8751895
2 14.797025      1 0.81196535  1  0.8803785 0.82101134 0.3648634
3  5.934559      1 0.60923418  1  0.5051163 0.56536790 0.3997803
4  7.223266      1 0.54550179  1  0.1105902 0.32417202 1.2169470
5 15.128553      1 0.86115736  0 -0.2928586 0.05825095 0.1835962
6  5.135852      1 0.28915525  0  0.7723200 0.94126325 0.3809120
```

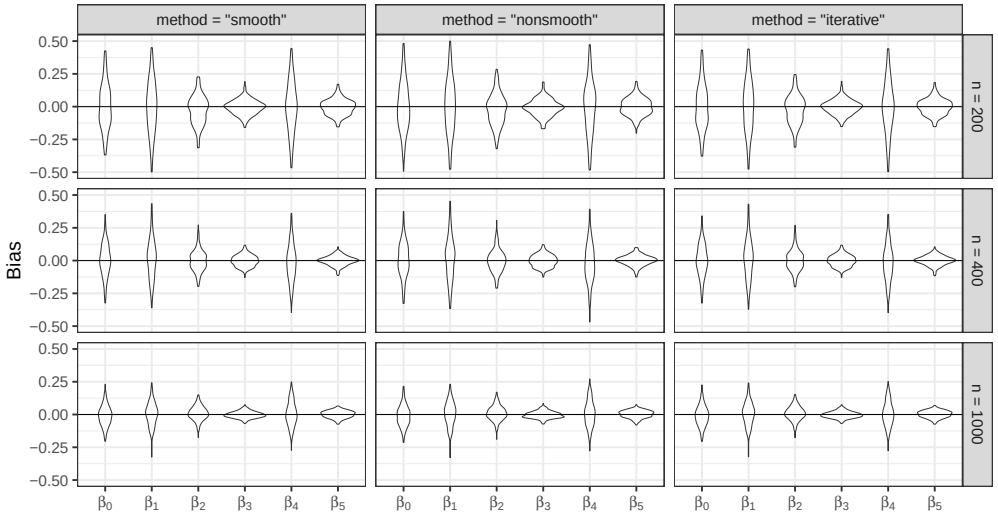
The `data.gen()` function generates a `data.frame` containing seven variables. The `Time` variable represents the observed survival time, while the `status` variable serves as the event indicator, taking the value 1 for observed events and 0 for censored observations. The variables `X1`, ..., `X5` are the covariates. The implementation in the `data.gen()` function generates the Weibull survival times using the inverse probability integral transform technique. Alternatively, users can use the `rweibull()` function with the parameters `shape = 2` and `scale = 1 / rho` to generate these Weibull survival times directly.

We assess the performance of the proposed implementation across various scenarios, including three sample sizes ($n = 200, 400, 1000$), three levels of t_0 $(0, 1, 2)$, two censoring proportions (10% and 30%), and two values of τ (0.25 and 0.50). For a given dataset, we apply the full-multiplier bootstrapping approach with 200 bootstrap samples to all three available estimating procedures: `method = "nonsmooth"`, `method = "smooth"`, and `method = "iterative"`. To facilitate the evaluation process, we create the `do_fmb()` function to record the coefficient estimates, standard errors, and computing times for fitting a single simulated dataset generated from `data.gen()`. The following is the implementation of the `do_fmb()` function and the corresponding code to run the simulation with 200 replications. We present the code and result of the simulation experiments conducted at three different sample sizes, with t_0 values set to 0 and 1, while holding the censoring proportion at 30% and τ value at 0.5. The results for other simulation scenarios are provided in the Supplementary Materials.

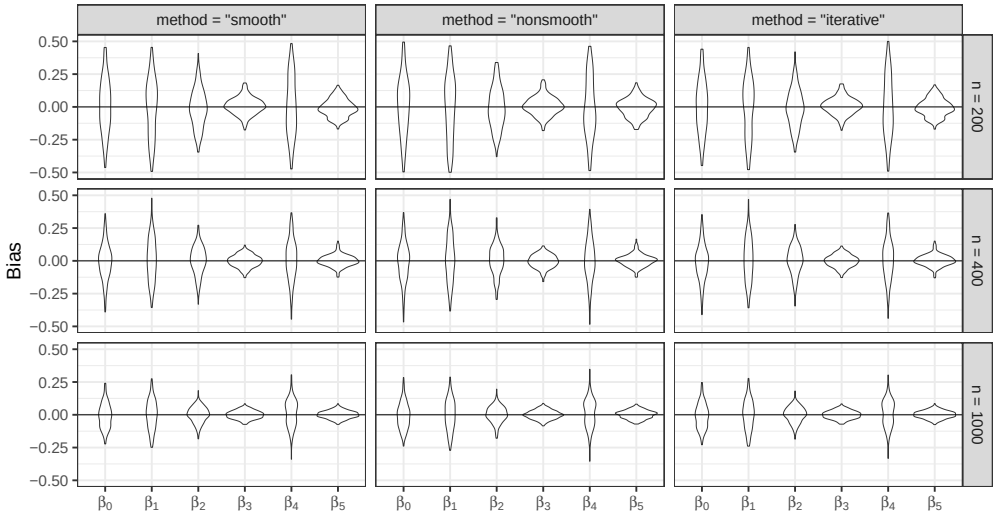
```
> do_fmb <- function(n, t0, cen, Q, nB) {
+   dat <- data.gen(n, t0, cen, Q)
+   fm <- Surv(Time, status) ~ X1 + X2 + X3 + X4 + X5
+   stamp <- NULL
+   stamp[1] <- Sys.time()
+   f1 <- qris(fm, data = dat, t0 = t0, Q = Q, nB = nB, method = "smooth", se = "fmb")
+   stamp[2] <- Sys.time()
+   f2 <- qris(fm, data = dat, t0 = t0, Q = Q, nB = nB, method = "nonsmooth", se = "fmb")
+   stamp[3] <- Sys.time()
+   f3 <- qris(fm, data = dat, t0 = t0, Q = Q, nB = nB, method = "iterative", se = "fmb")
+   stamp[4] <- Sys.time()
+   list(smooth = c(f1$coef, f1$std),
+     nonsmooth = c(f2$coef, f2$std),
+     iter = c(f3$coef, f3$std),
+     times = diff(stamp))
+ }
> B <- 200
> set.seed(2)
> sims0_fmb <- mapply(function(n, t0)
  +     replicate(B, do_fmb(n, t0 = t0, cen = .3, Q = .5, nB = 200)),
  +     n = c(200, 400, 1000), t0 = c(0, 0, 0), SIMPLIFY = F)
> sim1_fmb <- mapply(function(n, t0)
  +     replicate(B, do_fmb(n, t0 = t0, cen = .3, Q = .5, nB = 200)),
  +     n = c(200, 400, 1000), t0 = c(1, 1, 1), SIMPLIFY = F)
```

Figure 1 displays violin plots that provide visualizations of the empirical distribution of the coefficient estimates. As expected, all three estimators exhibit small biases, which are calculated as the difference between the point estimates (PE) and the true regression coefficients. Furthermore, the empirical distributions of the PEs demonstrate a normal-like shape, aligning with the asymptotic properties of the proposed method (Li et al., 2016; Kim et al., 2023). When the sample size is smaller (e.g., $n = 200$ and 400), the `nonsmooth` approach appears to yield slightly larger empirical standard errors (ESE) compared to the `smooth` or `iterative` approaches. However, when $n = 1000$, the ESEs

are similar across all approaches. On the other hand, the comprehensive simulation results presented in Table 1 of the Supplementary Materials confirm that all coefficient estimates closely approximate the true regression coefficients. On the other hand, the ESEs and the averaged estimated standard errors (ASE) are in close agreement for all scenarios, indicating the validity of the variance estimation. Furthermore, the computation times, which are presented separately in the upper panel of Table 1, indicate that when employing the full multiplier bootstrapping approach, the `nonsmooth` approach demonstrates a slight advantage in terms of computational efficiency over the `smooth` approach, while the `iterative` approach takes 5.1 to 9.5 times longer than the `smooth` approach. In summary, the timing results show that the proposed method can yield valid inference results within seconds, even with large datasets of up to 1000 observations or when using the computationally demanding full multiplier bootstrapping approach for variance estimation.



(a) $t_0 = 0$



(b) $t_0 = 1$

Figure 1: Comparison of the `smooth`, `nonsmooth` and `iterative` estimators with `se = "fmb"` under 30% censoring and $\tau = 0.5$.

When $t_0 = 0$, the targeted semiparametric quantile regression model for residual life simplifies to the standard quantile regression model for survival time. In such cases, existing functions like `crq()` from the **quantreg** package (Koenker, 2022) can be employed. A comparison between the performance of `crq()` and our proposed implementation when $t_0 = 0$ is presented in the Supplementary Materials, where the standard errors of the `crq()` are obtained from the bootstrap method with 200 bootstrap samples. Overall, the performance of `crq()` is comparable to the proposed methods in terms of bias

Table 1: Runtimes (in seconds) when `se = fmb` and `se = pmb`.

		$t_0 = 0$			$t_0 = 1$		
se	method	200	400	1000	200	400	1000
`fmb`	Smooth	0.103	0.174	0.471	0.106	0.178	0.480
	Nonsmooth	0.080	0.142	0.472	0.080	0.141	0.468
	Iterative	0.981	1.500	2.410	0.985	1.567	2.882
`pmb`	Smooth	0.022	0.052	0.223	0.022	0.053	0.224
	Iterative	0.296	0.580	1.407	0.296	0.581	1.435

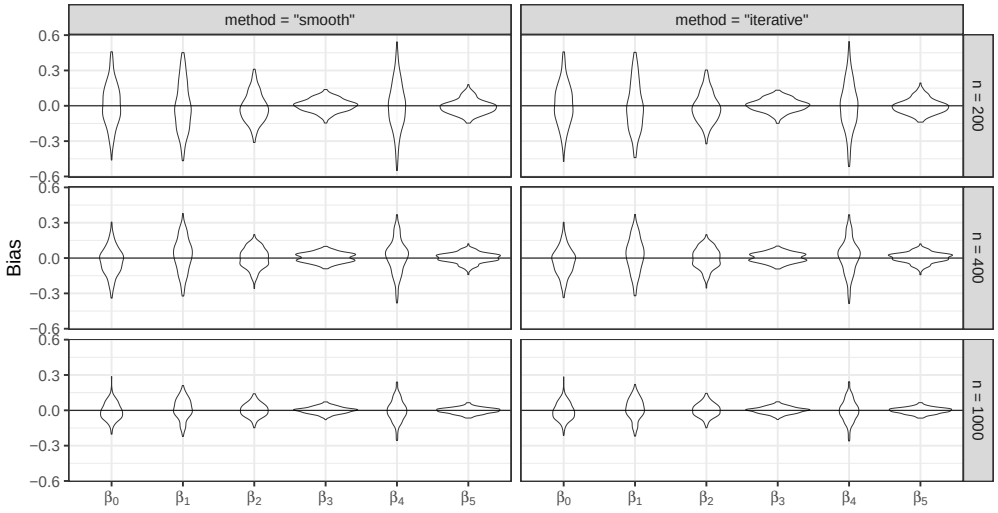
and standard errors. However, we have occasionally encountered situations where the `crq()` function fails to converge, particularly when the sample size is large, as in the case of $n = 1000$. In the other extended simulation scenarios outlined in the Supplementary Materials, which encompass various levels of t_0, censoring proportions, and τ, the proposed methods consistently exhibit satisfactory performance across all settings.

The true potential of the proposed smooth approach lies in its capability for efficient variance estimation through the implementation of the partial multiplier bootstrapping approach. This approach eliminates the need for repetitive solving of estimating equations, resulting in improved computational efficiency in variance estimation. To demonstrate its usefulness, we conducted a simulation using both the smooth approach and the iterative approach with the partial multiplier bootstrapping approach (`se = "pmb"`). This simulation was conducted under the settings of $\tau = 0.5$, $t_0 = 0$ and 1, and a 30% censoring rate. The `do_pmb()` function was accordingly modified as follows.

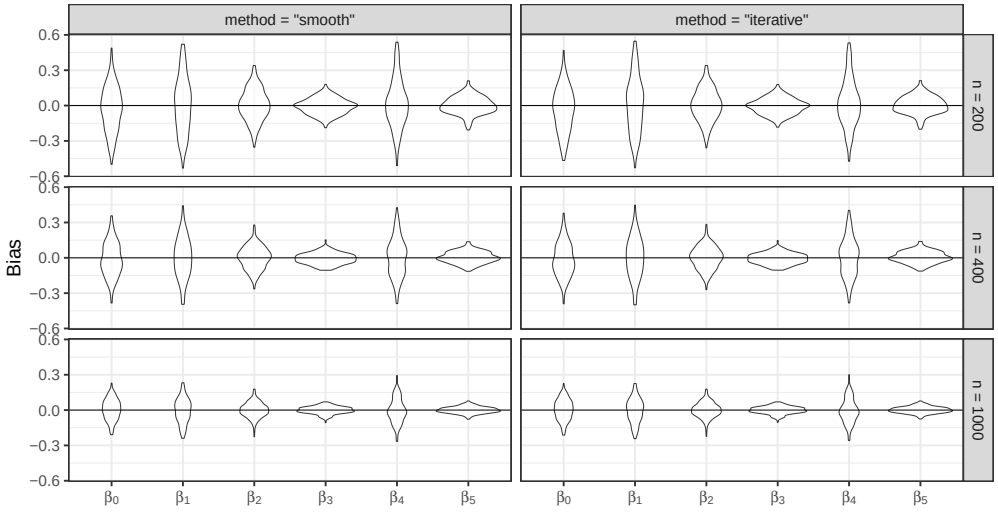
```
> do_pmb <- function(n, t0, cen, Q, nB) {
+   dat <- data.gen(n, t0, cen, Q)
+   fm <- Surv(Time, status) ~ X1 + X2 + X3 + X4 + X5
+   stamp <- NULL
+   stamp[1] <- Sys.time()
+   f1 <- qris(fm, data = dat, t0 = t0, Q = Q, nB = nB, method = "smooth", se = "pmb")
+   stamp[2] <- Sys.time()
+   f2 <- qris(fm, data = dat, t0 = t0, Q = Q, nB = nB, method = "iterative", se = "pmb")
+   stamp[3] <- Sys.time()
+   list(smooth = c(f1$coef, f1$std),
+     iter = c(f2$coef, f2$std),
+     times = diff(stamp))
+ }
> B <- 200
> set.seed(2)
> sims0_pmb <- mapply(function(n, t0)
+   replicate(B, do_pmb(n, t0 = t0, cen = .3, Q = .5, nB = 200)),
+   n = c(200, 400, 1000), t0 = c(0, 0, 0), SIMPLIFY = F)
> sims1_pmb <- mapply(function(n, t0)
+   replicate(B, do_pmb(n, t0 = t0, cen = .3, Q = .5, nB = 200)),
+   n = c(200, 400, 1000), t0 = c(1, 1, 1), SIMPLIFY = F)
```

The simulation results obtained using the partial multiplier bootstrapping approach are presented in Figure 2 and Tables 7 – 12 in the Supplementary Materials, while the computing times are displayed in the lower panel of Table 1. Overall, the estimation results obtained using `se = "pmb"` in Figure 2 closely resemble those in Figure 1 with `se = "fmb"`. As seen in Tables 7 and 8, the ESEs from the non-iterative and iterative methods are comparable, while the ASEs slightly overestimate the ESEs when the sample size is small. The gaps are slightly smaller for the iterative method, as shown in some cases (Johnson and Strawderman, 2009; Kim et al., 2021). The magnitudes of the differences are not large, and they also become smaller when the sample size reaches $n = 1000$. More importantly, the computing times with `se = "pmb"` show significant speed improvements compared to when `se = "fmb"` is used in every case; we observed up to 79% timing improvements.

After confirming the satisfactory performance of the proposed methodologies, we now proceed to illustrate the application of the `init` argument. This argument controls the initial values assigned to the root-finding algorithm's estimates and the plotting capacity of the **qris** package. For this illustrative example, we consider a simpler simulation scenario that involves a single binary covariate. This simplified simulation can be generated using the revised version of the `data.gen()` function provided below.



(a) $t_0 = 0$



(b) $t_0 = 1$

Figure 2: Comparison of the `smooth` and `iterative` estimators with `se = "pmb"` under 30% censoring and $\tau = 0.5$.

```
> ## Global parameters
+   rho0 <- .2 * sqrt(log(2))
+   rho1 <- .1 * sqrt(log(2))
> data.gen <- function(n) {
+   dat <- data.frame(censoring = runif(n, 0, 23.41),
+                     Time0 = sqrt(-log(1 - runif(n))),
+                     X = rbinom(n, 1, .5))
+   dat$Time0 <- ifelse(dat$X > 0, dat$Time0 / rho1, dat$Time0 / rho0)
+   dat$Time <- pmin(dat$Time0, dat$censoring)
+   dat$status <- 1 * (dat$Time0 < dat$censoring)
+   subset(dat, select = c(Time, status, X))
+ }
> set.seed(10)
> head(dat <- data.gen(200))
       Time status X
1  6.034713      1 1
2  7.181451      0 1
3  9.993908      0 1
```

```
4 16.225520      0 1
5  1.993033      0 1
6  5.277471      0 0
```

The updated `data.gen()` function returns a `data.frame` comprising three variables: `Time`, `status`, and `X`, representing the observed survival time, event indicator, and binary covariate, respectively. We will first illustrate the usage of the argument `init` by considering three different initial values: `init = "rq"`, `init = c(1,1)`, and a random vector `init = rnorm(2)`, all used in conjunction with the smooth estimator `method = "smooth"`. The following codes provide an example with different initial values.

```
> (random <- rnorm(2))
[1] 1.5025446 0.5904095
> f1 <- qris(Surv(Time, status) ~ X, data = dat, t0 = 1, init = "rq", nB = 0)
> f2 <- update(f1, init = c(1, 1))
> f3 <- update(f1, init = random)
> all.equal(f1$coef, f2$coef)
[1] TRUE
> all.equal(f2$coef, f3$coef)
[1] TRUE
```

The '`qris`' object, with its `call` component, is compatible with the `update()` function, a built-in function commonly used for updating the attributes of an existing object without requiring redundant and repetitive code. In the example above, we used the `update()` function to modify the initial value specification in `f1`. We observed that different initial values yield identical point estimates, thereby affirming the robustness of the proposed method against fluctuations in initial values.

The covariate effects, along with their associated 95% point-wise confidence intervals across various quantiles or base times, can be visually assessed by applying the generic function `plot()` to a '`qris`' object. We demonstrate this feature using the following `qris` fit, where the standard errors are obtained using `se = "pmb"`, $t_0 = 1$, and all other parameters are set to their default values. We update the `qris` fit with extended quantiles over $0.4, 0.5, 0.6, 0.7$ and plot the covariate effects against these quantiles using the `plot()` function.

```
> fit <- qris(Surv(Time, status) ~ X, data = dat, t0 = 1, se = "pmb")
> fit2 <- qris.extend(fit, Qs = 4:7 / 10)
```

The extended '`qris`' fit generated by the `qris.extend()` function inherits all the attributes from the original '`qris`' object and includes additional `ggdat` components. The following code compares the components of the returned values from the extended '`qris`' fit and the original '`qris`' fit.

```
> class(fit2)
[1] "qris"
> names(fit)
[1] "call"        "coefficient" "data"        "formula"     "para"
[6] "stderr"      "varNames"    "vcov"
> setdiff(names(fit2), names(fit))
[1] "ggdat"
```

Specifically, the extended '`qris`' fit inherits `call`, `coefficient`, `para`, `stderr`, `varNames`, and `vcov` from the original '`qris`' object. The `call` component is the function call from the original `qris()` fit, while `coefficient`, `stderr`, and `vcov` are used to store the point estimates, standard error estimates, and covariance matrix, respectively. The `para` component is a list containing the parameters specified during the fitting of the quantile regression model, and `varNames` is a character string representing the variable names in the function call. The newly added values are `ggdat` and `gg`. The `ggdat` is a data frame containing covariate information generated under the different quantiles and base times specified in the `qris.extend()`. Finally, the corresponding covariate effect plot can be generated by plotting the extended '`qris`' fit as follows.

```
> plot(fit2)
```

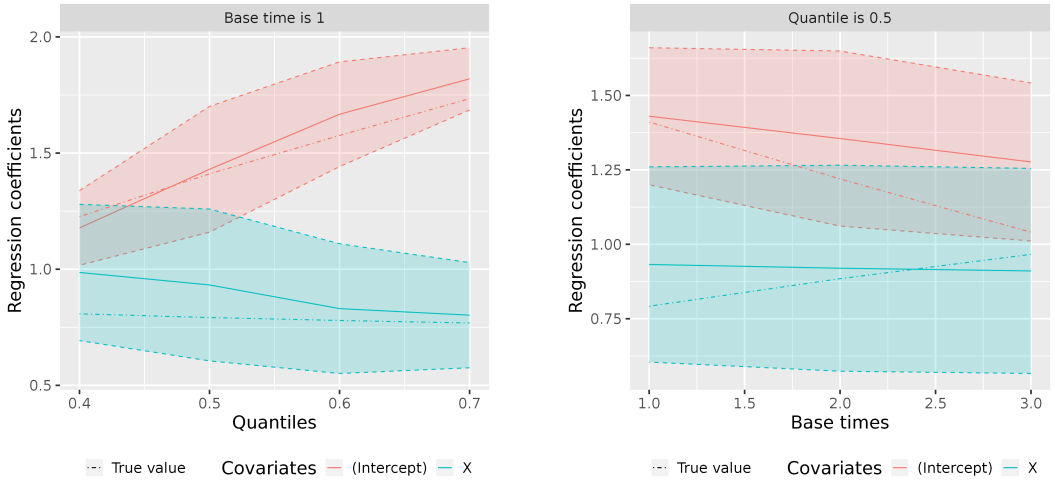
The true values of β's at different quantiles and base times, computed from Equation (7), can be implemented in the following commands.

```
> ## Global parameters
> r <- 2:1 * sqrt(log(2)) / 10
> k <- 2
> ## Function to calculate true beta
> trueB <- function(t0, tau) {
```

```
+     b <- log(1 / r * ((r * t0) ^ k - log(1 - tau))^(1 / k) - t0)
+     c(b[1], b[2] - b[1])
+ }
> ## True beta calculation
> true_Q <- c(t(sapply(4:7 / 10, trueB, t0 = 1)))
> true_t0 <- c(t(sapply(1:3, trueB, tau = .5)))
```

The following code extends the 'ggplot' objects generated by `plot.qris()` by adding additional layers of true value curves and incorporating various `ggplot` options. The resulting figures, Figure 3a and Figure 3b, present the output based on whether the covariate effects are plotted against quantiles or base times, respectively. This observed trend aligns with the specifications described in Equation (7), where increasing τ corresponds to an increasing β_0 while keeping ρ and X fixed. On the other hand, the covariate effect does not change with quantiles but slightly increases with base times, echoing the model specification where β_0 is inversely related to t_0 and β_1 increases as t_0 increases.

```
> library(ggplot2)
> plot(fit2) + theme(legend.position = "bottom") +
+    geom_line(aes(x = Qs, y = true_Q, col = variable, linetype = "True value")) +
+    scale_linetype_manual(name = "", values = c("True value" = "dotdash"))
> b <- plot(fit2, t0s = 1:3, byQs = F)
> b + theme(legend.position = "bottom") +
+    geom_line(aes(x = t0s, y = true_t0, col = variable,
+                  linetype = "True value")) +
+    scale_linetype_manual(name = "", values = c("True value" = "dotdash"))
```



(a) Plot for $Q \in \{0.4, \ldots, 0.7\}$ at $t_0 = 1$

(b) Plot for $t_0 \in \{1, \ldots, 3\}$ at $Q = 0.5$

Figure 3: (a) Estimated effects of covariate with the associated 95% pointwise confidence intervals for quantiles ranging from 0.4 to 0.7 at $t_0 = 1$. Red and blue solid lines are the point estimates of regression parameters for intercept and covariate X, respectively. Similarly, red and blue dotted lines are the upper and lower bounds of 95% pointwise confidence intervals for intercept and covariate X, respectively. (b) Estimated effects of covariate with the associated 95% pointwise confidence intervals for base times ranging from 1 to 3 at $\tau = 0.5$. Red and blue solid lines are the point estimates of regression parameters for intercept and covariate X, respectively. Similarly, red and blue dotted lines are the upper and lower bounds of 95% pointwise confidence intervals for intercept and covariate X, respectively.

4.2 North Central Cancer Treatment Group Lung Cancer Data

The North Central Cancer Treatment Group Lung Cancer Data records the survival of patients with advanced lung cancer, along with assessments of the patients' performance status measured by both physicians and the patients themselves (Loprinzi et al., 1994). The original objective of the study was to ascertain whether descriptive information from a patient-completed questionnaire could offer prognostic insights. The original objective of the study was to determine whether descriptive information from a patient-completed questionnaire could provide prognostic information. However,

for this illustration, we focus on how gender and weight loss affect the quantiles of residual life for patients diagnosed with advanced lung cancer at different time points. The lung cancer data are publicly available from the **survival** package (Therneau, 2021) as `lung`. The following code displays the structure of the `lung` dataset with variables of interest.

```
> data(cancer, package = "survival")
> str(subset(lung, select = c(time, status, sex, wt.loss)))
'data.frame':        228 obs. of  4 variables:
$ time   : num  306 455 1010 210 883 ...
$ status : num  2 2 1 2 2 1 2 2 2 2 ...
$ sex    : num  1 1 1 1 1 1 2 2 1 1 ...
$ wt.loss: num  NA 15 15 11 0 0 10 1 16 34 ...
```

The `lung` data contains 228 patients whose observed survival times in days and censoring status (1 = censored, 2 = dead) are recorded in the `time` and the `status` columns, respectively. Although the censoring status in this dataset is not recorded in the typical 0-1 fashion, the `Surv()` function is still applicable to create the corresponding "`Surv`" object. The `lung` data yields a censoring rate of 27.6% with a median survival time of 310 days. The covariates of interest are gender (`sex = 1` if male, `sex = 2` if female) and weight loss (`wt.loss`). In the following, we use the proposed semiparametric quantile regression models to assess the gender and standardized weight loss effects on different quantiles of residual life at different base times.

We first model the median residual life (`Q = 0.5`) when the base time is one month (`t0 = 30`). Since the estimated median survival times for combined lung cancers are typically less than one year, with a range of 8 to 13 months (Siegel et al., 2021), setting the base time at one month provides insight into how gender and weight loss impact the residual time in early follow-up. In the following, we obtain the regression coefficient estimates using the induced smoothing functions and the corresponding variance estimate with the partial multiplier bootstrap approach.

```
> lung$male <- factor(lung$sex, 1:2, c("Male", "Female"))
> lung$std.wt.loss <- scale(lung$wt.loss)
> fit1 <- qris(Surv(time, status) ~ male + std.wt.loss,
+              data = lung, t0 = 30, Q = .5, nB = 100,
+              method = "smooth", se = "pmb")
> summary(fit1)
Call:
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 30, Q = 0.5, nB = 100, method = "smooth",
se = "pmb")

qris Estimator
              estimate std.Error z.value p.value
(Intercept)     5.5611    0.0950  58.550  <2e-16 ***
maleFemale      0.4804    0.1805   2.661  0.0078 **
std.wt.loss    -0.0731    0.0837  -0.874  0.3824
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Subjects with missing values (in any of the variables relevant for the modeling task) are automatically removed when `qris()` is called. The estimated intercept implies that the median residual life for patients who have survived up to 30 days is $\exp(5.5611) = 260.1$ days for a male with an average weight loss. More interestingly, the summary shows that the gender effect is statistically significant at the 0.05 significance level, indicating that a female patient is expected to have a median residual life at 30 days that is $\exp(0.4804) = 1.617$ times that of a male patient with the same weight loss. The effect of the weight loss is not statistically significant at the 0.05 level. In addition to `summary()`, important statistics such as the coefficient and variance estimates can be extracted by `S3` methods `coef()` and `vcov()`, respectively.

```
> coef(fit1)
(Intercept)    maleFemale    std.wt.loss
5.56111984     0.48044228    -0.07307635
> vcov(fit1)
              (Intercept)   maleFemale    std.wt.loss
(Intercept)    0.009021459 -0.010944549   -0.003074041
maleFemale    -0.010944549  0.032594288    0.002847148
std.wt.loss   -0.003074041  0.002847148    0.006998314
```

Moreover, the corresponding 95% Wald-type confidence interval can be printed by applying the `confint()` function to the 'qris' object.

```
> confint(fit1)
                  2.5 %     97.5 %
(Intercept)   5.3749598 5.74727989
maleFemale    0.1265926 0.83429199
std.wt.loss  -0.2370390 0.09088626
```

The `update()` function can be conveniently applied to update existing 'qris' objects. The following examples update the `method` and `se` arguments from `fit1`. The updated results yield similar coefficient estimates, but the non-smooth procedure (`method = "nonsmooth"`) yields slightly greater standard error estimates.

```
> summary(fit2 <- update(fit1, method = "nonsmooth", se = "fmb"))
Call:
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 30, Q = 0.5, nB = 100, method = "nonsmooth",
se = "fmb")

qris Estimator
             estimate std.Error z.value p.value
(Intercept)    5.5585    0.1132  49.106  <2e-16 ***
maleFemale     0.4695    0.2015   2.331  0.0198 *
std.wt.loss   -0.0668    0.1029  -0.650  0.5159
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(update(fit1, method = "iterative"))
Call:
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 30, Q = 0.5, nB = 100, method = "iterative",
se = "pmb")

qris Estimator
             estimate std.Error z.value p.value
(Intercept)    5.5605    0.1016  54.712  <2e-16 ***
maleFemale     0.4807    0.1626   2.957  0.0031 **
std.wt.loss   -0.0720    0.0903  -0.797  0.4252
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

At a lower (`Q = 0.25`) and a higher (`Q = 0.75`) quantiles, the gender effect remains significant at the 0.05 significance level indicating female patients are associated with longer lower-quantile and higher-quantile residual life than male patients with the same weight loss. Among these models, we observed that female patients tend to have higher coefficient estimates when fitting higher-quantile residual life. While the sign of the estimated regression coefficient for weight loss changes to a negative value when considering the lower quantile, the effects remain statistically insignificant for both the lower and higher quantiles.

```
> summary(update(fit1, Q = 0.25))
Call:
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 30, Q = 0.25, nB = 100, method = "smooth",
se = "pmb")

qris Estimator
            estimate std.Error z.value p.value
(Intercept)   4.9111    0.1034  47.480  <2e-16 ***
maleFemale    0.4651    0.2041   2.279  0.0227 *
std.wt.loss   0.0543    0.0584   0.930  0.3525
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> summary(update(fit1, Q = 0.75))
Call:
```

```
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 30, Q = 0.75, nB = 100, method = "smooth",
se = "pmb")

qris Estimator
             estimate std.Error z.value p.value
(Intercept)    6.0748    0.1063  57.126  <2e-16 ***
maleFemale     0.5237    0.1487   3.522  0.0004 ***
std.wt.loss   -0.0171    0.1166  -0.147  0.8835
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

We also consider the base time at six months `t0 = 180`, which enables us to assess gender and weight loss effects in median residual time at a moderate length of follow-up. The estimated effect for the gender and weight loss increases as t_0 increases from 30 days to 180 days and becomes significant at the 0.05 significant level. Additionally, the effect of the weight loss seems to be associated with a shorter survival time after 180 days, with a p-value of 0.0008.

```
> summary(update(fit1, t0 = 180))
Call:
qris(formula = Surv(time, status) ~ male + std.wt.loss,
data = lung, t0 = 180, Q = 0.5, nB = 100, method = "smooth",
se = "pmb")

qris Estimator
             estimate std.Error z.value p.value
(Intercept)    5.2243    0.0912  57.255  <2e-16 ***
maleFemale     0.5821    0.1867   3.117  0.0018 **
std.wt.loss   -0.2515    0.0754  -3.337  0.0008 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

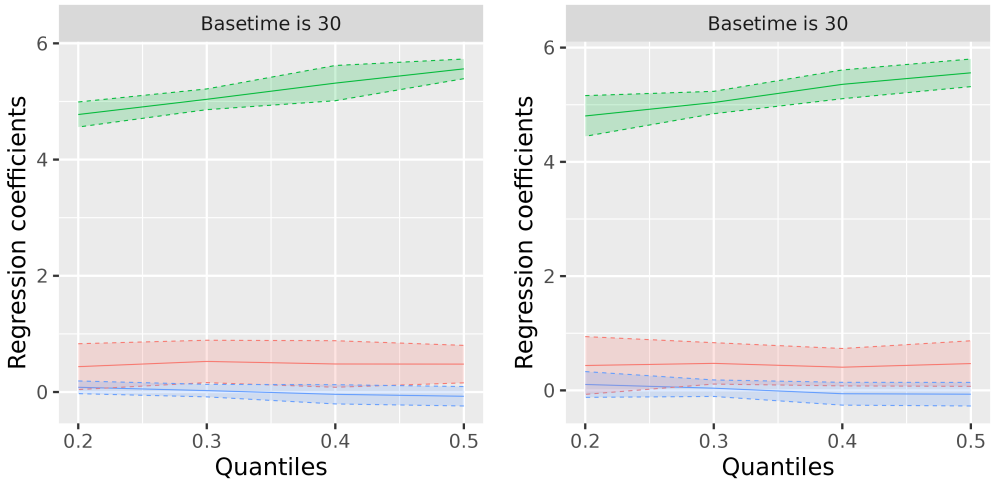
The ‘`qris`’ object is designed to be compatible with `S3` methods: `predict()` and `residuals()` functions. The following presents the fitted survival times for two hypothetical male and female patients with no weight loss, as well as the first five residual values for the dataset.

```
> lung.new <- data.frame(male = c("Male", "Female"), std.wt.loss = 0)
> predict(fit2, newdata = lung.new)
       1        2
444.9026 289.4422
> head(residuals(fit2), 5)
       1          2          3          4          5
-20.86127 -575.86127  232.44474 -416.82295 -555.82295
```

To better understand the covariate effects on different quantiles of residual time and across different base times, we plot the estimated regression coefficients of the intercept, sex, and weight loss in `fit1` and `fit2`. Figures 4a and 4b display the estimated regression coefficients when `method = "smooth"` and `method = "nonsmooth"`, respectively, at different quantiles ranging from 0.2 and 0.5 at $t_0 = 30$ days. The `plot.qris()` function is currently not available for the iterative estimator. This is mainly due to an extended computation time involved, as indicated by our simulation results, and the nature of plotting that necessitates computations across various quantiles or base times. As expected, the two plots show very similar patterns. We plot the estimated regression coefficients of the intercept, sex, and weight loss for different quantiles in the range of 0.2 to 0.5 at $t_0 = 50$, 60, 70, and 80 days (Figure 4c), as well as for different base times in the range of 50 to 80 days at $\tau = 0.2$, 0.3, 0.4, and 0.5 (Figure 4d). The estimation method used is non-iterative induced smoothed estimation (`method = "smooth"`). In Figure 4c, the estimated intercept increases as the quantile increases (for a given base time). The estimated slopes for sex remain largely the same, but those for weight loss tend to decrease slightly across different quantiles (for a given base time). These patterns remain consistent for different base times. In Figure 4d, the estimated intercepts increase as the quantiles increase, but with a given quantile, they remain flat across the different base times considered. The estimated regression coefficients for the two covariates do not appear to change significantly for different base times.

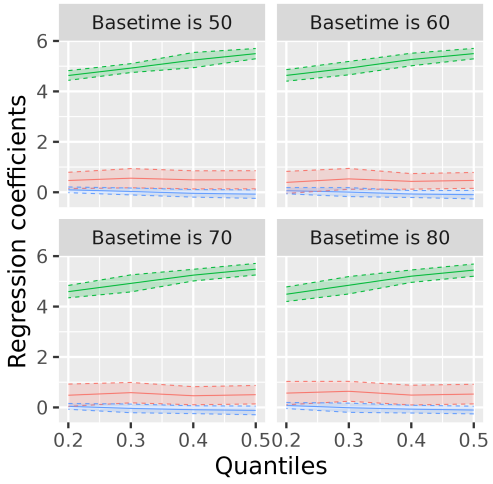
```
> hide <- theme(legend.position = "none")
> plot(fit1, Qs = 2:5 / 10, byQs = TRUE, ggextra = hide)
> plot(fit2, Qs = 2:5 / 10, byQs = TRUE, ggextra = hide)
```

```
> plot(fit1, Qs = 2:5 / 10, t0s = 5:8 * 10, byQs = TRUE, ggextra = hide)
> plot(fit1, Qs = 2:5 / 10, t0s = 5:8 * 10, byQs = FALSE, ggextra = hide)
```

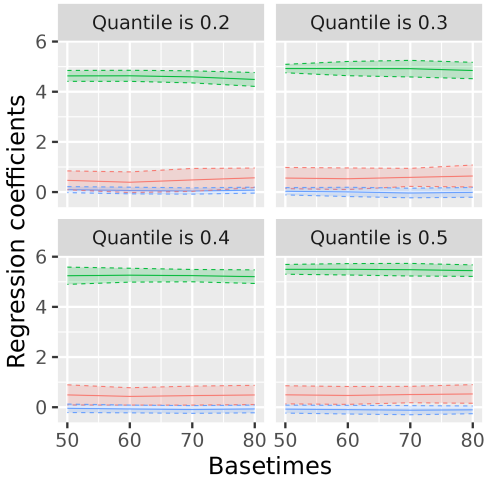


(a) `method = "smooth"` and `se = "pmb"`

(b) `method = "nonsmooth"` and `se = "fmb"`



(c) `method = "smooth"` and `se = "pmb"`



(d) Multiple covariate effect plot against base time

Figure 4: Green, red and blue lines are the point estimates of regression parameters for intercept, covariate sex and covariate weight loss, respectively. Solid line and dotted line are the point estimates and the upper and lower bounds of 95% pointwise confidence intervals for each regression coefficient. (a) `method = "smooth"` and `se = "pmb"` ($\tau = 0.2, 0.3, 0.4, 0.5, t_0 = 30$) (b) `method = "nonsmooth"` and `se = "fmb"` ($\tau = 0.2, 0.3, 0.4, 0.5, t_0 = 30$) (c) `method = "smooth"` and `se = "pmb"` against quantiles ($\tau = 0.2, 0.3, 0.4, 0.5, t_0 = 50, 60, 70, 80$) (d) `method = "smooth"` and `se = "pmb"` against base times ($\tau = 0.2, 0.3, 0.4, 0.5, t_0 = 50, 60, 70, 80$)

5 Conclusion

The purpose of the qris package is to provide a comprehensive tool for fitting quantile regression models on residual life for right-censored survival data, with the aim of promoting widespread dissemination and utilization. This package implements one estimation method based on non-smooth estimating functions and two estimation methods based on their induced smoothed versions. The non-smooth estimator is calculated through L_1-type minimization while incorporating the IPCW technique, and its variance is calculated using full multiplier bootstrapping. The first type of the induced smoothed estimator, a non-iterative version, directly solves estimating functions, and its variance can be calculated using either the full multiplier bootstrapping or the robust sandwich form with partial multiplier bootstrapping. As evidenced by the simulation results, this enables one to substantially reduce computing times without sacrificing estimation accuracy and stability

compared to the original non-smooth function-based method. The iterative smoothed estimator has an advantage in obtaining more precise estimates than its non-iterative version, although it requires longer computing times. For all these methods, estimates of the regression coefficients and their variances can be calculated at user-defined quantiles and base times, as long as they are identifiable. Additionally, the package provides features for plotting estimates with associated 95% confidence intervals against quantiles and base times using the generic `plot` function. These plots visualize patterns of estimates at different quantiles and base times, helping users to easily grasp the overall picture. The package **qris** and its included functions are verified through illustrations using simulated data with interpretation of the results demonstrated through a real data application.

Some possible directions for extending our package are as follows. Efforts can be made to reduce the computational burden associated with variance estimation, which currently accounts for a significant portion of the computing time. In particular, the iterative-induced smoothed method employs the partial multiplier bootstrap method to calculate variance estimates in each iteration. Since this method requires multiple iterations, it is crucial to explore more computationally efficient variance estimation procedures for each iteration to reduce the currently relatively longer computation time. One approach is to utilize a closed-form estimation of the mid-part of the sandwich-type variance, as discussed in Chiou et al. (2014); Choi et al. (2018). Implementing this direct variance estimation in each iteration is expected to further enhance computation efficiency. Another direction is to generalize the approaches to allow for the inclusion of sampling weights, which is useful for bias correction when failure time data are generated from non-random sampling designs, such as case-cohort designs (Prentice, 1986; Chiou et al., 2015b). The current estimating functions implemented in the **qris** package assume that the data are randomly sampled, with sampling weights set to 1." To the best of our knowledge, there is a lack of model-checking procedures and model-comparison methods specifically designed for the non-smooth estimator, and a logical next step would be to develop these procedures for subsequent integration into the package.

References

R. Alhamzawi. *Brq: Bayesian Analysis of Quantile Regression Models*, 2020. URL https://CRAN.R-project.org/package=Brq. R package version 3.0. [p115]

B. Brown and Y.-G. Wang. Standard errors and covariance matrices for smoothed rank estimators. *Biometrika*, 92(1):149–158, 2005. URL https://doi.org/10.1093/biomet/92.1.149. [p114]

P.-C. Bürkner. Advanced Bayesian multilevel modeling with the R package brms. *The R Journal*, 10(1):395–411, 2018. doi: 10.32614/RJ-2018-017. [p115]

Y. Chen, N. Jewell, X. Lei, and S. Cheng. Semiparametric estimation of proportional mean residual life model in presence of censoring. *Biometrics*, 61(1):170–178, 2005. URL https://doi.org/10.1111/j.0006-341X.2005.030224.x. [p114]

Y. Q. Chen. Additive expectancy regression. *Journal of the American Statistical Association*, 102 (477):153–166, 2007. URL https://doi.org/10.1198/016214506000000870. [p114]

Y. Q. Chen and S. Cheng. Linear life expectancy regression with censored data. *Biometrika*, 93(2): 303–313, 2006. URL https://doi.org/10.1093/biomet/93.2.303. [p114]

S. Chiou, S. Kang, and J. Yan. Rank-based estimating equations with general weight for accelerated failure time models: An induced smoothing approach. *Statistics in Medicine*, 34(9):1495–1510, 2015a. URL https://doi.org/10.1002/sim.6415. [p117]

S. H. Chiou, S. Kang, and J. Yan. Fast accelerated failure time modeling for case-cohort data. *Statistics and Computing*, 24(4):559–568, 2014. URL https://doi.org/10.1007/s11222-013-9388-2. [p118, 131]

S. H. Chiou, S. Kang, and J. Yan. Semiparametric accelerated failure time modeling for clustered failure times from stratified sampling. *Journal of the American Statistical Association*, 110(510): 621–629, 2015b. URL https://doi.org/10.1080/01621459.2014.917978. [p115, 118, 131]

S. H. Chiou, S. Kang, and J. Yan. *aftgee: Accelerated failure time model with generalized estimating equations*, 2021. URL https://CRAN.R-project.org/package=aftgee. R package version 1.1.6. [p115, 118]

S. Choi, S. Kang, and X. Huang. Smoothed quantile regression analysis of competing risks. *Biometrical Journal*, 60(5):934–946, 2018. URL https://doi.org/10.1002/bimj.201700104. [p115, 118, 131]

S. Dlugosz, L. Peng, R. Li, and S. Shi. *cmprskQR: Analysis of competing risks using quantile regressions*, 2019. URL https://CRAN.R-project.org/package=cmprskQR. R package version 0.9.2. [p115]

D. Eddelbuettel, R. Francois, J. Allaire, K. Ushey, Q. Kou, N. Russell, I. Ucar, D. Bates, and J. Chambers. *Rcpp: Seamless R and C++ Integration*, 2022a. URL https://CRAN.R-project.org/package=Rcpp. R package version 1.0.9. [p118]

D. Eddelbuettel, R. Francois, D. Bates, B. Ni, and C. Sanderson. *RcppArmadillo: 'Rcpp' Integration for the 'Armadillo' Templated Linear Algebra Library*, 2022b. URL https://CRAN.R-project.org/package=RcppArmadillo. R package version 0.11.1.1.0. [p118]

P. Frumento. *ctqr: Censored and truncated quantile regression*, 2021. URL https://CRAN.R-project.org/package=ctqr. R package version 2.0. [p115]

Y. Huang. Quantile calculus and censored regression. *Annals of Statistics*, 38(3):1607, 2010. doi: 10.1214/09-AOS771. [p114]

L. M. Johnson and R. L. Strawderman. Induced smoothing for the semiparametric accelerated failure time model: Asymptotics and extensions to clustered data. *Biometrika*, 96(3):577–590, 2009. URL https://doi.org/10.1093/biomet/asp025. [p115, 118, 123]

S.-H. Jung. Quasi-likelihood for median regression models. *Journal of the American Statistical Association*, 91(433):251–257, 1996. URL https://doi.org/10.1080/01621459.1996.10476683. [p114]

S.-H. Jung, J.-H. Jeong, and H. Bandos. Regression on quantile residual life. *Biometrics*, 65(4): 1203–1212, 2009. URL https://doi.org/10.1111/j.1541-0420.2009.01196.x. [p114]

S. Kang. Fitting semiparametric accelerated failure time models for nested case–control data. *Journal of Statistical Computation and Simulation*, 87(4):652–663, 2017. URL https://doi.org/10.1080/00949655.2016.1222611. [p115]

K. Kim, J. Ko, and S. Kang. Comparison of variance estimation methods in semiparametric accelerated failure time models for multivariate failure time data. *Japanese Journal of Statistics and Data Science*, 4(2):1179–1202, 2021. URL https://doi.org/10.1007/s42081-021-00126-y. [p123]

K. H. Kim, S. Kang, and S. H. Chiou. *qris: Quantile regression model for residual lifetime using an induced smoothing approach*, 2022. URL https://CRAN.R-project.org/package=qris. R package version 1.0.0. [p114, 115]

K. H. Kim, D. J. Caplan, and S. Kang. Smoothed quantile regression for censored residual life. *Computational Statistics*, 38:1001–1022, 2023. URL https://doi.org/10.1007/s00180-022-01262-z. [p115, 116, 117, 121]

M.-O. Kim, M. Zhou, and J.-H. Jeong. Censored quantile regression for residual lifetimes. *Lifetime Data Analysis*, 18(2):177–194, 2012. URL https://doi.org/10.1007/s10985-011-9212-2. [p114, 115, 116]

R. Koenker. *quantreg: Quantile regression*, 2022. URL https://CRAN.R-project.org/package=quantreg. R package version 5.87. [p115, 122]

R. Koenker and G. Bassett Jr. Regression quantiles. *Econometrica: Journal of the Econometric Society*, pages 33–50, 1978. URL https://doi.org/10.2307/1913643. [p114, 116]

R. Koenker and I. Mizera. Penalized triograms: Total variation regularization for bivariate smoothing. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 66(1):145–163, 2004. URL https://doi.org/10.1111/j.1467-9868.2004.00437.x. [p115]

R. Koenker, P. Ng, and S. Portnoy. Quantile smoothing splines. *Biometrika*, 81(4):673–680, 1994. URL https://doi.org/10.1093/biomet/81.4.673. [p115]

R. Li, X. Huang, and J. E. Cortes. Quantile residual life regression with longitudinal biomarker measurements for dynamic prediction. *Journal of the Royal Statistical Society. Series C: Applied Statistics*, 65(5):755–773, 2016. URL http://www.jstor.org/stable/44681854. [p114, 116, 121]

S. Liu and S. K. Ghosh. Regression analysis of mean residual life function. Technical report, North Carolina State University. Dept. of Statistics, 2008. URL https://repository.lib.ncsu.edu/bitstream/handle/1840.4/3041/mimeo2613.pdf?sequence=1. [p114]

C. L. Loprinzi, J. A. Laurie, H. S. Wieand, J. E. Krook, P. J. Novotny, J. W. Kugler, J. Bartel, M. Law, M. Bateman, and N. E. Klatt. Prospective evaluation of prognostic variables from patient-completed questionnaires. North Central Cancer Treatment Group. *Journal of Clinical Oncology*, 12(3):601–607, 1994. URL https://doi.org/10.1200/JCO.1994.12.3.601. [p126]

G. Maguluri and C.-H. Zhang. Estimation in the mean residual life regression model. *Journal of the Royal Statistical Society: Series B (Methodological)*, 56(3):477–489, 1994. URL https://doi.org/10.1111/j.2517-6161.1994.tb01994.x. [p114]

D. Oakes and T. Dasu. A note on residual life. *Biometrika*, 77(2):409–410, 1990. URL https://doi.org/10.1093/biomet/77.2.409. [p114]

D. Oakes and T. Dasu. Inference for the proportional mean residual life model. *Lecture Notes-Monograph Series*, pages 105–116, 2003. URL http://www.jstor.org/stable/4356266. [p114]

L. Peng and Y. Huang. Survival analysis with quantile regression models. *Journal of the American Statistical Association*, 103(482):637–649, 2008. URL https://doi.org/10.1198/016214508000000355. [p114, 115]

S. Portnoy. Censored regression quantiles. *Journal of the American Statistical Association*, 98(464): 1001–1012, 2003. URL https://doi.org/10.1198/016214503000000954. [p114, 115]

S. Portnoy and R. Koenker. The gaussian hare and the laplacian tortoise: computability of squared-error versus absolute-error estimators. *Statistical Science*, 12(4):279–300, 1997. URL https://doi.org/10.1214/ss/1030037960. [p114]

J. L. Powell. Censored regression quantiles. *Journal of Econometrics*, 32(1):143–155, 1986. URL https://doi.org/10.1016/0304-4076(86)90016-3. [p115]

R. L. Prentice. A case-cohort design for epidemiologic cohort studies and disease prevention trials. *Biometrika*, 73(1):1–11, 1986. URL https://doi.org/10.1093/biomet/73.1.1. [p131]

R Core Team. *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2021. URL https://www.R-project.org/. [p120]

R. L. Siegel, K. D. Miller, H. E. Fuchs, and A. Jemal. Cancer statistics, 2021. *CA: A Cancer Journal for Clinicians*, 71(1):7–33, 2021. URL https://doi.org/10.3322/caac.21654. [p127]

T. M. Therneau. *survival: Survival analysis*, 2021. URL https://CRAN.R-project.org/package=survival. R package version 3.2-13. [p118, 127]

Y. Wei, A. Pere, R. Koenker, and X. He. Quantile regression methods for reference growth charts. *Statistics in Medicine*, 25(8):1369–1382, 2006. URL https://doi.org/10.1002/sim.2271. [p114]

H. Wickham, W. Chang, L. Henry, T. L. Pedersen, K. Takahashi, C. Wilke, K. Woo, H. Yutani, and D. Dunnington. *ggplot2: Create elegant data visualisations using the grammar of graphics*, 2022. URL https://CRAN.R-project.org/package=ggplot2. R package version 3.3.6. [p115]

Z. Ying, S.-H. Jung, and L.-J. Wei. Survival analysis with median regression models. *Journal of the American Statistical Association*, 90(429):178–184, 1995. URL https://doi.org/10.1080/01621459.1995.10476500. [p114]

Z. Zhang, X. Zhao, and L. Sun. Goodness-of-fit tests for additive mean residual life model under right censoring. *Lifetime Data Analysis*, 16(3):385–408, 2010. URL https://doi.org/10.1007/s10985-010-9152-2. [p114]

Kyu Hyun Kim
Department of Statistics and Data Science and *Department of Applied Statistics*
Yonsei University
50 Yonsei-ro, Seodaemun-gu, Seoul
Republic of Korea
kyuhyunkim07@yonsei.ac.kr

Sangwook Kang
Department of Statistics and Data Science and *Department of Applied Statistics*
Yonsei University
50 Yonsei-ro, Seodaemun-gu, Seoul

Republic of Korea
kanggi1@yonsei.ac.kr

Sy Han Chiou
Department of Statistics and Data Science
Southern Methodist University
P.O. Box 750332, Dallas, TX
USA
schiou@smu.edu
https://www.sychiou.com/

nortsTest: An R Package for Assessing Normality of Stationary Processes

by Asael Alonzo Matamoros, Alicia Nieto-Reyes, and Claudio Agostinelli

Abstract Normality is the central assumption for analyzing dependent data in several time series models, and the literature has widely studied normality tests. However, the implementations of these tests are limited. The nortsTest package is dedicated to fill this void. The package performs the asymptotic and bootstrap versions of the tests of Epps and Lobato and Velasco and the tests of Psaradakis and Vavra, random projections and El Bouch for normality of stationary processes. These tests are for univariate stationary processes but for El Bouch that also allows bivariate stationary processes. In addition, the package offers visual diagnostics for checking stationarity and normality assumptions for the most used time series models in several R packages. This work aims to show the package's functionality, presenting each test performance with simulated examples and the package utility for model diagnostic in time series analysis.

1 Introduction

Normality (*a set of observations sampled from a Gaussian process*) is an essential assumption in various statistical models. Therefore, developing procedures for testing this assumption is a topic that has gained popularity over several years. Most existing literature and implementation is dedicated to independent and identically distributed random variables (D'Agostino and Stephens, 1986); no results show that these tests are consistent when applied to stationary processes. For this context, several tests have been proposed over the years, but as far as we know, no `R` package or consistent implementation exists.

The proposed **nortsTest** package provides seven test implementations to check normality of stationary processes. This work aims to present a review of these tests and introduce the package functionality. Thus, its novelty lies in being the first package and paper dedicated to the implementation of normality tests for stationary processes. The implemented tests are: (i) the asymptotic *Epps* test, (Epps, 1987) and (Nieto-Reyes et al., 2014), based on the characteristic function and (ii) its sieve bootstrap approximation (Psaradakis and Vávra, 2020), (iii) the corrected *Skewness-Kurtosis* (SK) test implemented by Lobato and Velasco (2004) as an asymptotic test and (iv) by Psaradakis and Vávra (2020) with a sieve bootstrap approximation, (v) the *random projections test* proposed by Nieto-Reyes et al. (2014), which makes use of the tests in (i) and (iii), (vi) the *Psadarakis and Vávra test* (Psaradakis and Vávra, 2017) that uses a bootstrap approximation of the Anderson and Darling (1952) test statistic for stationary linear processes and (vii) a normality test by El Bouch et al. (2022) for multivariate dependent samples. Tests (i) to (vi) are for univariate stationary processes.

Furthermore, we propose the `check_residual()` function for checking time-series models' assumptions. This function returns a report for stationarity, seasonality, normality tests and visual diagnostics. `check_residual()` supports models from the most used packages for time-series analysis, such as the packages **forecast** (Hyndman and Khandakar, 2008) and **aTSA** (Qiu, 2015) and even functions in the base `R` (Team, 2018); for instance, it supports the `HoltWinters` (stats `R` package) function for the Holt and Winters method (Holt, 2004). In addition, the proposed **nortsTest** package has already been applied in the literature, see Nieto-Reyes (2021) and Nieto-Reyes (2022).

Section 2 provides the theoretical background, including preliminary concepts and results. Section 3 introduces the normality tests for stationary processes, each subsection introducing a test framework and including examples of the tests functions with simulated data. Section 4 provides numerical experiments with simulated data and a real-world application: Subsection 4.1 reports a simulation study for the implemented normality tests and Subsection 4.2 the package's functionality for model checking in a real data application. The *carbon dioxide* data measured in the Malua Loa Observatory (Stoffer, 2020) is analyzed using a state space model from the **forecast** package, evaluating the model's assumptions using our proposed `check_residuals()` function. Section 5 discusses the package functionality and provides our conclusions. Furthermore, we mention our future intended work on the package.

2 Preliminary concepts

This section provides some theoretical aspects of stochastic processes that are a necessary theoretical framework for the following sections. Shumway and Stoffer (2010) and Tsay (2010) give more details of the following definitions and results below.

For the purpose of this work, T is a set of real values denoted as time, $T \subseteq \mathbb{R}$, for instance $T = \mathbb{N}$ or $T = \mathbb{Z}$, the naturals or integer numbers respectively. We denote by $X := \{X_t\}_{t\in T}$ a *stochastic process* with X_t a real random variable for each $t \in T$. Following this notation, a *time-series* is just a finite collection of ordered observations of X (Shumway and Stoffer, 2010). An important measure for a stochastic process is its mean function $\mu(t) := E[X_t]$ for each $t \in T$, where $E[\cdot]$ denotes the usual expected value of a random variable. A generalization of this measure is the k-th order centered moment function $\mu_k(t) := E[(X_t - \mu(t))^k]$ for each $t \in T$ and $k > 1$; with the process variance function being the second order centered moment, $\sigma^2(t) := \mu_2(t)$. Other important measures are the auto-covariance and auto-correlation functions, which measure the linear dependency between two different time points of a given process. For any $t, s \in T$, they are, respectively,

$$\gamma(t,s) := E[(X_t - \mu(t))(X_s - \mu(s))] \text{ and } \rho(t,s) := \frac{\gamma(t,s)}{\sqrt{\mu_2(t)}\sqrt{\mu_2(s)}}.$$

Other widely used measure functions for the analysis of processes are the skewness and kurtosis functions, defined as $s(t) := \mu_3(t)/[\mu_2(t)]^{3/2}$ and $k(t) := \mu_4(t)/[\mu_2(t)]^2$ for each $t \in T$, respectively.

A generally used assumption for stochastic processes is stationarity. It has a key role in forecasting procedures of classic time-series modeling (Tsay, 2010) or as a principal assumption in de-noising methods for signal theory (Wasserman, 2006).

Definition 1 A stochastic process X is said to be *strictly stationary* if, for every collection $\tau = \{t_1, t_2, \ldots, t_k\} \subset T$ and $h > 0$, the joint distribution of $\{X_t\}_{t\in\tau}$ is identical to that of $\{X_{t+h}\}_{t\in\tau}$.

The previous definition is strong for applications. A milder version of it, which makes use of the process' first two moments, is weak stationarity.

Definition 2 A stochastic process X is said to be *weakly stationary* if its mean function is constant in time, $\mu(t) = \mu$, its auto-covariance function only depends on the difference between times, $\gamma(s,t) = \sigma|t-s|$ for a $\sigma \in \mathbb{R}$, and it has a finite variance function, $\mu_2(t) = \mu_2 < \infty$.

For the rest of this work, the term *stationary* will be used to specify a weakly stationary process. A direct consequence of the stationarity assumption is that the previous measure functions get simplified. Thus, given a stationary stochastic process X, its mean function, k-th order centered moment, for $k > 1$, and auto-covariance function are respectively,

$$\mu = E[X_{t_1}],\ \mu_k = E[(X_{t_1} - \mu)^k] \text{ and } \gamma(h) = E[(X_{t_1+h} - \mu)(X_{t_1} - \mu)],$$

which are independent of $t_1 \in T$.

Given a sample $x_1, \ldots, x_n$, $n \in \mathbb{N}$, of equally spaced observations of X, their corresponding estimators, sample mean, sample k-th order centered moment and sample auto-covariance, are respectively

$$\widehat{\mu} := n^{-1}\sum_{i=1}^{n} x_i,\ \widehat{\mu}_k := n^{-1}\sum_{i=1}^{n}(x_i - \widehat{\mu})^k \text{ and } \widehat{\gamma}(h) := n^{-1}\sum_{i=1}^{n-h}(x_{i+h} - \widehat{\mu})(x_i - \widehat{\mu}).$$

A particular case in which stationarity implies strictly stationarity is a Gaussian process.

Definition 3 A stochastic process X is said to be a *Gaussian process* if for every finite collection $\tau = \{t_1, t_2, \ldots, t_k\} \subset T$, the joint distribution of $\{X_t\}_{t\in\tau}$ has a multivariate normal distribution.

A series of mean zero uncorrelated random variables with finite constant variance is known as *white noise*. If additionally, it is formed of independent and identically distributed (i.i.d) normal random variables, it is known as *Gaussian white noise*; which is a particular case of stationary Gaussian process. For the rest of the work, $X_t \sim N(\mu, \sigma^2)$ denotes that the random variable X_t is normally distributed with mean μ and variance σ^2 and $\chi^2(v)$ denotes the Chi square distribution with v degrees of freedom.

Other classes of stochastic processes can be defined using collections of white noise, for instance, the linear process.

Definition 4 Let X be a stochastic process. X is said to be *linear* if it can be written as

$$X_t = \mu + \sum_{i\in\mathbb{Z}} \phi_i \epsilon_{t-i},$$

where $\{\epsilon_i\}_{i\in\mathbb{Z}}$ is a collection of white noise random variables and $\{\phi_i\}_{i\in\mathbb{Z}}$ is a set of real values such that $\sum_{i\in\mathbb{Z}} |\phi_j| < \infty$.

An important class of processes is the *auto-regressive moving average* ($ARMA$). Box and Jenkins (1990) introduced it for time series analysis and forecast, becoming very well-known in the 90s and early 21st century.

Definition 5 For any non-negative integers p, q, a stochastic process X is an $ARMA(p,q)$ process if it is a stationary process and

$$X_t = \sum_{i=0}^{p} \phi_i X_{t-i} + \sum_{i=0}^{q} \theta_i \epsilon_{t-i}, \tag{1}$$

where $\{\phi_i\}_{i=0}^{p}$ and $\{\theta_i\}_{i=0}^{q}$ are sequences of real values with $\phi_0 = 0$, $\phi_p \neq 0$, $\theta_0 = 1$ and $\theta_q \neq 0$ and $\{\epsilon_i\}_{i\in\mathbb{Z}}$ is a collection of white noise random variables.

Particular cases of $ARMA$ processes are those known as auto-regressive ($AR(p) := ARMA(p,0)$) and mean average ($MA(q) := ARMA(0,q)$) processes. Additionally, a *random walk* is a non stationary AR(1) process satisfying (1) with $p = 1$, $\phi_1 = 1$ and $q = 0$. Several properties of an $ARMA$ process can be extracted from its structure. For that, the AR and MA polynomials are introduced

$$AR:\ \phi(z) = 1 - \sum_{i=0}^{p} \phi_i z^i \text{ and } MA:\ \theta(z) = \sum_{i=0}^{q} \theta_i z^i,$$

where z is a complex number and, as before, $\phi_0 = 0$, $\phi_p \neq 0$, $\theta_0 = 1$ and $\theta_q \neq 0$. Conditions for stationarity, order selection and, process behavior are properties studied from these two polynomials.

For modeling volatility in financial data, Bollerslev (1986) proposed the *generalized auto-regressive conditional heteroscedastic* (GARCH) class of processes as a generalization of the *auto-regressive conditional heteroscedastic* (ARCH) processes (Engle, 1982).

Definition 6 For any $p, q \in \mathbb{N}$, a stochastic process X is a $GARCH(p,q)$ process if it satisfies $X_t = \mu + \sigma_t \epsilon_t$ with

$$\sigma_t^2 = \alpha_0 + \sum_{i=1}^{p} \alpha_i \epsilon_{t-i}^2 + \sum_{i=1}^{q} \beta_i \sigma_{t-i}^2.$$

μ is the process mean, σ_0 is a positive constant value, $\{\alpha_i\}_{i=1}^{p}$ and $\{\beta_i\}_{i=1}^{q}$ are non-negative sequences of real values and $\{\epsilon_t\}_{t\in T}$ is a collection of i.i.d. random variables.

A more general class of processes are the *state-space models* ($SSMs$), which have gained popularity over the years because they do not impose on the process common restrictions such as linearity or stationarity and are flexible in incorporating the process different characteristics (Petris et al., 2007). They are widely used for smoothing (West and Harrison, 2006) and forecasting (Hyndman and Khandakar, 2008) in time series analysis. The main idea is to model the process dependency with two equations: the *state equation*, which models how parameters change over time, and the *innovation equation*, which models the process in terms of the parameters. Some particular SSMs that analyze the level, trend and seasonal components of the process are known as *error, trend, and seasonal* (ETS) models. There are over 32 different variations of ETS models (Hyndman et al., 2008). One of them is the *multiplicative error, additive trend-seasonality* ($ETS(M,A,A)$) model.

Definition 7 A SSM process X follows an ETS(M,A,A) model, if the process accepts

$$X_t = [L_{t-1} + T_{t-1} + S_{t-1}](1 + \epsilon_t)$$

as innovation equation and

$$\begin{aligned} L_t &= L_{t-1} + T_{t-1} + \alpha(L_{t-1} + T_{t-1} + S_{t-m})\epsilon_t \\ T_t &= T_{t-1} + \beta(L_{t-1} + T_{t-1} + S_{t-m})\epsilon_t \\ S_t &= S_{t-m} + \gamma(L_{t-1} + T_{t-1} + S_{t-m})\epsilon_t, \end{aligned}$$

as state equations. $\alpha, \beta, \gamma \in [0,1]$, $m \in \mathbb{N}$ denotes the period of the series and $\{\epsilon_t\}$ are i.i.d normal random variables. For each $t \in \mathbb{Z}$, L_t, T_t and S_t represent respectively the level, trend and seasonal components.

3 Normality tests for stationary processes

Extensive literature exists on goodness of fit tests for normality under the assumption of independent and identically distributed random variables, including, among others, Pearson's chi-squared test (Pearson and Henrici, 1895), Kolmogorov-Smirnov test (Smirnov, 1948), Anderson-Darling test (Anderson and Darling, 1952), SK test (Jarque and Bera, 1980) and Shapiro-Wilk test, (Shapiro and Wilk, 1965) and (Royston, 1982). These procedures have been widely used in many studies and applications, see D'Agostino and Stephens (1986) for further details. There are no results, however, showing that the above tests are consistent in the context of stationary processes, in which case the independence assumption is violated. For instance, Gasser (1975) provides a simulation study where Pearson's chi-squared test has an excessive rejection rate under the null hypothesis for dependent data. For this matter, several tests for stationary processes have been proposed over the years. A selection of which we reference here. Epps (1987) provides a test based on the characteristic function, Hinich (1982) proposes a similar test based on the process' spectral density function (Berg et al., 2010, for further insight). Gasser (1975) gives a correction of the SK test, with several modifications made in Lobato and Velasco (2004), Bai and Ng (2005) and Psaradakis (2017), which are popular in many financial applications. Bontemps and Meddahi (2005) constructs a test based on Stein's characterization of a Gaussian distribution. Using the random projection method (Cuesta-Albertos et al., 2007), Nieto-Reyes et al. (2014) build a test that upgrades the performance of Epps (1987) and Lobato and Velasco (2004) procedures. Furthermore, Psaradakis and Vávra (2017) adapts the Anderson and Darling (1952) statistic for stationary linear processes approximating its sample distribution with a sieve bootstrap procedure.

Despite the existing literature, consistent implementations of goodness of fit test for normality of stationary processes in programming languages such as `R` or `Python` are limited. This is not the case for normality of independent data, the **nortest** package (Gross and Ligges, 2015) implements tests such as Lilliefors (Dallal and Wilkinson, 1986), Shapiro-Francia (Royston, 1993), Pearson's chi-squared, Cramer von Misses (Anderson, 1962) and Anderson-Darling. For a multivariate counterpart, the **mvnTest** package (Pya et al., 2016) implements the multivariate Shapiro-Wilk, Anderson-Darling, Cramer von Misses, Royston (Royston, 1992), Doornik and Hansen (Doornik and Hansen, 2008), Henze and Zirkler (Henze and Zirkler, 1990) and the multivariate Chi square test (Vassilly Voinov and Voinov, 2016). For the case of dependent data, we present here the **nortsTest** package. Type within `R` `install.packages("nortsTest", dependencies = TRUE)` to install its latest released version from `CRAN`. **nortsTest** performs the tests proposed in Epps (1987), Lobato and Velasco (2004), Psaradakis and Vávra (2020), Nieto-Reyes et al. (2014), Psaradakis and Vávra (2017) and El Bouch et al. (2022).

Additionally, the package offers visualization functions for descriptive time series analysis and several diagnostic methods for checking stationarity and normality assumptions for the most used time series models of several `R` packages. To elaborate on this, Subsection 3.1 introduces the package functionality and software and Subsection 3.2 provides an overview of tests for checking stationary and seasonality. Finally, Subsections 3.3-3.5 present a general framework of each of the implemented normality tests and their functionality by providing simulated data examples.

3.1 Software

The package works as an extension of the **nortest** package (Gross and Ligges, 2015), which performs normality tests in random samples but for independent data. The building block functions of the **nortsTest** package are:

- `epps.test()`, function that implements the test of Epps,
- `epps_bootstrap.test()`, function that implements a bootstrap approximation of the test of Epps,
- `lobato.test()`, function that implements the asymptotic test of Lobato and Velasco,
- `lobato_bootstrap.test()`, function that implements a bootstrap approximation of the test of Lobato and Velasco,
- `rp.test()`, function that implements the random projection test of Nieto-Reyes, Cuesta-Albertos and Gamboa,
- `vavra.test()`, function that implements the test of Psaradaki and Vavra, and
- `elbouch.test()`, function that implements the test of El Bouch, Michel and Comon.

Each of these functions accepts a `numeric` (*numeric*) or `ts` (*time series*) class object for storing data, and returns a `htest` (*hypothesis test*) class object with the main results for the test. To guarantee the accuracy of the results, each test performs unit root tests for checking stationarity and seasonality (see Subsection 3.2) and displays a warning message if any of them is not satisfied.

For visual diagnostic, the package offers different plot functions based on the **ggplot2** package (Wickham, 2009): the `autoplot()` function plots `numeric`, `ts` and `mts` (*multivariate time series*) classes while the `gghist()` and `ggnorm()` functions are for plotting histogram and qq-plots respectively; and on the **forecast** package (Hyndman and Khandakar, 2008): `ggacf()` and `ggPacf()` for the display of the auto-correlation and partial auto-correlations functions respectively.

Furthermore, inspired in the function `checkresiduals()` of the **forecast** package, we provide the `check_residuals()` function to test the model assumptions using the estimated residuals. The upgrade of our proposal is that, besides providing plots for visual diagnosis (setting the `plot` option as `TRUE`), it does check stationarity, seasonality (*Subsection 3.2*) and normality, presenting a report of the used tests and conclusions for assessing the model's assumptions. An illustration of these functions is provided in Subsection 4.2, where we show the details of the functions and their utility for assumptions commonly checked in time series modeling.

3.2 Tests for stationarity

For checking stationarity, the **nortsTest** package uses *unit root* and *seasonal unit root* tests. These tests work similarly, checking whether a specific process follows a random walk model, which clearly is a non-stationary process.

Unit root tests

A linear stochastic process X that follows a random walk model is non stationary. Its AR polynomial is $\phi(z) = 1 - z$, whose solution (root) is unique and equal to one. Thus, it is common to test the non stationarity of a linear process by checking whether its AR polynomial has a unit root (a root equal to one).

The most commonly used tests for unit root testing are *Augmented Dickey-Fuller* (Said and Dickey, 1984), *Phillips-Perron* (Perron, 1988), *kpps* (Kwiatkowski et al., 1992) and *Ljung-Box* (Box and Pierce, 1970). In particular, the *Ljung-Box* test contrasts the null auto-correlation hypothesis of identically distributed Gaussian random variables, which is equivalent to test stationarity. The `uroot.test()` and `check_residual()` functions perform these tests, making use of the **tseries** package (Trapletti and Hornik, 2019).

Seasonal unit root tests

Let X be a stationary process and m its period. Note that for observed data, m generally corresponds to the number of observations per unit of time. X follows a seasonal random walk if it can be written as

$$X_t = X_{t-m} + \epsilon_t,$$

where ϵ_t is a collection of i.i.d random variables. In a similar way, the process X is non-stationary if it follows a seasonal random walk. Or equivalently, X is non stationary if the seasonal AR(1) polynomial ($\phi_m(z) = 1 - \phi z^m$) has a unit root. The `seasonal.test()` and `check_residuals()` functions perform the *OCSB test* (Osborn et al., 1988) from the **forecast** package and the *HEGY* (Beaulieu and Miron, 1993) and *Ch* (Canova and Hansen, 1995) tests from the **uroot** package (de Lacalle, 2019).

3.3 Tests of Epps

The χ^2 test for normality proposed by Epps (1987) compares the empirical characteristic function of the one-dimensional marginal of the process with the one of a normally distributed random variable evaluated at certain points on the real line. Several authors, including Lobato and Velasco (2004), Psaradakis and Vávra (2017) and El Bouch et al. (2022), point out that the greatest challenge in the Epps' test is its implementation procedure, which we address with the **nortsTest** package. Other existing tests based on the empirical characteristic function of the one-dimensional marginal of the process include Hong (1999) and the references therein. This test differs, however, in that it uses spectral analysis and derivatives.

Furthermore, Meintanis (2016) reviews on testing procedures based on the empirical characteristic function. There, it is commented about the random projection test (Nieto-Reyes et al., 2014, and here below) as a recent development of Epps' test. In fact, in Nieto-Reyes et al. (2014) the consistency of Epps test is improved by taking at random the elements at which the characteristic function is evaluated. Additionally, El Bouch et al. (2022) proposes a sieve bootstrap modification of the Epps' test. In addition to the classical asymptotic Epps' test, we include these last two approaches here, and

in the package, see the Example below and the paragraph before it. Let us provide now the foundation behind the Epps' tests.

Let X be a stationary stochastic process that satisfies

$$\sum_{t=-\infty}^{\infty} |t|^k |\gamma(t)| < \infty \text{ for some } k > 0. \tag{2}$$

The null hypothesis is that the one-dimensional marginal distribution of X is a Gaussian process. The procedure for constructing the test consists of defining a function g, estimating its inverse spectral matrix function, minimizing the generated quadratic function in terms of the unknown parameters of the random variable and, finally, obtaining the test statistic, which converges in distribution to a χ^2.

Given $N \in \mathbb{N}$ with $N \geq 2$, let

$$\Lambda := \{\lambda := (\lambda_1, \ldots, \lambda_N) \in \mathbb{R}^N : \lambda_i \leq \lambda_{i+1} \text{ and } \lambda_i > 0, \text{ for } i = 1, 2, \ldots, N\},$$

and $g : \mathbb{R} \times \Lambda \to \mathbb{R}^n$ be a measurable function, where

$$g(x, \lambda) := [\cos(\lambda_1 x), \sin(\lambda_1 x), \ldots, \cos(\lambda_N x), \sin(\lambda_N x)].$$

Additionally, let $g_\theta : \Lambda \to \mathbb{R}^N$ be a function defined by

$$g_\theta(\lambda) := [\mathrm{Re}(\Phi_\theta(\lambda_1)), \mathrm{Im}(\Phi_\theta(\lambda_1)), \ldots, \mathrm{Re}(\Phi_\theta(\lambda_N)), \mathrm{Im}(\Phi_\theta(\lambda_N))]^t,$$

where the $\mathrm{Re}(\cdot)$ and $\mathrm{Im}(\cdot)$ are the real and imaginary components of a complex number and Φ_θ is the characteristic function of a normal random variable with parameters $\theta := (\mu, \sigma^2) \in \Theta$, an open bounded set contained in $\mathbb{R} \times \mathbb{R}^+$. For any $\lambda \in \Lambda$, let us also denote

$$\widehat{g}(\lambda) := \frac{1}{n} \sum_{t=1}^{n} [\cos(\lambda_1 x_t), \sin(\lambda_1 x_t), \ldots, \cos(\lambda_N x_t), \sin(\lambda_N x_t)]^t.$$

Let $f(v; \theta, \lambda)$ be the spectral density matrix of $\{g(X_t, \lambda)\}_{t \in \mathbb{Z}}$ at a frequency v. Then, for $v = 0$, it can be estimated by

$$\widehat{f}(0; \theta, \lambda) := \frac{1}{2\pi n} \left(\sum_{t=1}^{n} \widehat{G}(x_{t,0}, \lambda) + 2 \sum_{i=1}^{\lfloor n^{2/5} \rfloor} (1 - i/\lfloor n^{2/5} \rfloor) \sum_{t=1}^{n-i} \widehat{G}(x_{t,i}, \lambda) \right),$$

where $\widehat{G}(x_{t,i}, \lambda) = (\widehat{g}(\lambda) - g(x_t, \lambda))(\widehat{g}(\lambda) - g(x_{t+i}, \lambda))^t$ and $\lfloor \cdot \rfloor$ denotes the floor function. The test statistic general form under H_0 is

$$Q_n(\lambda) := \min_{\theta \in \Theta} \{Q_n(\theta, \lambda)\},$$

with

$$Q_n(\theta, \lambda) := (\widehat{g}(\lambda) - g_\theta(\lambda))^t G_n^+(\lambda)(\widehat{g}(\lambda) - g_\theta(\lambda)),$$

where G_n^+ is the generalized inverse of the spectral density matrix $2\pi \widehat{f}(0; \theta, \lambda)$. Let

$$\widehat{\theta} := \arg\min_{\theta \in \Theta} \{Q_n(\theta, \lambda)\},$$

be the argument that minimizes $Q_n(\theta, \lambda)$ such that $\widehat{\theta}$ is in a neighborhood of $\widehat{\theta}_n := (\widehat{\mu}, \widehat{\gamma}(0))$. To guarantee its' existence and uniqueness, the following assumptions are required. We refer to them as assumption $(A.)$.

$(A.)$ Let θ_0 be the true value of θ under H_0, then for every $\lambda \in \Lambda$ the following conditions are satisfied.

- $f(0; \theta, \lambda)$ is positive definite.
- $\Phi_\theta(\lambda)$ is twice differential with respect to θ in a neighborhood of θ_0.
- The matrix $D(\theta_0, \lambda) = \dfrac{\partial \Phi_\theta(\lambda)}{\partial \theta|_{\theta=\theta_0}} \in \mathbb{R}^{N \times 2}$ has rank 2.
- The set $\Theta_0(\lambda) := \{\theta \in \Theta : \Phi_\theta(\lambda_i) = \Phi_{\theta_0}(\lambda_i), i = 1, \ldots, N\}$ is a finite bounded set in Θ. And θ is a bounded subset $\mathbb{R} \times \mathbb{R}^+$.
- $f(0; \theta, \lambda) = f(0; \theta_0, \lambda)$ and $D(\theta_0, \lambda) = D(\theta, \lambda)$ for all $\theta \in \Theta_0(\lambda)$.

Under these assumptions, the Epps's main result is presented as follows.

Theorem 1 (Epps, 1987, Theorem 2.1) Let X be a stationary Gaussian process such that (2) and $(A.)$ are satisfied, then $nQ_n(\lambda) \to_d \chi^2(2N-2)$ for every $\lambda \in \Lambda$.

The current **nortsTest** version, uses $\Lambda := \{\texttt{lambda}/\widehat{\gamma}(0)\}$ as the values to evaluate the empirical characteristic function, where $\widehat{\gamma}(0)$ is the sample variance. By default `lambda = c(1, 2)`. Therefore, the implemented test statistic converges to a χ^2 distribution with two degrees of freedom. The user can change these Λ values as desired by simply specifying the function's `lambda` argument, as we show in the Example below.

Example 1 A stationary $AR(2)$ process is drawn using a beta distribution with `shape1 = 9` and `shape2 = 1` parameters, and performed the implementation of the test of Epps, `epps.test()`. At significance level $\alpha = 0.05$, the null hypothesis of normality is correctly rejected.

```
set.seed(298)
x = arima.sim(250,model = list(ar =c(0.5,0.2)),
                 rand.gen = rbeta,shape1 = 9,shape2 = 1)

# Asymptotic Epps test
epps.test(x)
#>
#>  Epps test
#>
#> data:  x
#> epps = 22.576, df = 2, p-value = 1.252e-05
#> alternative hypothesis: x does not follow a Gaussian Process
```

Asymptotic Epps test with random Lambda values as proposed in Nieto-Reyes et al. (2014).

```
set.seed(298)
epps.test(x, lambda = abs(rnorm(mean = c(1, 2), 2)))
#>
#>  Epps test
#>
#> data:  x
#> epps = 25.898, df = 2, p-value = 2.379e-06
#> alternative hypothesis: x does not follow a Gaussian Process
```

Approximated sieve bootstrap Epps test using 1000 repetitions of 250 units.

```
set.seed(298)
epps_bootstrap.test(x, seed = 298)
#>
#>  Sieve-Bootstrap epps test
#>
#> data:  y
#> bootstrap-epps = 22.576, p-value < 2.2e-16
#> alternative hypothesis: y does not follow a Gaussian Process
```

3.4 Tests of Lobato and Velasco

Lobato and Velasco (2004) provides a consistent estimator for the corrected SK test statistic for stationary processes, see Lomnicki (1961) and Gasser (1975) for further insight. Note that the SK test is also known as the Jarque-Bera test (Jarque and Bera, 1980), which is already available in several R packages (Trapletti and Hornik, 2019, for instance). The improvement of this proposal over those implementations is a correction in the skewness and kurtosis estimates by the process' auto-covariance function, resulting in a consistent test statistic under the assumption of correlated data. The test in Lobato and Velasco (2004) is asymptotic, which is computationally efficient, as opposed to a bootstrap based test. Psaradakis and Vávra (2020) show that the bootstrap modification of the Lobato and Velasco's test is a fair competitor against the original asymptotic test, beating other tests for normality of the one-dimensional marginal distribution in terms of power. Thus, the package incorporates both the asymptotic, `lobato.test()` and its bootstrap version `lobato_bootstrap.test()`.

The general framework for the test is presented in what follows. On the contrary to the test of Epps, this proposal does not require additional parameters for the computation of the test sample statistic.

Let X be a stationary stochastic process that satisfies

$$\sum_{t=0}^{\infty} |\gamma(t)| < \infty. \tag{3}$$

The null hypothesis is that the one-dimensional marginal distribution of X is normally distributed, that is

$$H_0 : X_t \sim N(\mu, \sigma^2) \text{ for all } t \in \mathbb{R}.$$

Let $k_q(j_1, j_2, \ldots, j_{q-1})$ be the q-th order cummulant of $X_1, X_{1+j_1}, \ldots, X_{1+j_{q-1}}$. H_0 is fulfilled if all the marginal cummulants above the second order are zero. In practice, it is tested just for the third and fourth order marginal cummulants. Equivalently, in terms of moments, the marginal distribution is normal by testing whether $\mu_3 = 0$ and $\mu_4 = 3\mu_2^2$. For non-correlated data, the SK test compares the SK statistic against upper critical values from a $\chi^2(2)$ distribution (Bai and Ng, 2005). For a Gaussian process X satisfying (3), it holds the limiting result

$$\sqrt{n}\begin{pmatrix} \widehat{\mu}_3 \\ \widehat{\mu}_4 - 3\widehat{\mu}_2^2 \end{pmatrix} \to_d N[0_2, \Sigma_F)],$$

where $0_2 := (0,0)^t \in \mathbb{R}^2$ and $\Sigma_F := \text{diag}(6F^{(3)}, 24F^{(4)}) \in \mathbb{R}^{2x2}$ is a diagonal matrix with $F^{(k)} := \sum_{j=-\infty}^{\infty} \gamma(j)^k$ for $k = 3, 4$ (Gasser, 1975).

The following consistent estimator in terms of the auto-covariance function is proposed in Lobato and Velasco (2004)

$$\widehat{F}^{(k)} := \sum_{t=1-n}^{n-1} \widehat{\gamma}(t)[\widehat{\gamma}(t) + \widehat{\gamma}(n - |t|)]^{k-1},$$

to build a *generalized SK test* statistic

$$G := \frac{n\widehat{\mu}_3^2}{6\widehat{F}^{(3)}} + \frac{n(\widehat{\mu}_4 - 3\widehat{\mu}_2)^2}{24\widehat{F}^{(4)}}.$$

Similar to the SK test for non-correlated data, the G statistic is compared against upper critical values from a $\chi^2(2)$ distribution. This is seen in the below result that establishes the asymptotic properties of the test statistics, so that the general test procedure can be constructed. The result requires the following assumptions, denoted by $(B.)$, for the process X.

(B.)

- $E[X_t^{16}] < \infty$ for $t \in T$.
- $\sum_{j_1=-\infty}^{\infty} \cdots \sum_{j_{q-1}=-\infty}^{\infty} |k_q(j_1, \ldots, j_{q-1})| < \infty$ for $q = 2, 3, \ldots, 16$.
- $\sum_{j=1}^{\infty} \left(E\left[E[(X_0 - \mu)^k | B_j] - \mu_k \right]^2 \right)^{1/2} < \infty$ for $k = 3, 4$, where B_j denotes the σ-field generated by X_t, $t \leq -j$.
- $E\left[Z_k\right]^2 + 2\sum_{j=1}^{\infty} E\left([Z_k] \left[(X_j - \mu)^k - \mu_k \right] \right) > 0$ for $k = 3, 4$, with $Z_k = (X_0 - \mu)^k - \mu_k$.

Note that these assumptions imply that the higher-order spectral densities up to order 16 are continuous and bounded.

Theorem 2 (Lobato and Velasco, 2004, Theorem 1) Let X be a stationary process. If X is Gaussian and satisfies (3) then $G \to_d \chi^2(2)$, and under assumption (B.), the test statistic G diverges whenever $\mu_3 \neq 0$ or $\mu_4 \neq 3\mu_2^2$.

Example 2 A stationary $MA(3)$ process is drawn using a gamma distribution with `rate = 3` and `shape = 6` parameters. The `lobato.test()` function performs the test of *Lobato and Velasco* to the simulated data. At significance level $\alpha = 0.05$, the null hypothesis of normality is correctly rejected.

```
set.seed(298)
x = arima.sim(250,model = list(ma = c(0.2, 0.3, -0.4)),
                rand.gen = rgamma, rate = 3, shape = 6)
# Asymptotic Lobato & Velasco
lobato.test(x)
#>
#>  Lobato and Velasco's test
```

```
#>
#> data:  x
#> lobato = 65.969, df = 2, p-value = 4.731e-15
#> alternative hypothesis: x does not follow a Gaussian Process
```

Approximated sieve bootstrap Lobato and Velasco test using 1000 repetitions of 250 units.

```
lobato_bootstrap.test(x, seed = 298)
#>
#>  Sieve-Bootstrap lobato test
#>
#> data:  y
#> bootstrap-lobato = 65.969, p-value < 2.2e-16
#> alternative hypothesis: y does not follow a Gaussian Process
```

3.5 The Random Projections test

The previous proposals only test for the normality of the one-dimensional marginal distribution of the process, which is inconsistent against alternatives whose one-dimensional marginal is Gaussian. Nieto-Reyes et al. (2014) provides a procedure to fully test normality of a stationary process using a Crammér-Wold type result (Cuesta-Albertos et al., 2007) that uses random projections to differentiate among distributions. In Nieto-Reyes et al. (2014) existing tests for the normality of the one dimensional marginal are applied to the random projections and the resulting p-values combined using the false discovery rate for dependent data (Benjamini and Yekutieli, 2001). The **nortsTest** package improves on this test by allowing to use the less conservative false discovery rate in Benjamini and Hochberg (1995).

We show the Crammér-Wold type result below. The result works for separable Hilbert spaces, however here, for its later application, we restrict it to l^2, the space of square summable sequences over $\mathbb{N}$, with inner product $\langle \cdot, \cdot \rangle$.

Theorem 3 (Cuesta-Albertos et al., 2007, Theorem 3.6) Let η be a dissipative distribution on l^2 and Z a l^2-valued random element, then Z is Gaussian if and only if

$$\eta\{h \in l^2 : \langle Z, h\rangle \text{ has a Gaussian distribution}\} > 0.$$

A dissipative distribution (Nieto-Reyes et al., 2014, Definition 2.1) is a generalization of the concept of absolutely continuous distribution to the infinite-dimensional space. A Dirichlet process (Gelman et al., 2013) produces random elements with a dissipative distribution in l^2. In practice, generate draws of $h \in l^2$ with a stick-breaking process that makes use of beta distributions.

Let $X = \{X_t\}_{t\in\mathbb{Z}}$ be a stationary process. As X is normally distributed if the process $X^{(t)} := \{X_k\}_{k\leq t}$ is Gaussian for each $t \in \mathbb{Z}$, using the result above, Nieto-Reyes et al. (2014) provides a procedure for testing that X is a Gaussian process by testing whether the process $Y^h = \{Y_t^h\}_{t\in\mathbb{Z}}$ is Gaussian.

$$Y_t^h := \sum_{i=0}^{\infty} h_i X_{t-i} = \langle X^{(t)}, h\rangle, \tag{4}$$

where $\langle X^{(t)}, h\rangle$ is a real random variable for each $t \in \mathbb{Z}$ and $h \in l^2$. Thus, Y^h is a stationary process constructed by the projection of $X^{(t)}$ on the space generated by h. Therefore, X is a Gaussian process if and only if the one dimensional marginal distribution of Y^h is normally distributed. Additionally, the hypothesis of the tests *Lobato and Velasco* or *Epps*, such as (2), (3), (A) and (B), imposed on X are inherited by Y^h. Then, those tests can be applied to evaluate the normality of the one dimensional marginal distribution of Y^h. Further considerations include the specific beta parameters used to construct the distribution from which to draw h and selecting a proper number of combinations to establish the number of projections required to improve the method performance. All of these details are discussed in Nieto-Reyes et al. (2014).

Next, we summarize the test of random projections in practice:

1. Select k, which results in $2k$ independent random projections (*by default* `k = 1`).
2. Draw the $2k$ random elements to project the process from a dissipative distribution that uses a particular beta distribution. By default, use a $\beta(2,7)$ for the first k projections and a $\beta(100,1)$ for the later k.

3. Apply the tests of *Lobato and Velasco* to the even projected processes and *Epps* to the odd projections.
4. Combine the obtained $2k$ `p-values` using the false discover rate. By default, use Benjamini and Yekutieli (2001) procedure.

The `rp.test()` function implements the above procedure. The user might provide optional parameters such as the number of projections `k`, the parameters of the first beta distribution `pars1` and those of the second `pars2`. The next example illustrates the application of the `rp.test()` to a stationary GARCH(1,1) process drawn using normal random variables.

Example 3 A stationary `GARCH(1,1)` process is drawn with a standard normal distribution and parameters $\alpha_0 = 0$, $\alpha_1 = 0.2$ and $\beta_1 = 0.3$ using the (**fGarch** package, Wuertz et al., 2017). Note that a `GARCH(1,1)` process is stationary if the parameters α_1 and β_1 satisfy the inequality $\alpha_1 + \beta_1 < 1$ (Bollerslev, 1986).

```
set.seed(3468)
library(fGarch)
spec = garchSpec(model = list(alpha = 0.2, beta = 0.3))
x = ts(garchSim(spec, n = 300))
rp.test(x)
#>
#>  k random projections test.
#>
#> data:  x
#> k = 1, p.value adjust = Benjamini & Yekutieli, p-value = 1
#> alternative hypothesis: x does not follow a Gaussian Process
```

At significance level $\alpha = 0.05$, the applied *random projections* test with `k = 1` as the number of projections shows no evidence to reject the null hypothesis of normality.

3.6 The Psaradakis and Vavra's test

Psaradakis and Vávra (2017) adapted a distance test for normality for a one-dimensional marginal distribution of a stationary process. Initially, the test was based on the Anderson (1952) test statistic and used an auto-regressive sieve bootstrap approximation to the null distribution of the sample test statistic. Later, Psaradakis and Vávra (2020) considered this test as the ultimate normality test based on the empirical distribution function, and adapted its methodology to a wide range of tests, including Shapiro-Wilk (Shapiro and Wilk, 1965), Jarque-Bera (Jarque and Bera, 1980), Cramer von Mises (Anderson, 1962), Epps, and Lobato-Velasco. Their experiments show that the Lobato-Velasco and Jarque-Bera test's bootstrap version performs best in small samples.

Although the test is said to be applicable to a wide class of non-stationary processes by transforming them into stationary by means of a fractional difference operator, no theoretic result was apparently provided to sustain this transformation. This work restricts the presentation of the original procedure to stationary processes.

Let X be a stationary process satisfying

$$X_t = \sum_{i=0}^{\infty} \theta_i \epsilon_{t-i} + \mu_0, \; t \in \mathbb{Z}, \tag{5}$$

where $\mu_0 \in \mathbb{R}$, $\{\theta_i\}_{i=0}^{\infty} \in l^2$ with $\theta_0 = 1$ and $\{\epsilon_t\}_{i=0}^{\infty}$ is a collection of mean zero i.i.d random variables. The null hypothesis is that the one dimensional marginal distribution of X is normally distributed,

$$H_0 : F(\mu_0 + \sqrt{\gamma(0)}x) - F_N(x) = 0, \text{ for all } x \in \mathbb{R},$$

where F is the cumulative distribution function of X_0, and F_N denotes the standard normal cumulative distribution function. Note that if ϵ_0 is normally distributed, then the null hypothesis is satisfied. Conversely, if the null hypothesis is satisfied, then ϵ_0 is normally distributed and, consequently, X_0. The considered test for H_0 is based on the Anderson-Darling distance statistic

$$A_d = \int_{-\infty}^{\infty} \frac{[F_n(\widehat{\mu} + \sqrt{\widehat{\gamma}(0)}x) - F_N(x)]^2}{F_N(x)[1 - F_N(x)]} dF_N(x), \tag{6}$$

where $F_n(\cdot)$ is the empirical distribution function associated to F based on a simple random sample of size n. Psaradakis and Vávra (2017) proposes an auto-regressive sieve bootstrap procedure to approximate the sampling properties of A_d arguing that making use of classical asymptotic inference for A_d is problematic and involved. This scheme is motivated by the fact that under some assumptions for X, including (5), ϵ_t admits the representation

$$\epsilon_t = \sum_{i=1}^{\infty} \phi_i (X_{t-i} - \mu_0),\ t \in \mathbb{Z}, \tag{7}$$

for certain type of $\{\phi_i\}_{i=1}^{\infty} \in l^2$. The main idea behind this approach is to generate a bootstrap sample ϵ_t^* to approximate ϵ_t with a finite-order auto-regressive model. This is because the distribution of the processes ϵ_t and ϵ_t^* coincide asymptotically if the order of the auto-regressive approximation grows simultaneously with n at an appropriate rate (Bühlmann, 1997). The procedure makes use of the $\epsilon_t^{*'}s$ to obtain the $X_t^{*'}s$ through the bootstrap analog of (7). Then, generate a bootstrap sample of the A_d statistic, A_d^*, making use of the bootstrap analog of (5).

The `vavra.test()` function implements Psaradakis and Vávra (2020) procedure. By default, it generates 1,000 sieve-bootstrap replications of the Anderson-Darling statistic. The user can provide different test procedures, such as the *Shapiro-Wilk, Jarque-Bera, Cramer von Mises, Epps* or *Lobato-Velasco* test, by specifying a text value to the `normality` argument. The presented values are Monte Carlo estimates of the A_d statistic and `p.value`.

Example 4 A stationary $ARMA(1,1)$ process is simulated using a standard normal distribution and performs *Psaradakis and Vávra* procedure using Anderson-Darling and Cramer von Mises test statistics. At significance level $\alpha = 0.05$, there is no evidence to reject the null hypothesis of normality.

```
set.seed(298)
x = arima.sim(250,model = list(ar = 0.2, ma = 0.34))
# Default, Psaradakis and Vavra's procedure
vavra.test(x, seed = 298)
#>
#>  Psaradakis-Vavra test
#>
#> data:  x
#> bootstrap-ad = 0.48093, p-value = 0.274
#> alternative hypothesis: x does not follow a Gaussian Process
```

Approximate Cramer von Mises test for the Psaradakis and Vavra's procedure

```
vavra.test(x, normality = "cvm", seed = 298)
#>
#>  Sieve-Bootstrap cvm test
#>
#> data:  x
#> bootstrap-cvm = 0.056895, p-value = 0.49
#> alternative hypothesis: x does not follow a Gaussian Process
```

3.7 The multivariate kurtosis test

The literature contains some procedures to test the null hypothesis that a multivariate stochastic process is Gaussian. Those include Moulines et al. (1992), a test based on the characteristic function, and Steinberg and Zeitouni (1992), a test based on properties of the entropy of Gaussian processes that does not make use of cumulant computations. According to El Bouch et al. (2022), these tests may hardly be executable in real time. Consequently, they propose a test based on multivariate kurtosis (Mardia, 1970). The proposed procedure is for $p = 1, 2$, and we elaborate on it in what follows. In Section 6.3 of El Bouch et al. (2022), they suggest to apply random projections for higher dimensions but they do not investigate the procedure any further.

The p-value of this test is obtained as $2(1 - F_N(z))$ where, as above, F_N denotes the standard normal cumulative distribution function. There,

$$z := (\hat{B}_p - E[\hat{B}_p]) / \sqrt{E[(\hat{B}_p - E[\hat{B}_p])^2]},$$

where

$$\hat{B}_p := n^{-1} \sum_{t=1}^{n} (x_t^t \hat{S}^{-1} x_t)^2,$$

and

$$\hat{S} := n^{-1} \sum_{t=1}^{n} x_t x_t^t.$$

In El Bouch et al. (2022), there reader can found the exact computations of $E[\hat{B}_p]$ and $E[(\hat{B}_p - E[\hat{B}_p])^2]$.

This test is implemented in the `elbouch.test()` function. By default, the function computes the univariate El Bouch test. If the user provides a secondary data set, the function computes the bivariate counterpart.

Example 5 Simulate a two-dimensional stationary VAR(2) process using independent AR(1) and AR(2) processes with standard normal distributions and apply the bivariate El Bouch test. At significance level $\alpha = 0.05$, there is no evidence to reject the null hypothesis of normality.

```
set.seed(23890)
x = arima.sim(250,model = list(ar = 0.2))
y = arima.sim(250,model = list(ar = c(0.4,0,.1)))
elbouch.test(y = y,x = x)
#>
#>  El Bouch, Michel & Comon's test
#>
#> data:  w = (y, x)
#> Z = 0.92978, p-value = 0.1762
#> alternative hypothesis: w = (y, x) does not follow a Gaussian Process
```

4 Simulations and data analysis

4.1 Numerical experiments

Inspired by the simulation studies in Psaradakis and Vávra (2017) and Nieto-Reyes et al. (2014), we propose here a procedure that involves drawing data from the $AR(1)$ process

$$X_t = \phi X_{t-1} + \epsilon_t,\ t \in \mathbb{Z},\ \text{for } \phi \in \{0, \pm 0.25, \pm 0.4\}, \tag{8}$$

where the $\{\epsilon_t\}_{t\in\mathbb{Z}}$ are i.i.d random variables. For the distribution of the ϵ_t we consider different scenarios: standard normal (N), standard log-normal ($\log N$), Student t with 3 degrees of freedom (t_3), chi-squared with 10 degrees of freedom ($\chi^2(10)$) and gamma with $(7,1)$ shape and scale parameters ($\Gamma(7,1)$).

As in Psaradakis and Vávra (2017), $m = 1,000$ independent draws of the above process are generated for each pair of parameter ϕ and distribution. Each draw is taken of length $past + n$, with $past = 500$ and $n \in \{100, 250, 500, 1000\}$. The first 500 data points of each realization are then discarded in order to eliminate start-up effects. The n remaining data points are used to compute the value of the test statistic of interest. In each particular scenario, the rejection rate is obtained by computing the proportion of times that the test is rejected among the m trials.

Tables 1 and 2 present the rejection rate estimates. For every process of length n, the columns represent the used $AR(1)$ parameter and the rows the distribution used to draw the process. The obtained results are consistent with those obtained in the publications where the different tests were proposed. As expected, rejection rates are around 0.05 when the data is drawn from a standard normal distribution, as in this case the data is drawn from a Gaussian process. Conversely, high rejection rates are registered for the other distributions. Low rejection rates are observed, however, for the $\chi^2(10)$ distribution when making use of some of the tests. For instance, the *Epps* and *bootstrap Epps* tests, although they consistently tend to 1 when the length of the process, n, increases. Another case is the El Bouch test. However, this one maintains low rates for large values of $|\phi|$ when n increases. Furthermore, for the random projections test, the number of projections used in this study is the default $k = 1$, which is by far a lower number than the recommended by Nieto-Reyes et al. (2014). However, even in these conditions, the obtained results are satisfactory, with the random projection test having even better performance than the tests of Epps (1987) or Psaradakis and Vávra (2017).

An important aspect in selecting a procedure is its computation time. Thus, for each length of the process, n, there is an additional column, max.phi, in *Tables* 1 and 2. Each entry in this column refers to

Table 1: Part 1. Rejection rate estimates over $m = 1,000$ trials of the seven studied goodness of fit test for the null hypothesis of normality. The data is drawn using the process defined in (8) for different values of *phi* and *n* displayed in the columns and different distributions for $epsilon_t$ in the rows. *phi* in 0, 0.25, 0.4, n in 100, 250. For each test and distribution, max.phi represents the maximum rejection rate's running time in seconds among the different values of the AR parameter.

	n = 100						n = 250					
phi	-0.4	-0.25	0.0	0.25	0.4	max.phi	-0.4	-0.25	0.0	0.25	0.4	max.phi
Lobato and Velasco												
N	0.041	0.044	0.047	0.032	0.035	0.769	0.059	0.037	0.054	0.040	0.037	0.646
logN	1.000	1.000	1.000	1.000	1.000	0.610	1.000	1.000	1.000	1.000	1.000	0.653
t3	0.797	0.853	0.902	0.875	0.829	0.627	0.990	0.994	0.998	0.999	0.983	0.674
chisq10	0.494	0.698	0.770	0.707	0.610	0.620	0.930	0.995	0.998	0.997	0.977	0.657
Gamma(7,1)	0.995	1.000	0.999	0.996	0.988	0.634	1.000	1.000	1.000	1.000	1.000	0.665
Epps												
N	0.056	0.051	0.062	0.060	0.063	0.695	0.048	0.058	0.053	0.066	0.063	0.736
logN	0.908	0.917	0.972	0.985	0.984	0.729	1.000	1.000	1.000	0.999	1.000	0.777
t3	0.243	0.291	0.370	0.317	0.248	0.722	0.776	0.872	0.908	0.881	0.780	0.769
chisq10	0.267	0.440	0.548	0.469	0.360	0.699	0.611	0.850	0.930	0.866	0.721	0.739
Gamma(7,1)	0.866	0.961	0.996	0.993	0.965	0.722	1.000	1.000	1.000	1.000	1.000	0.782
Random Projections												
N	0.051	0.042	0.045	0.039	0.050	1.301	0.045	0.033	0.046	0.038	0.050	1.905
logN	1.000	1.000	1.000	1.000	1.000	1.330	1.000	1.000	1.000	1.000	1.000	1.906
t3	0.790	0.863	0.879	0.823	0.727	1.320	0.982	0.994	0.995	0.991	0.975	1.949
chisq10	0.589	0.730	0.757	0.640	0.542	1.295	0.957	0.994	0.994	0.969	0.888	1.926
Gamma(7,1)	0.998	1.000	1.000	0.998	0.989	1.308	1.000	1.000	1.000	1.000	1.000	1.963
Psaradakis and Vavra												
N	0.052	0.048	0.051	0.058	0.050	17.905	0.061	0.046	0.038	0.051	0.045	22.115
logN	1.000	1.000	1.000	1.000	1.000	17.149	1.000	1.000	1.000	1.000	1.000	21.841
t3	0.700	0.799	0.851	0.780	0.695	17.503	0.960	0.979	0.991	0.977	0.960	22.183
chisq10	0.498	0.673	0.804	0.689	0.550	18.029	0.902	0.983	0.997	0.988	0.933	22.197
Gamma(7,1)	0.989	1.000	1.000	1.000	0.998	18.467	1.000	1.000	1.000	1.000	1.000	22.292
Bootstrap Lobato												
N	0.057	0.052	0.047	0.059	0.052	37.141	0.035	0.049	0.048	0.058	0.049	40.532
logN	1.000	1.000	1.000	1.000	1.000	32.509	1.000	1.000	1.000	1.000	1.000	40.793
t3	0.797	0.867	0.899	0.869	0.809	32.755	0.989	0.994	0.996	0.996	0.989	41.158
chisq10	0.567	0.729	0.801	0.745	0.649	32.242	0.942	0.990	1.000	0.994	0.963	40.950
Gamma(7,1)	0.999	1.000	1.000	0.998	0.991	31.763	1.000	1.000	1.000	1.000	1.000	41.277
Bootstrap Epps												
N	0.047	0.053	0.048	0.052	0.044	57.749	0.058	0.052	0.053	0.048	0.043	65.367
logN	0.846	0.877	0.963	0.974	0.959	56.756	1.000	1.000	1.000	1.000	0.999	65.968
t3	0.183	0.238	0.313	0.230	0.196	57.350	0.752	0.863	0.913	0.841	0.754	65.699
chisq10	0.252	0.364	0.527	0.450	0.358	56.627	0.596	0.813	0.913	0.854	0.685	65.369
Gamma(7,1)	0.816	0.948	0.993	0.979	0.931	56.986	1.000	1.000	1.000	1.000	1.000	65.315
El Bouch												
N	0.040	0.047	0.044	0.033	0.050	0.798	0.040	0.054	0.052	0.061	0.059	1.020
logN	0.990	0.998	0.998	0.995	0.980	0.805	1.000	1.000	1.000	1.000	1.000	1.025
t3	0.833	0.883	0.928	0.886	0.846	0.824	0.996	0.999	0.998	0.998	0.991	1.044
chisq10	0.041	0.152	0.281	0.155	0.046	0.812	0.062	0.386	0.597	0.388	0.065	1.031
Gamma(7,1)	0.833	0.905	0.929	0.898	0.818	0.818	0.993	0.998	0.999	0.995	0.989	1.042

a different distribution and contains the maximum running time in seconds to obtain the rejection rate among the different values of the AR parameter. That is, for a fix distribution, the rejection rates are computed for each of the five possibilities of ϕ and the time that it takes recorded. The running time in the table is the largest among the five. Furthermore, in *Table* 3 we show the time in seconds that each studied test takes to check whether a given process is Gaussian. In particular, the table contains the average running time over 1,000 trials that takes to generate and check a Gaussian AR(1) process with parameter $\phi = 0.5$. This is done for different sample sizes, $n \in \{1000, 2000, 3000, 4000, 5000\}$. According to the table, the asymptotic tests (Lobato and Velasco, Epps, random projections and El Bouch) have similar running times. On the contrary, the bootstrap based tests (Psaradakis and Vavra, Bootstrap Epps and Lobato and Velasco) have, as expected, higher running times on average. Furthermore, Tables 1 and 2 show similar results in time performance. There, the maximum running time of the bootstrap based tests exceeds in more than ten seconds the time obtained with the asymptotic based tests. It is worth saying that the tables have been obtained with R version 4.3.1 (2023-06-16) and platform aarch64-apple-darwin20 (64-bit),running under macOS Sonoma 14.2.1.

Table 2: Part 2. Rejection rate estimates over $m = 1,000$ trials of the seven studied goodness of fit test for the null hypothesis of normality. The data is drawn using the process defined in (8) for different values of *phi* and *n* displayed in the columns and different distributions for $epsilon_t$ in the rows. *phi* is in 0, 0.25, 0.4 and n in 500, 1000. For each test and distribution, max.phi represents the maximum rejection rate's running time in seconds among the different values of the AR parameter.

	n = 500						n = 1,000					
phi	-0.4	-0.25	0.0	0.25	0.4	max.phi	-0.4	-0.25	0.0	0.25	0.4	max.phi
Lobato and Velasco												
N	0.041	0.035	0.052	0.035	0.049	0.729	0.048	0.050	0.040	0.062	0.040	1.065
logN	1.000	1.000	1.000	1.000	1.000	0.743	1.000	1.000	1.000	1.000	1.000	1.076
t3	1.000	1.000	1.000	1.000	1.000	0.844	1.000	1.000	1.000	1.000	1.000	1.116
chisq10	0.999	1.000	1.000	1.000	1.000	0.824	1.000	1.000	1.000	1.000	1.000	1.082
Gamma(7,1)	1.000	1.000	1.000	1.000	1.000	0.825	1.000	1.000	1.000	1.000	1.000	1.105
Epps												
N	0.048	0.046	0.056	0.065	0.050	0.905	0.034	0.038	0.046	0.033	0.059	1.182
logN	1.000	1.000	1.000	1.000	1.000	0.931	1.000	1.000	1.000	1.000	1.000	1.294
t3	0.991	0.994	0.996	0.997	0.985	0.936	1.000	0.998	1.000	1.000	0.999	1.235
chisq10	0.924	0.991	0.999	0.991	0.969	0.917	0.997	1.000	1.000	1.000	1.000	1.202
Gamma(7,1)	1.000	1.000	1.000	1.000	1.000	0.873	1.000	1.000	1.000	1.000	1.000	1.239
Random Projections												
N	0.044	0.043	0.040	0.040	0.048	2.723	0.021	0.027	0.043	0.043	0.047	4.544
logN	1.000	1.000	1.000	1.000	1.000	2.759	1.000	1.000	1.000	1.000	1.000	4.588
t3	1.000	1.000	1.000	1.000	1.000	2.755	1.000	1.000	1.000	1.000	1.000	4.531
chisq10	1.000	1.000	1.000	1.000	0.998	2.782	1.000	1.000	1.000	1.000	1.000	4.520
Gamma(7,1)	1.000	1.000	1.000	1.000	1.000	2.843	1.000	1.000	1.000	1.000	1.000	4.527
Psaradakis and Vavra												
N	0.048	0.050	0.045	0.053	0.039	26.957	0.055	0.045	0.047	0.043	0.033	37.993
logN	1.000	1.000	1.000	1.000	1.000	27.209	1.000	1.000	1.000	1.000	1.000	37.282
t3	1.000	1.000	1.000	1.000	1.000	26.599	1.000	1.000	1.000	1.000	1.000	37.642
chisq10	1.000	1.000	1.000	1.000	1.000	27.418	1.000	1.000	1.000	1.000	1.000	37.731
Gamma(7,1)	1.000	1.000	1.000	1.000	1.000	27.659	1.000	1.000	1.000	1.000	1.000	38.232
Bootstrap Lobato												
N	0.055	0.048	0.053	0.037	0.035	53.110	0.050	0.046	0.067	0.049	0.047	72.528
logN	1.000	1.000	1.000	1.000	1.000	52.632	1.000	1.000	1.000	1.000	1.000	71.845
t3	1.000	1.000	1.000	1.000	1.000	52.763	1.000	1.000	1.000	1.000	1.000	71.454
chisq10	1.000	1.000	1.000	1.000	1.000	52.455	1.000	1.000	1.000	1.000	1.000	73.413
Gamma(7,1)	1.000	1.000	1.000	1.000	1.000	53.204	1.000	1.000	1.000	1.000	1.000	72.253
Bootstrap Epps												
N	0.051	0.043	0.033	0.043	0.051	78.920	0.055	0.054	0.056	0.044	0.064	101.883
logN	1.000	1.000	1.000	1.000	1.000	78.194	1.000	1.000	1.000	1.000	1.000	101.753
t3	0.979	0.995	0.998	0.996	0.985	79.735	1.000	1.000	1.000	1.000	1.000	100.766
chisq10	0.911	0.986	0.996	0.995	0.945	80.841	0.997	1.000	1.000	1.000	0.998	101.250
Gamma(7,1)	1.000	1.000	1.000	1.000	1.000	78.688	1.000	1.000	1.000	1.000	1.000	101.360
El Bouch												
N	0.065	0.053	0.047	0.061	0.059	1.419	0.055	0.064	0.051	0.048	0.045	2.467
logN	1.000	1.000	1.000	1.000	1.000	1.435	1.000	1.000	1.000	1.000	1.000	2.500
t3	1.000	1.000	1.000	1.000	1.000	1.453	1.000	1.000	1.000	1.000	1.000	2.492
chisq10	0.100	0.609	0.871	0.609	0.076	1.439	0.176	0.858	0.984	0.865	0.173	2.470
Gamma(7,1)	1.000	1.000	1.000	1.000	1.000	1.444	1.000	1.000	1.000	1.000	1.000	2.483

4.2 Real data application

As an illustrative example, we analyze the monthly mean carbon dioxide, in parts per million (*ppm*), measured at the Mauna Loa Observatory, in Hawaii, from March 1958 to November 2018. The carbon dioxide data measured as the mole fraction in dry air on Mauna Loa constitute the longest record of direct measurements of *CO*2 in the atmosphere. This dataset is available in the **astsa** package (Stoffer, 2020) under the name *cardox* data and it is displayed in the left panel of Figure 1. The plot's grid is created using the **cowplot** package (Wilke, 2020).

The objective of this subsection is to propose a model to analyze this time series and check the assumptions on the residuals of the model using our implemented `check_residuals()` function. The time series clearly has trend and seasonal components (see left panel of Figure 1), therefore, an adequate model that filters both components has to be selected. We make use of an ETS model. For its implementation, we make use the `ets()` function from the **forecast** package (Hyndman and Khandakar, 2008). This function fits 32 different ETS models and selects the best model according to information criteria such as *Akaike's information criterion* (AIC) or *Bayesian Information criteria* (BIC) (Chen and Chen, 2008). The results provided by the `ets()` function are:

Table 3: Average running time in seconds, over 1000 iterations, to compute the null hypothesis of Gaussianity for each of the studied tests (first column) and different sample sizes, $n = 1000$ (second column), $n = 2000$ (third column), $n = 3000$ (fourth column), $n = 4000$ (fifth column) and $n = 5000$ (sixth column). Each iteration makes use of a Gaussian AR(1) process with parameter $phi = 0.5$.

tests	n = 1000	n = 2000	n = 3000	n = 4000	n = 5000
Lobato and Velasco	0.0010	0.0014	0.0020	0.0026	0.0035
Epps	0.0010	0.0015	0.0021	0.0027	0.0035
Random Projections	0.0026	0.0045	0.0063	0.0082	0.0105
El Bouch	0.0023	0.0046	0.0074	0.0109	0.0152
Psaradakis and Vavra	0.0286	0.0429	0.0565	0.0012	0.0014
Bootstrap Lobato	0.0542	0.0014	0.0019	0.0025	0.0032
Bootstrap Epps	0.0013	0.0018	0.0023	0.0029	0.0037

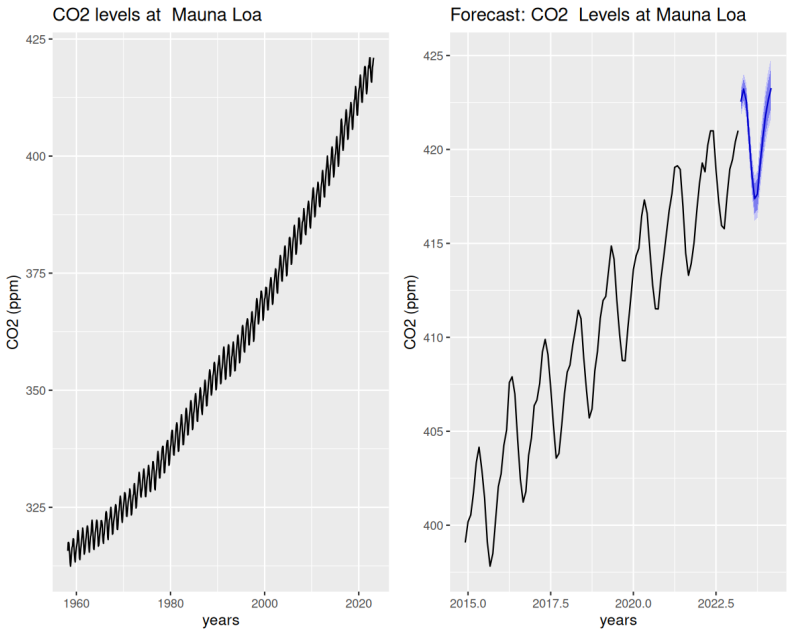


Figure 1: Left panel: CO2 Levels at Mauna Loa, time-series plot. The cardox data show a positive tendency and strong seasonality. Right panel: forecast of the next 12 months for the CO2 levels at Mauna Loa, the model's predictions capture the time-series behaviour.

```
library(forecast)
library(astsa)
model = ets(cardox)
summary(model)
#> ETS(M,A,A)
#>
#> Call:
#> ets(y = cardox)
#>
#>   Smoothing parameters:
#>     alpha = 0.5451
#>     beta  = 0.0073
#>     gamma = 0.1076
#>
#>   Initial states:
#>     l = 314.4546
#>     b = 0.0801
#>     s = 0.6986 0.0648 -0.8273 -1.8999 -3.0527 -2.7629
#>            -1.2769 0.7015 2.1824 2.6754 2.3317 1.165
#>
#>   sigma:  9e-04
#>
#>      AIC     AICc      BIC
#> 3429.637 3430.439 3508.867
#>
#> Training set error measures:
#>                     ME      RMSE       MAE          MPE       MAPE     MASE
#> Training set 0.018748 0.3158258 0.2476335 0.005051657 0.06933903 0.152935
#>                     ACF1
#> Training set 0.09308391
```

The resulting model, proposed by the `ets()` function, for analyzing the *carbon dioxide* data in *Mauna Loa* is an $ETS[M, A, A]$ model. The parameters α, β and γ (see Definition 1) have being estimated using the least squares method. If the assumptions on the model are satisfied, then the errors of the model behave like a Gaussian stationary process. To check it, we make use of the function `check_residuals()`. For more details on the compatibility of this function with the models obtained by other packages see the **nortsTest** repository. In the following, we display the results of using the *Augmented Dickey-Fuller* test (*Subsection 3.1*) to check the stationary assumption and the *random projection* test with `k = 1` projections to check the normality assumption. For the other test options see the function's documentation.

```
check_residuals(model,unit_root = "adf",normality = "rp",
                  plot = TRUE)

#>
#>  ************************************************
#>
#>  Unit root test for stationarity:
#>
#>  Augmented Dickey-Fuller Test
#>
#> data:  y
#> Dickey-Fuller = -9.8935, Lag order = 9, p-value = 0.01
#> alternative hypothesis: stationary
#>
#>
#>  Conclusion: y is stationary
#>  ************************************************
#>
#>  Goodness of fit test for Gaussian Distribution:
#>
#>  k random projections test.
#>
#> data:  y
```

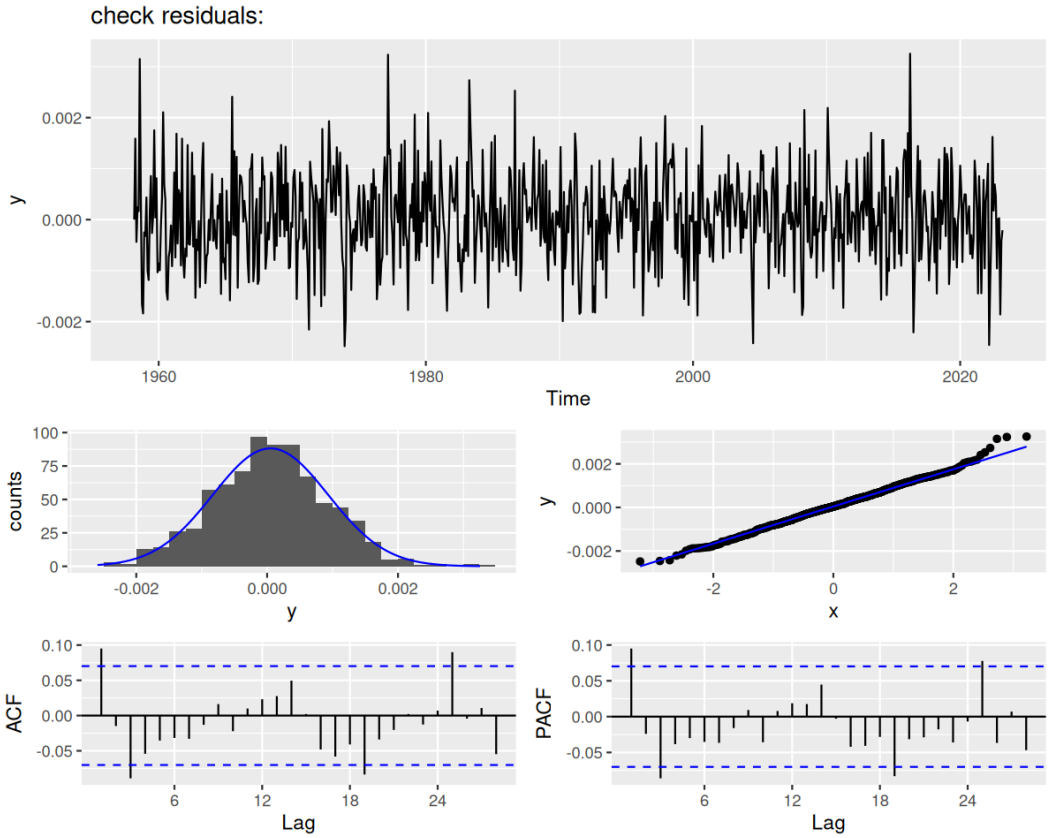


Figure 2: Check residuals plot for the ETS(M,A,A) model. The upper panel shows the residuals time-series plot, showing small oscillations around zero, which insinuates stationarity. The middle plots are the residuals histogram (middle-left) and quantile-quantile plot (middle-right), both plots suggest that the residuals have a normal distribution. The lower panel shows the autocorrelation functions, for both plots, the autocorrelations are close to zero giving the impression of stationarity.

```
#> k = 1, p.value adjust = Benjamini & Yekutieli, p-value = 1
#> alternative hypothesis: y does not follow a Gaussian Process
#>
#>
#>  Conclusion: y follows a Gaussian Process
#>
#>  ***************************************************
```

The obtained results indicate that the null hypothesis of non stationarity is rejected at significance level $\alpha = 0.01$. Additionally, there is no evidence to reject the null hypothesis of normality at significance level $\alpha = 0.05$. Consequently, we conclude that the residuals follow a stationary Gaussian process, having that the resulting $ETS[M, A, A]$ model adjusts well to the *carbon dioxide* data in *Mauna Loa*.

In the above displayed `check_residuals()` function, the `plot` argument is set to `TRUE`. The resulting plots are shown in Figure 2. The plot in the *top* panel and the auto-correlation plots in the bottom panels insinuate that the residuals have a stationary behavior. The *top* panel plot shows slight oscillations around zero and the auto-correlations functions in the *bottom* panels have values close to zero in every lag. The histogram and qq-plot in the *middle* panels suggest that the marginal distribution of the residuals is normally distributed. Therefore, Figure 2 agrees with the reported results, indicating that the assumptions of the model are satisfied.

As the assumptions of the model have been checked, it can be used for instance to forecast. The result of applying the following function is displayed in Figure 1. It presents the carbon dioxide data for the last 8 years and a forecast of the next 12 months. It is observable from the plot that the model captures the process trend and periodicity.

```
autoplot(forecast(model,h = 12),include = 100,
         xlab = "years",ylab = "CO2 (ppm)",
```

```
        main = "Forecast: Carbon Dioxide Levels at Mauna Loa")
```

5 Conclusions

For independent data, the **nortest** package (Gross and Ligges, 2015) provides five different tests for normality, the **mvnormtest** package (Jarek, 2012) performs the Shapiro-Wilks test for multivariate data and the **MissMech** package (Jamshidian et al., 2014) provides tests for normality in multivariate incomplete data. To test the normality of dependent data, some authors such as Psaradakis and Vávra (2017) and Nieto-Reyes et al. (2014) have available undocumented `Matlab` code, which is almost only helpful in re-doing their simulation studies.

To our knowledge, no consistent implementation or package of tests for normality of stationary processes has been done before. Therefore, the **nortsTest** is the first package to implement normality tests in stationary processes. This work gives a general overview of a careful selection of tests for normality in the stationary process, which consists of the most available types of tests. It additionally provides examples that illustrate each of the test implementations.

For checking the model's assumptions, the **forecast** and **astsa** packages contain functions for visual diagnostic. Following the same idea, **nortsTest** provides similar diagnostic methods; it also reports the results of testing stationarity and normality, the main assumptions for the residuals in time series analysis.

6 Future work and projects

A further version of the **nortsTest** package will incorporate additional tests such as Bispectral (Hinich, 1982) and Stein's characterization (Bontemps and Meddahi, 2005). Further future work will include a Bayesian version of a *residuals check* procedure that uses the random projection method. Any future version under development can be installed from `GitHub` using the following code.

```
if (!requireNamespace("remotes")) install.packages("remotes")
remotes::install_github("asael697/nortsTest",dependencies = TRUE)
```

Acknowledgment

This work was supported by grant PID2022-139237NB-I00 funded by "ERDF A way of making Europe" and MCIN/AEI/10.13039/501100011033.

References

T. W. Anderson. On the distribution of the two-sample Cramer-von Mises criterion. *The Annals of Mathematical Statistics*, 33(3):1148 – 1159, 1962. doi: 10.1214/aoms/1177704477. URL https://doi.org/10.1214/aoms/1177704477. [p138, 144]

T. W. Anderson and D. A. Darling. Asymptotic theory of certain goodness of fit criteria based on stochastic processes. *Annals of Mathematical Statistics*, 23(2):193–212, 06 1952. doi: 10.1214/aoms/1177729437. [p135, 138]

J. Bai and S. Ng. Tests for skewness, kurtosis, and normality for time series data. *Journal of Business & Economic Statistics*, 23(1):49–60, 2005. doi: 10.1198/073500104000000271. [p138, 142]

J. Beaulieu and J. A. Miron. Seasonal unit roots in aggregate U.S. data. *Journal of Econometrics*, 55(1): 305 – 328, 1993. ISSN 0304-4076. doi: 10.1016/0304-4076(93)90018-Z. [p139]

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995. ISSN 00359246. URL http://www.jstor.org/stable/2346101. [p143]

Y. Benjamini and D. Yekutieli. The control of the false discovery rate in multiple testing under dependency. *The Annals of Statistics*, 29(4):1165–1188, 2001. ISSN 00905364. URL http://www.jstor.org/stable/2674075. [p143, 144]

A. Berg, E. Paparoditis, and D. N. Politis. A bootstrap test for time series linearity. *Journal of Statistical Planning and Inference*, 140(12):3841 – 3857, 2010. ISSN 0378-3758. doi: 10.1016/j.jspi.2010.04.047. [p138]

T. Bollerslev. Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3): 307 – 327, 1986. ISSN 0304-4076. doi: 10.1016/0304-4076(86)90063-1. [p137, 144]

C. Bontemps and N. Meddahi. Testing normality: a gmm approach. *Journal of Econometrics*, 124(1):149 – 186, 2005. ISSN 0304-4076. doi: 10.1016/j.jeconom.2004.02.014. [p138, 152]

G. Box and D. A. Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American Statistical Association*, 65(332):1509–1526, 1970. doi: 10.1080/01621459.1970.10481180. [p139]

G. E. P. Box and G. Jenkins. *Time series analysis, forecasting and control*. Holden-Day, Inc., USA, 1990. ISBN 0816211043. URL https://www.wiley.com/en-us/Time+Series+Analysis. [p137]

P. Bühlmann. Sieve bootstrap for time series. *Bernoulli*, 3(2):123–148, 1997. ISSN 13507265. URL http://www.jstor.org/stable/3318584. [p145]

F. Canova and B. E. Hansen. Are seasonal patterns constant over time? a test for seasonal stability. *Journal of Business & Economic Statistics*, 13(3):237–252, 1995. doi: 10.1080/07350015.1995.10524598. [p139]

J. Chen and Z. Chen. Extended bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008. ISSN 0006-3444. doi: 10.1093/biomet/asn034. [p148]

J. Cuesta-Albertos, E. del Barrio, R. Fraiman, and C. Matrán. The random projection method in goodness of fit for functional data. *Computational Statistics & Data Analysis*, 51(10):4814 – 4831, 2007. ISSN 0167-9473. doi: 10.1016/j.csda.2006.09.007. [p138, 143]

R. B. D'Agostino and M. A. Stephens. Goodness-of-fit techniques. *Quality and Reliability Engineering International*, 3(1):71–71, 1986. doi: 10.1002/qre.4680030121. [p135, 138]

G. E. Dallal and L. Wilkinson. An analytic approximation to the distribution of lilliefors's test statistic for normality. *The American Statistician*, 40(4):294–296, 1986. doi: 10.1080/00031305.1986.10475419. URL https://www.tandfonline.com/doi/abs/10.1080/00031305.1986.10475419. [p138]

J. L. de Lacalle. *'uroot': Unit root tests for seasonal time series*, 2019. URL https://CRAN.R-project.org/package=uroot. 'R' package version 2.1-0. [p139]

J. A. Doornik and H. Hansen. An omnibus test for univariate and multivariate normality. *Oxford Bulletin of Economics and Statistics*, 70(s1):927–939, December 2008. doi: 10.1111/j.1468-0084.2008. URL https://ideas.repec.org/a/bla/obuest/v70y2008is1p927-939.html. [p138]

S. El Bouch, O. Michel, and P. Comon. A normality test for multivariate dependent samples. *Signal Processing*, 201:108705, 2022. doi: 10.1016/j.sigpro.2022.108705. [p135, 138, 139, 145, 146]

R. F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica*, 50(4):987–1007, 1982. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1912773. [p137]

T. W. Epps. Testing that a stationary time series is Gaussian. *The Annals of Statistics*, 15(4):1683–1698, 12 1987. doi: 10.1214/aos/1176350618. [p135, 138, 139, 141, 146]

T. Gasser. Goodness-of-fit tests for correlated data. *Biometrika*, 62(3):563–570, 1975. ISSN 00063444. URL http://www.jstor.org/stable/2335511. [p138, 141, 142]

A. Gelman, J. Carlin, H. Stern, D. Dunson, A. Vehtari, and D. Rubin. *Bayesian data analysis, third edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013. ISBN 9781439840955. URL https://books.google.nl/books?id=ZXL6AQAAQBAJ. [p143]

J. Gross and U. Ligges. *'nortest': Tests for normality*, 2015. URL https://CRAN.R-project.org/package=nortest. 'R' package version 1.0-4. [p138, 152]

N. Henze and B. Zirkler. A class of invariant consistent tests for multivariate normality. *Communications in Statistics - Theory and Methods*, 19(10):3595–3617, 1990. doi: 10.1080/03610929008830400. URL https://doi.org/10.1080/03610929008830400. [p138]

M. J. Hinich. Testing for Gaussianity and linearity of a stationary time series. *Journal of Time Series Analysis*, 3(3):169–176, 1982. doi: 10.1111/j.1467-9892.1982.tb00339. [p138, 152]

C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5 – 10, 2004. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2003.09.015. [p135]

Y. Hong. Hypothesis testing in time series via the empirical characteristic function: a generalized spectral density approach. *Journal of the American Statistical Association*, 94(448):1201–1220, 1999. doi: 10.2307/2669935. [p139]

R. Hyndman and Y. Khandakar. Automatic time series forecasting: The 'forecast' package for 'R'. *Journal of Statistical Software, Articles*, 27(3):1–22, 2008. ISSN 1548-7660. doi: 10.18637/jss.v027.i03. [p135, 137, 139, 148]

R. J. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder. *Forecasting with exponential smoothing: The state space approach*. Springer, 2008. ISBN 9783540719168. doi: 10.1111/j.1751-5823.2009.00085_17. [p137]

M. Jamshidian, S. Jalal, and C. Jansen. 'missmech': An 'R' package for testing homoscedasticity, multivariate normality, and missing completely at random (mcar). *Journal of Statistical Software*, 56 (6):1–31, 2014. URL http://www.jstatsoft.org/v56/i06/. [p152]

S. Jarek. *'mvnormtest': Normality test for multivariate variables*, 2012. URL https://CRAN.R-project.org/package=mvnormtest. 'R' package version 0.1-9. [p152]

C. M. Jarque and A. K. Bera. Efficient tests for normality, homoscedasticity and serial independence of regression residuals. *Economics Letters*, 6(3):255 – 259, 1980. ISSN 0165-1765. doi: 10.1016/0165-1765(80)90024-5. [p138, 141, 144]

D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of Econometrics*, 54(1):159 – 178, 1992. ISSN 0304-4076. doi: 10.1016/0304-4076(92)90104-Y. [p139]

I. Lobato and C. Velasco. A simple test of normality for time series. *Econometric Theory*, 20:671–689, 08 2004. doi: 10.1017/S0266466604204030. [p135, 138, 139, 141, 142]

Z. Lomnicki. Tests for departure from normality in the case of linear stochastic processes. *Metrika: International Journal for Theoretical and Applied Statistics*, 4(1):37–62, 1961. URL https://EconPapers.repec.org/RePEc:spr:metrik:v:4:y:1961:i:1:p:37-62. [p141]

K. V. Mardia. Measures of multivariate skewness and kurtosis with applications. *Biometrika*, 57(3): 519–530, 1970. URL http://www.jstor.org/stable/2334770. [p145]

S. G. Meintanis. A review of testing procedures based on the empirical characteristic function. *South African Statistical Journal*, 50(1):1–14, 2016. doi: 10.10520/EJC186846. [p139]

E. Moulines, K. Choukri, and M. Sharbit. Testing that a multivariate stationary time-series is Gaussian. In *[1992] IEEE Sixth SP Workshop on Statistical Signal and Array Processing*, pages 185–188. IEEE, 1992. doi: 10.1109/SSAP.1992.246818. [p145]

A. Nieto-Reyes. On the non-Gaussianity of the height of sea waves. *Journal of Marine Science and Engineering*, 9(12), 2021. ISSN 2077-1312. URL https://www.mdpi.com/2077-1312/9/12/1446. [p135]

A. Nieto-Reyes. On the non-Gaussianity of sea surface elevations. *Journal of Marine Science and Engineering*, 10(9), 2022. ISSN 2077-1312. doi: 10.3390/jmse10091303. URL https://www.mdpi.com/2077-1312/10/9/1303. [p135]

A. Nieto-Reyes, J. A. Cuesta-Albertos, and F. Gamboa. A random-projection based test of Gaussianity for stationary processes. *Computational Statistics & Data Analysis*, 75:124 – 141, 2014. ISSN 0167-9473. doi: 10.1016/j.csda.2014.01.013. [p135, 138, 139, 141, 143, 146, 152]

D. R. Osborn, A. P. L. Chui, J. P. Smith, and C. R. Birchenhall. Seasonality and the order of integration for consumption. *Oxford Bulletin of Economics and Statistics*, 50(4):361–377, 1988. doi: 10.1111/j.1468-0084.1988.mp50004002.x. [p139]

K. Pearson and O. M. F. E. Henrici. X. Contributions to the mathematical theory of evolution.-II Skew variation in homogeneous material. *Philosophical Transactions of the Royal Society of London. (A.)*, 186: 343–414, 1895. doi: 10.1098/rsta.1895.0010. [p138]

P. Perron. Trends and random walks in macroeconomic time series: Further evidence from a new spproach. *Journal of Economic Dynamics and Control*, 12(2):297 – 332, 1988. ISSN 0165-1889. doi: 10.1016/0165-1889(88)90043-7. [p139]

G. Petris, S. Petrone, and P. Campagnoli. Dynamic linear models with 'R', 2007. ISSN 03067734. [p137]

Z. Psaradakis. Normality tests for dependent data. Working and Discussion Papers WP 12/2017, Research Department, National Bank of Slovakia, 2017. URL https://ideas.repec.org/p/svk/wpaper/1053.html. [p138]

Z. Psaradakis and M. Vávra. Normality tests for dependent data: large-sample and bootstrap approaches. *Communications in statistics-simulation and computation*, 49(2):283–304, 2020. doi: 10.1080/03610918.2018.1485941. [p135, 138, 141, 144, 145]

Z. Psaradakis and M. Vávra. A distance test of normality for a wide class of stationary processes. *Econometrics and Statistics*, 2:50 – 60, 2017. ISSN 2452-3062. doi: 10.1016/j.ecosta.2016.11.005. [p135, 138, 139, 144, 145, 146, 152]

N. Pya, V. Voinov, R. Makarov, and Y. Voinov. *'mvnTest': Goodness of fit tests for multivariate normality*, 2016. URL https://CRAN.R-project.org/package=mvnTest. 'R' package version 1.1-0. [p138]

D. Qiu. *'aTSA': Alternative time series analysis*, 2015. URL https://CRAN.R-project.org/package=aTSA. 'R' package version 3.1.2. [p135]

J. P. Royston. An extension of Shapiro and Wilk's W test for normality to large samples. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(2):115–124, 1982. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2347973. [p138]

J. P. Royston. Approximating the shapiro-wilk W-test for non-normality. *Journal of Statistics and Computing*, 2(3):117–119, 1992. URL https://doi.org/10.1007/BF01891203. [p138]

P. Royston. A pocket-calculator algorithm for the Shapiro-Francia test for non-normality: An application to medicine. *Statistics in Medicine*, 12(2):181–184, 1993. doi: 10.1002/sim.4780120209. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sim.4780120209. [p138]

S. E. Said and D. A. Dickey. Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 71(3):599–607, 12 1984. ISSN 0006-3444. doi: 10.1093/biomet/71.3.599. [p139]

S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3-4):591–611, 12 1965. ISSN 0006-3444. doi: 10.1093/biomet/52.3-4.591. [p138, 144]

R. Shumway and D. Stoffer. *Time series analysis and itts applications: with 'R' examples*. Springer Texts in Statistics. Springer New York, 2010. ISBN 9781441978646. URL https://books.google.es/books?id=dbS5IQ8P5gYC. [p135, 136]

N. Smirnov. Table for estimating the goodness of fit of empirical distributions. *Annals of Mathematical Statistics*, 19(2):279–281, 06 1948. doi: 10.1214/aoms/1177730256. [p138]

Y. Steinberg and O. Zeitouni. On tests for normality. *IEEE Transactions on Information Theory*, 38(6):1779–1787, 1992. doi: 10.1109/18.165450. [p145]

D. Stoffer. *'astsa': Applied statistical time series analysis*, 2020. URL https://CRAN.R-project.org/package=astsa. 'R' package version 1.10. [p135, 148]

R. C. Team. *'R': A language and environment for statistical computing*. 'R' Foundation for Statistical Computing, Vienna, Austria, 2018. URL https://www.R-project.org/. [p135]

A. Trapletti and K. Hornik. *'tseries': Time series analysis and computational finance*, 2019. URL https://CRAN.R-project.org/package=tseries. 'R' package version 0.10-47. [p139, 141]

R. Tsay. *Analysis of financial time series*. Wiley-Interscience, Chicago, second edition, 2010. ISBN 978-0470414354. doi: 10.1002/0471264105. [p135, 136]

R. M. Vassilly Voinov, Natalie Pya and Y. Voinov. New invariant and consistent chi-squared type goodness-of-fit tests for multivariate normality and a related comparative simulation study. *Communications in Statistics - Theory and Methods*, 45(11):3249–3263, 2016. doi: 10.1080/03610926.2014.901370. URL https://doi.org/10.1080/03610926.2014.901370. [p138]

L. Wasserman. *All of nonparametric statistics*. Springer, New York, 2006. ISBN 9780387251455. doi: 10.1007/0-387-30623-4. [p136]

M. West and J. Harrison. *Bayesian forecasting and dynamic models*. Springer Series in Statistics. Springer New York, 2006. ISBN 9780387227771. URL https://books.google.nl/books?id=0mPgBwAAQBAJ. [p137]

H. Wickham. *'ggplot2': Elegant graphics for data analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. URL http://ggplot2.org. [p139]

C. O. Wilke. *'cowplot': Streamlined plot theme and plot annotations for 'ggplot2'*, 2020. URL https://CRAN.R-project.org/package=cowplot. 'R' package version 1.1.1. [p148]

D. Wuertz, T. Setz, Y. Chalabi, C. Boudt, P. Chausse, and M. Miklovac. *'fGarch': Rmetrics - autoregressive conditional heteroskedastic modelling*, 2017. URL https://CRAN.R-project.org/package=fGarch. 'R' package version 3042.83. [p144]

Asael Alonzo Matamoros
Aalto University
Department of Computer Science
Eespo, Finland
https://asael697.github.io
izhar.alonzomatamoros@aalto.fi

Alicia Nieto-Reyes
Universidad de Cantabria
Departmento de Mathemáticas, Estadística y Computación
Avd. de los Castros s/n. 39005 Santander, Spain
https://orcid.org/0000-0002-0268-3322
alicia.nieto@unican.es

Claudio Agostinelli
University of Trento
Department of Mathematics
Via Sommarive, 14 - 38123 Povo
https://orcid.org/0000-0001-6702-4312
claudio.agostinelli@unitn.it

shinymgr: A Framework for Building, Managing, and Stitching Shiny Modules into Reproducible Workflows

by Laurence A. Clarfeld, Caroline Tang, and Therese Donovan

Abstract The R package shinymgr provides a unifying framework that allows Shiny developers to create, manage, and deploy a master Shiny application comprised of one or more "apps", where an "app" is a tab-based workflow that guides end-users through a step-by-step analysis. Each tab in a given "app" consists of one or more Shiny modules. The shinymgr app builder allows developers to "stitch" Shiny modules together so that outputs from one module serve as inputs to the next, creating an analysis pipeline that is easy to implement and maintain. Apps developed using shinymgr can be incorporated into R packages or deployed on a server, where they are accessible to end-users. Users of shinymgr apps can save analyses as an RDS file that fully reproduces the analytic steps and can be ingested into an RMarkdown or Quarto report for rapid reporting. In short, developers use the shinymgr framework to write Shiny modules and seamlessly combine them into Shiny apps, and end-users of these apps can execute reproducible analyses that can be incorporated into reports for rapid dissemination. A comprehensive overview of the package is provided by 12 learnr tutorials.

1 Introduction

The **shiny** R package allows users to build interactive web apps straight from R, without advanced knowledge of HTML or JavaScript (Chang et al., 2022). A *shiny* web app can permit an expedient analysis pipeline or workflow. Ideally, the pipeline can produce outputs that are fully reproducible (Peng, 2011; Gentleman and Lang, 2007; Alston and Rick, 2021). Moreover, the pipeline can permit rapid reporting to convey the results of an analysis workflow to a target audience (Stoudt et al., 2021) (Figure 1).

shiny applications range from simple to complex, each with an intended purpose developed for an intended user audience. Several R packages provide a development framework for building multi-faceted master applications, including **shinipsum** for prototyping (Fay and Rochette, 2020), **golem** (Fay et al., 2021), and **rhino** (Żyła et al., 2023).

From the developer's perspective, complex *shiny* applications can result in many lines of code, creating challenges for collaborating, debugging, streamlining, and maintaining the overall product. *shiny* modules are a solution to this problem. As stated by Winston Chang (shi, 2020), "A *shiny* module is a piece of a *shiny* app. It can't be directly run, as a *shiny* app can. Instead, it is included as part of a larger app . . . Once created, a *shiny* module can be easily reused – whether across different apps, or multiple times in a single app." *shiny* modules, and modularization in general, are a core element of agile software development practices (Larman, 2004). Several authors have contributed R packages for distributing pre-written *shiny* modules for general use, including the **datamods** (Perrier et al., 2022), **shiny.reglog** (Kosinski, 2022), **periscope** (Brett and Neuhaus, 2022), **shinyauthr** (Campbell, 2021), and

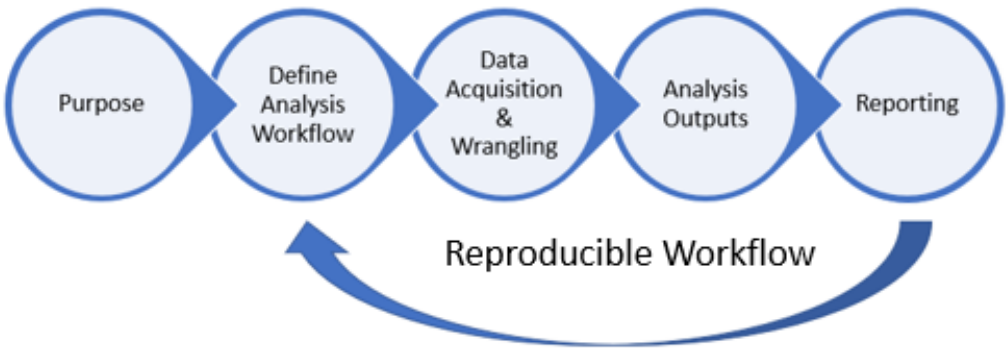


Figure 1: Stages of a reproducible workflow, a process that moves an inquiry from raw data to insightful contribution.

jsmodule (Kim and Lee, 2022) packages.

However, as the number of available modules increases, there is a pressing need for documenting available *shiny* modules and easily incorporating them into new workflows. For example, consider a toy modular-based app that guides a user through an analysis of the famous "Iris Dataset," which contains 150 records of 3 species of iris, including measurements of the length and width of the flowers' sepals and petals (Fisher, 1936). The app, called "Iris Explorer," consists of 5 tabs to be worked through in sequence (Figure 2, top).

Tab 1 displays instructions for use, while tab 2 performs a k-means clustering of the data, where k is specified by the user. The resulting clusters are displayed with two variables of the user's choosing as depicted in Figure 2. In tab 3, the user will choose a value n, indicating the number of rows by which to randomly subset the data, and in tab 4 the user selects a single variable to be plotted as a bar chart. Finally, in tab 5 the user can save their outputs as an RDS file. This contrived example includes some key elements of a typical workflow in that the five tabs introduce a dataset, guide the user through light data wrangling, produce analysis outputs, and offer the ability to save the results.

The app's blueprint (Figure 2, bottom) identifies the *shiny* modules in each tab, showing how outputs from one module can serve as inputs to the next. Note that while this example shows a single module in each tab with differing inputs/outputs, in the general case tabs can contain an arbitrary number of *shiny* modules (including multiple instances of the same module) and each module can have multiple inputs/outputs.

While two of the *shiny* modules within the "iris_explorer" app pertain to the iris dataset specifically ("iris_intro" and "iris_cluster"), the remaining *shiny* modules ("subset_rows", "single_column_plot", and "save") may be incorporated into other apps.

Figure 2: Top: The 'iris_explorer' app guides a user through an analysis of the iris dataset in a tab-based sequence. Bottom: A blueprint of the 'iris_explorer' app shows the 5 tabs, each containing a single module identified by name within blue ovals. Some of the shiny modules require inputs and generate outputs as identified in gray polygons.

Developers who utilize the same *shiny* modules within different apps will naturally be faced with several questions:

1. Which *shiny* modules have been written? Are they well documented with unit testing?
2. What are the module's inputs (arguments) and outputs (returns)?
3. Where are the *shiny* modules stored?
4. How can *shiny* modules be combined into a cohesive, well-documented app?
5. How can production-ready apps be deployed for end-users?

Users of an app created with the *shinymgr* framework may wish to know:

6. Can analysis outputs be saved as a fully reproducible workflow?
7. Can outputs be ingested into a *Rmarkdown* or *Quarto* template for rapid reporting?

1.1 Introducing *shinymgr*

The R package, **shinymgr**, was developed to meet these challenges (Clarfeld et al., 2024). The *shinymgr* package includes a general framework that allows developers to create *shiny* modules, stitch them together as individual "apps" that are embedded within the master *shiny* application, and then deploy them on a *shiny* server or incorporate them into R packages. *shinymgr* was motivated from our first-hand experience in our work building tools that assist scientists in remote wildlife monitoring with the R package *AMMonitor* (Balantic and Donovan, 2020). Dependencies of *shinymgr* include the packages **DBI** (R Special Interest Group on Databases (R-SIG-DB) et al., 2022), **reactable** (Lin, 2022), **RSQLite** (Müller et al., 2022), **renv** (Ushey, 2023), **shiny** (Chang et al., 2022), **shinyjs** (Attali, 2021), and **shinydashboard** (Chang and Borges Ribeiro, 2021).

From the developer's perspective, an "app" consists of an ordered set of tabs, each of which contain specified *shiny* modules. *shiny* modules are the basic element in the *shinymgr* framework; they can be used and re-used across different tabs and different apps. Information about each module and app is stored in a SQLite database (Hipp, 2020). The *shinymgr* app builder "stitches" *shiny* modules together so that outputs from one module serve as inputs to the next, creating an analysis pipeline that is easy to implement and maintain. When apps are production-ready , developers can deploy a stand-alone *shiny* application independent of *shinymgr* on a server or within an R package. From the end-user's perspective, an "app" created with the *shinymgr* framework consists of an ordered series of *shiny* tabs, establishing an analysis. Users can save their inputs and outputs as an RDS file to ensure full reproducibility. Furthermore, the RDS file may be loaded into an R Markdown (Rmd) or Quarto (qmd) template for rapid reporting. We are unaware of existing packages that unify the elements of modularization, documentation, reproducibility, and reporting in a single framework.

We introduce *shinymgr* in sections 2-4 below. In section 2 we describe how developers can create apps using the *shinymgr* framework. In section 3 we describe how developers can deploy a *shinymgr* project on a local machine, server, or within an R package. In section 4 describes the end-user experience, where end-users execute an "app" and store results for reproducibility and reporting. The package tutorials and cheat sheet are described in section 5. The *shinymgr* package comes with a series of **learnr** (Schloerke et al., 2020) tutorials described at the end of the paper.

2 Developing *shinymgr* apps

2.1 Setting up *shinymgr*

The canonical home of *shinymgr* is `https://code.usgs.gov/vtcfwru/shinymgr/` where *shinymgr* users may post merge requests and bug fix requests. *shinymgr* may also be downloaded from CRAN.

```
install.packages("shinymgr")
```

The development version can be downloaded with:

```
remotes::install_gitlab(
  repo = "vtcfwru/shinymgr",
  auth_token = Sys.getenv("GITLAB_PAT"),
  host = "code.usgs.gov",
  build_vignettes = FALSE)
```

Once installed, a new *shinymgr* project can be created within a parent directory:

```
# set the directory path that will house the shinymgr project
parentPath <- getwd()

# set up raw directories and fresh database
shinymgr_setup(
  parentPath = parentPath,
  demo = TRUE)
```

The `shinymgr_setup()` function produces the following directory structure within the primary "shinymgr" directory. This structure consists of 3 files that make up the "master" app (global.R, server.R, and ui.R), and 9 directories. If the argument demo is set to FALSE, these directories will be largely empty, except for the "modules_mgr" and "database" directories, which will contain *shiny* modules for rendering *shinymgr*'s UI and an empty SQLite database, respectively. If the argument demo is set to TRUE, each directory will include several demo files as shown, including a pre-populated database. Here, we highlight a subset of the demo files related to the "iris_explorer" app to guide developers through the key elements of *shinymgr* (additional demo files come with package but are omitted here for clarity).

```
shinymgr
+-- analyses
|   \-- iris_explorer_Gandalf_2023_06_05_16_30.RDS
+-- data
|   \-- iris.RData
+-- database
|   \-- shinymgr.sqlite
+-- global.R
+-- modules
|   +-- iris_cluster.R
|   +-- iris_intro.R
|   +-- single_column_plot.R
|   \-- subset_rows.R
+-- modules_app
|   \-- iris_explorer.R
+-- modules_mgr
|   +-- add_app.R
|   +-- add_mod.R
|   +-- add_report.R
|   +-- add_tab.R
|   +-- app_builder.R
|   +-- my_db.R
|   +-- new_analysis.R
|   +-- new_report.R
|   +-- queries.R
|   +-- save_analysis.R
|   +-- stitch_script.R
|   \-- table.R
+-- reports
|   \-- iris_explorer
|       \-- iris_explorer_report.Rmd
+-- server.R
+-- tests
|   +-- shinytest
|   |   +-- test-iris_explorer-expected
|   |   |   +-- 001.json
|   |   |   +-- 001.png
|   |   |   +-- 002.json
|   |   |   \-- 002.png
|   |   \-- test-iris_explorer.R
|   +-- shinytest.R
|   +-- testthat
|   |   +-- test-iris_cluster.R
|   |   \-- test-subset_rows.R
|   \-- testthat.R
+-- ui.R
```

```
\-- www
    +-- dark_mode.css
    \-- shinymgr-hexsticker.png
```

The directory structure produced by `shinymgr_setup()` includes the following:

- The **analyses** directory provides the developer an example of a previously run analysis that was created using the *shinymgr* framework (an RDS file). An analysis file name includes the app name (e.g. "iris_explorer"), the name of the person who ran the analysis (e.g. "Gandalf"), and the date and time of the analysis (e.g., "iris_explorer_Gandalf_2023_06_05_16_30.RDS").
- The **data** directory stores RData files that can be used by various *shinymgr* apps (e.g., "iris.RData").
- The **database** directory stores the *shinymgr* SQLite database, named "shinymgr.sqlite." The database is used by the developer to track all *shiny* modules, their arguments (inputs), returns (outputs), and how they are combined into *shinymgr* apps.
- The **modules** directory stores stand-alone *shiny* modules. These files are largely written by the developer with the help of the `mod_init()` function, and are registered in the database with the `mod_register()` function. Four of the example *shiny* modules listed are used in the "iris_explorer" app.
- The **modules_app** directory stores *shiny* modules that are *shinymgr* "apps" – the stitching together of *shiny* modules into a tab-based layout that provides an analysis workflow (Figure 2 shows the "iris_explorer" app layout). Files within the "modules_app" directory are not written by hand - instead, they are created with the *shinymgr* "app builder."
- The **modules_mgr** directory stores *shiny* modules that build the overall *shinymgr* framework.
- The **reports** directory provides an example of an *RMarkdown* (Rmd) template (e.g., "iris_explorer_report.Rmd"), allowing for rapid reporting by an end-user.
- The **tests** directory stores both **testthat** (Wickham, 2011) and **shinytest** (Chang et al., 2021) code testing scripts.
- The **www** directory stores images that may be used by a *shiny* app.
- In addition to these directories, three files are created for launching the master *shinymgr shiny* application:

  1. **ui.R** - This file contains code to set the user interface for the master *shinymgr* app.
  2. **server.R** - The master server file.
  3. **global.R** - The global.R file is sourced into the server.R file at start-up. It sources all of the *shiny* modules within the *shinymgr* framework so they are available when *shinymgr* is launched.

2.2 The *shinymgr* developer's portal

Once set-up is complete, the `launch_shinymgr()` function will launch the *shinymgr* "Developer's Portal" UI, allowing developers to create and test new *shinymgr* apps.

```
# launch shinymgr
launch_shinymgr(shinyMgrPath = paste0(parentPath, "/shinymgr"))
```

The portal is recognizable by the *shinymgr* logo in the upper left corner (Figure 3). The portal consists of three main tabs in the left menu. The "Developer Tools" tab is used to create apps, view the *shinymgr* database, and register reports, while the "Analysis (beta)" and "Reports (beta)" tabs allow developers to evaluate apps from the user's perspective.

The "Developer Tools" section includes 4 tabs for app development: The "Build App" tab allows the developer to create new *shinymgr* apps from existing modules using the *shinymgr* app builder; the "Database" tab displays the *shinymgr* database tables, the "Queries" tab contains a set of standard database queries, and the "Add Reports" tab allows the developer to link a report (Rmd or qmd) to a given *shinymgr* app (Figure 3), as described below.

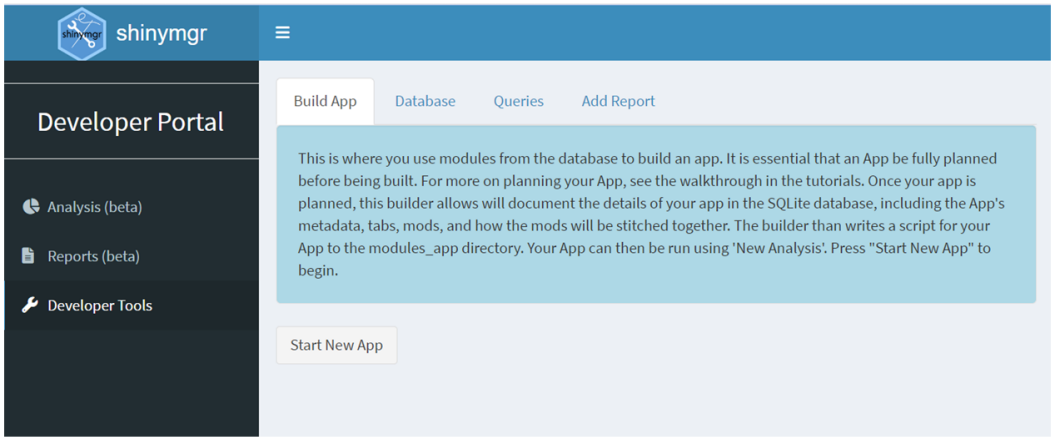


Figure 3: The shinymgr Developer Portal consists of a sidebar panel where developers can create new shiny modules and new apps, and test-drive analyses and reports from the user's perspective. The main panel shows the 'Build App' tab within the 'Developer Tools' section.

2.3 The *shinymgr* database

The *shinymgr* SQLite database ("shinymgr.sqlite") is a single file created by the `shinymgr_setup()` function. The database tracks all *shiny* modules, their arguments (inputs), returns (outputs), their package dependencies and version numbers, how they are combined into an "app," and any reports that are associated with apps. The database tables are populated via dedicated *shinymgr* functions.

The *shinymgr* database consists of 11 tables in total (Figure 4). These tables are connected to each other as a typical relational database, with primary keys establishing unique records in each table, and foreign keys that reference primary keys in other tables (see Appendix A for a full database schema and the "database" *learnr* tutorial for additional information).

The "apps," "appReports," "reports," "appTabs," and "tabs" tables largely store information on what a user would see when they run an analysis. The table "apps" stores information about apps such as "iris_explorer." Apps consist of tabs, which are listed in the "tabs" table. Tabs are linked to apps via the "appTabs" table. The table "reports" lists any Rmd or qmd files that serve as a report template, and the table "appReports" links a specific report with a specific app.

The remaining 6 tables in Figure 4 are "modules," "modFunctionArguments," "modFunctionReturns," "modPackages," "tabModules," and "appStitching." These tables largely store information about *shiny* modules that a developer creates, i.e., what *shiny* modules have been written, what are their arguments and returns, and what packages they use. The "tabModules" table identifies which tabs call which *shiny* modules (with a single tab capable of calling multiple *shiny* modules), and the "appStitching" table specifies how *shiny* modules are "stitched" together, i.e., which module returns are passed in as arguments to downstream *shiny* modules.

Four of the 11 database tables focus on modules, highlighting that *shiny* modules are basic building blocks of any *shinymgr* app. Developers create new *shiny* modules with the `mod_init()` function, which copies a *shinymgr* module template (an R file template) that includes a header with key-value

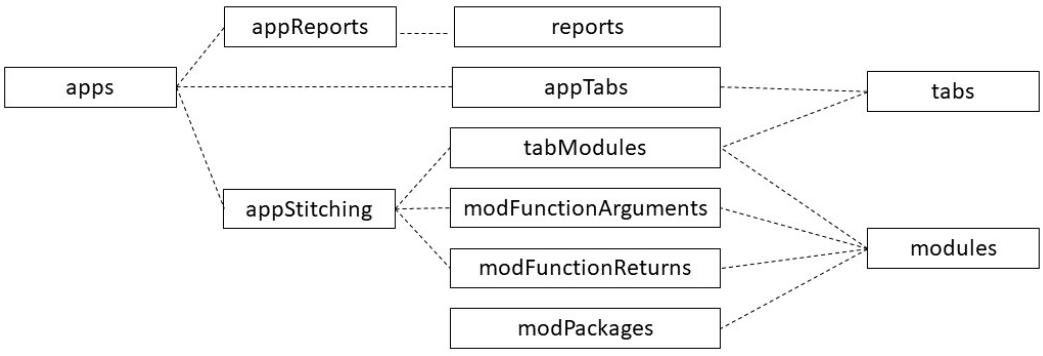


Figure 4: The 11 tables of the shinymgr SQLite database. Lines indicate how the tables are related to each other.

that describe the module, including the module name, display name, description, citation, notes, and module arguments and returns (if any). For example, the header of the iris_cluster module is:

```
#!! ModName = iris_cluster
#!! ModDisplayName = Iris K-Means Clustering
#!! ModDescription = Clusters iris data based on 2 attributes
#!! ModCitation = Baggins, Bilbo.  (2023). iris_cluster. [Source code].
#!! ModNotes = Demo module for the shinymgr package.
#!! ModActive = 1
#!! FunctionReturn = returndf !! selected attributes and their assigned clusters !! data.frame
```

The module code is written beneath the header (see Appendix B for an example). Function calls within the module code should be written with `package::function()` notation, making explicit any R package dependencies. Once the module is completed, unit tests can written and stored in the *shinymgr* project's "tests" directory. The final module file is saved to the "modules" directory and registered into the database with the `mod_register()` function. The `mod_register()` function populates the modules, "modFunctionArguments", and "modFunctionReturns" SQLite database tables. Further, it uses the *renv* package to identify any package dependencies and inserts them into the modPackages table. Readers are referred to the "modules" "tests", and "shinymgr_modules" *learnr* tutorials that come with the *shinymgr* package for more details.

Once modules are registered in the database, the developer can incorporate them into new apps. As *shiny* modules and apps in the database represent files that contain their scripts, deleting a module or an app from the database will delete all downstream database entries as well as (optionally) the actual files themselves. Deletion of a module will fail if it is being used in other apps. Module updates can be versioned by creating a new module and then referencing its precursor in the "modules" database table.

2.4 The *shinymgr* app builder

Once developers create and register their own stand-alone *shiny* modules, apps are generated with *shinymgr*'s app builder (Figure 5).

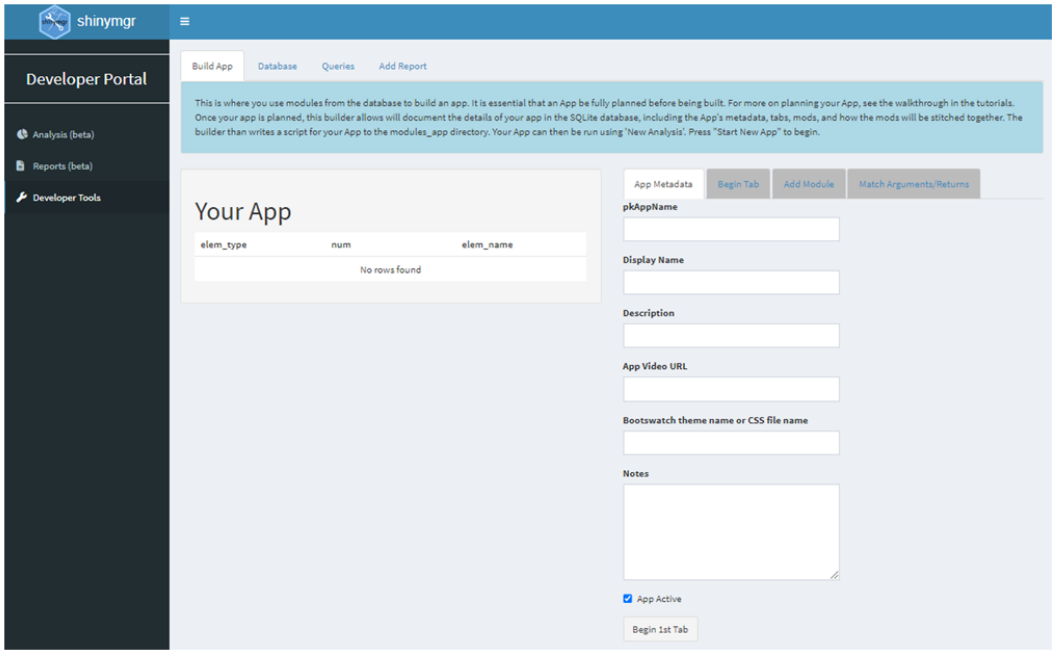


Figure 5: The shinymgr Developer Portal layout, showing the app builder in the Developer Tools.

Developers are guided through a process where they design their app from *shiny* modules they have registered. The builder then populates the *shinymgr* database with instructions on how to construct the app and writes the app's script based on those instructions. The newly created script is saved to the "modules_app" directory. Through this structured process, apps produced by the builder are well-documented and generate highly reproducible analyses. Readers are encouraged to peruse the tutorial, "apps", for more information.

The `qry_app_flow()` function will query the database to return a list of the *shiny* modules and tabs included in a specified app, such as "iris_explorer":

```
# look at the appTabs table in the database
qry_app_flow("iris_explorer", shinyMgrPath = paste0(getwd(),"/shinymgr"))

      fkAppName      fkTabName tabOrder       fkModuleName modOrder
1 iris_explorer       IE_intro        1         iris_intro        1
2 iris_explorer   IE_iris_data        2       iris_cluster        1
3 iris_explorer IE_subset_rows        3        subset_rows        1
4 iris_explorer   IE_plot_data        4 single_column_plot        1
```

As shown in Figure 2, this app has 5 tabs, and each tab features a single module. The "Save" tab is the final tab in all *shinymgr* apps and is not listed in the query result.

Developers can "beta test" apps prior to deployment by selecting the Analysis (beta) tab in the Developer's Portal (Figure 3). They can also create *RMarkdown* or *Quarto* report templates that accept the outputs from an analysis and incorporate them into a report. Report metadata are logged in the "reports" table of the database, and then linked with a specific app in the "appReports" table. An end-user will run an analysis and render a report, a process described more fully in the "Using *shinymgr* Apps" section below.

To summarize this section, developers use the `shinymgr_setup()` function to create the directory structure and underlying database needed to build and run *shiny* apps with *shinymgr*. Developers use the `mod_init()` and `mod_register()` functions to create modules and make them available for inclusion in new apps built with the *shinymgr* app builder. A developer can create as many *shinymgr* projects as needed. In each case, the *shinymgr* project is simply a fixed directory structure with three R files (ui.R, server.R, and global.R), and a series of subdirectories that contain the apps and *shiny* modules created by the developer, along with a database for tracking everything.

3 Deploying *shinymgr* projects

Once development is completed, developers can deploy their *shinymgr* project on a server or within an R package by copying portions of the *shinymgr* project to a new location while retaining the original project for future development. Once deployed, a *shinymgr* project no longer requires the *shinymgr* package or database to be run. Thus, the files and directories to be copied for deployment include only:

```
shinymgr
+-- data
+-- global.R
+-- modules
+-- modules_app
+-- modules_mgr
+-- reports
+-- server.R
+-- ui.R
\-- www
```

The master app files, ui.R, global.R, and server.R, are needed to run the *shinymgr* framework.

When deploying a *shinymgr* project within an R package, objects within the data folder should be copied into the package's "data" folder. The remaining files should be copied into a directory within the package's "inst" folder that will house the master *shiny* application. Deployment on a server such as shinyapps.io will require similar adjustments.

After files are copied to the correct location, a few key adjustments are needed. First, the "modules_app" directory should contain only those apps (and dependent modules and reports) that can be used by end-users; unused apps, modules, and reports can be deleted. Second, the new.analysis.R script within the modules_mgr folder will require minor updates to remove dependencies on the *shinymgr* database. Third, the ui.R and server.R scripts should be updated to no longer showcase *shinymgr* and the Developer's Portal; rather, it should be customized by the developer to create their own purpose-driven apps. For example, Figure 6 shows a hypothetical deployment of the master app titled "Deployed Project" that is based on the *shinymgr* framework. Notice the absence of the Developer Tools tab and the absence of references to *shinymgr*. The "deployment" *learnr* tutorial provides more in-depth discussion.

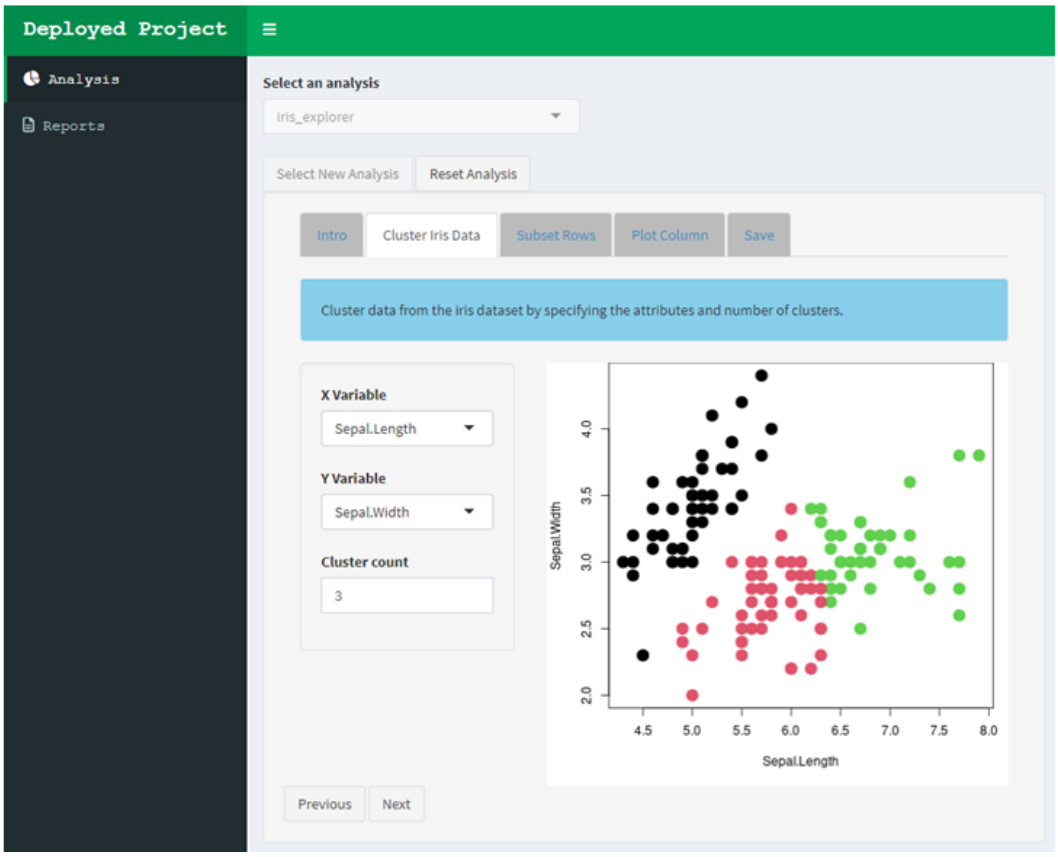


Figure 6: An example of a deployed shinymgr app. The deployed version excludes the Developers Tools tab and is an example of what the end user sees when using a deployed app.

To summarize this section, deploying the *shinymgr* framework involves copying key elements of the *shinymgr* developer project into package or server directories, updated as needed for use by end-users. Readers are referred to the "deployment" tutorial for further information.

4 Using *shinymgr* apps

Apps built with *shinymgr* can appeal to various types of end-users. When deployed as part of an R package, end-users would be anyone who uses that package. Apps may also be distributed as stand-alone scripts, or hosted on a server, as described above. Developers may also use *shinymgr* to produce apps for their own use (i.e., the developer *is* the end-user). Regardless of who the intended end-user is, this section discusses that user's experience after the master app is deployed.

Whoever the intended audience for the app, this section discusses how an app can be used *after* it has been deployed.

4.1 Reproducible analyses

The final tab in any *shinymgr* app provides the opportunity to save the analysis itself. Reproducibility is a core tenet of *shinymgr*. Therefore, a robust set of metadata are saved as an RDS file to allow a user to understand and replicate their results. An example of a completed analysis is the file, "iris_explorer_Gandalf_2023_06_05_16_30.RDS," which stores a user's analytic steps for a run of the "iris explorer" app. The code below reads in this example file, and shows the structure (a list with 23 elements):

```
rds_filepath <- paste0(getwd(),"/shinymgr/analyses/iris_explorer_Gandalf_2023_06_05_16_30.RDS")
old_analysis <- readRDS(rds_filepath)
str(old_analysis, max.level = 2, nchar.max = 20, vec.len = 15)

List of 23
```

```
$ analysisName                          : chr "iri"| __truncated__
$ app                                   : chr "iris_explorer"
$ username                              : chr "Gandalf"
$ mod2-clusters                         : int 3
$ mod2-xcol                             : chr "Sepal.Length"
$ mod2-ycol                             : chr "Petal.Length"
$ mod3-full_table__reactable__pageSize  : int 10
$ mod3-resample                         : 'shinyActionButtonValue' int 1
$ mod3-full_table__reactable__pages     : int 15
$ mod3-subset_table__reactable__page    : int 1
$ mod3-full_table__reactable__page      : int 1
$ mod3-sample_num                       : int 20
$ mod3-subset_table__reactable__pages   : int 2
$ mod3-subset_table__reactable__pageSize: int 10
$ returns                               :List of 3
 ..$ data1:List of 1
 ..$ data2:List of 1
 ..$ data3:List of 2
$ notes                                 : chr "Thi"| __truncated__
$ timestamp                             : POSIXct[1:1], format: "202"| __truncated__
$ metadata                              :List of 6
 ..$ appDescription: chr "Clu"| __truncated__
 ..$ mod1          :List of 7
 ..$ mod2          :List of 7
 ..$ mod3          :List of 7
 ..$ mod4          :List of 7
 ..$ lockfile      :List of 2
$ app_code                              : chr "# T"| __truncated__
$ iris_intro_code                       : chr "#!!"| __truncated__
$ iris_cluster_code                     : chr "#!!"| __truncated__
$ subset_rows_code                      : chr "#!!"| __truncated__
$ single_column_plot_code               : chr "#!!"| __truncated__
```

The list stores a great deal of information:

- **analysisName** is the name of the analysis and is equivalent to the filename of the RDS file (without the extension)
- **app** is the name of the app that produced the saved analysis results.
- **username** was entered in the "Save" tab when the analysis was performed.
- **mod#-value** indicate the values of each *shiny* module's arguments (inputs), if any exist, at the time the analysis was saved.
- **returns** includes values of all outputs (returns) of each module.
- **notes** were entered in the "Save" tab when the analysis was performed.
- **timestamp** is the date/time when the analysis was saved.
- **metadata** includes robust information about each module, including the app description and the description of each module as it was originally stored in the *shinymgr* database tables. The metadata list element also includes an *renv* "lockfile": a list that describes the R version and R package dependencies (including *shinymgr*) used by the app itself. The lockfile captures the state of the app's package dependencies at the time of its creation; in the case of *shinymgr*, it contains the dependencies used by the developer who created the app. Each lockfile record includes the name and version of the package and their installation source.
- ***_code** attributes with this format contain the source code for the app.

The code list element allows an end user to revisit the full analysis with *shinymgr*'s `rerun_analysis()` function, supplying the file path to a saved *shinymgr* analysis (RDS file).

```
rerun_analysis(analysis_path = rds_filepath)
```

The `rerun_analysis()` function will launch a *shiny* app with two tabs (Figure 7); it can only be run during an interactive R session, with no other *shiny* apps running.

The first tab is called "The App", and will be visible when the `rerun_analysis()` function is called. It contains a header with the app's name, a subheading of "Analysis Rerun," and a fully functioning, identical copy of the *shiny* app used to generate the saved analysis. Below that, a disclaimer appears, indicating the app was produced from a saved analysis. A summary of the analysis is presented on the second tab that displays the values used to produce the given analysis output.

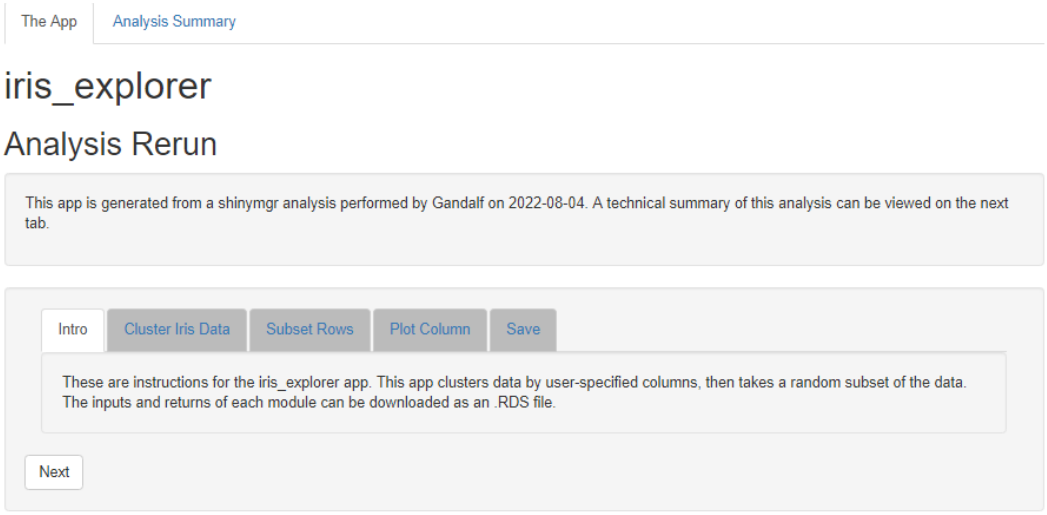


Figure 7: A screenshot of the rerun_analysis() function, as called on the saved analysis from the iris_explorer app (RDS file). The active tab, called 'The App', allows a user to rerun a previously executed analysis. The 'Analysis Summary' tab displays the values of all module arguments and returns, captured when the analysis was saved, along with a detailed description of the app, it's modules, the App's source code, and all package dependencies.

If the `rerun_analysis()` function fails, it could be due to a change in R and package versions currently installed on the end-user's machine. To that end, the lockfile that is included in the metadata section of the RDS file can be used to restore the necessary R packages and R version with the `restore_analysis()` function. This function will attempt to create a self-contained *renv* R project that includes all of the packages and the R version used by the developer when the app was created. The analysis RDS is added to this new project, where the `rerun_analysis()` function can be attempted again. Readers are referred to the "analyses" tutorial for further information.

4.2 Rapid reporting

Another important feature of *shinymgr* is the ability to share results of an analysis with others in a friendly, readable format with *RMarkdown* or *Quarto*. Apps produce an RDS file, which may be passed into an Rmd or qmd file as a parameterized input. For example, the demo database includes a report template called "iris_explorer_report.Rmd." This file, with code shown below, allows users to navigate to the RDS file produced by the "iris explorer" app and render the rapid report.

```
---
title: 'Annual Report for Iris Explorer'
output: html_document
params:
  user:
    label: "User"
    value: "Bilbo"
    placeholder: "Enter user name"
  year:
    label: "Year"
    value: 2017
    input: slider
    min: 2010
    max: 2018
    step: 1
    sep: ""
  file:
   input: file
   label: "Choose RDS"
   value: ""
   multiple: FALSE
```

```
   buttonLabel: "Browse to analysis output..."
---

```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = FALSE)
library(knitr)
ps <- readRDS(params$file)
```

This report summarizes an analysis of iris data by
`r params$user` conducted in `r params$year`. Iris
data was clustered into `r ps$'mod2-clusters'` groups
based on `r ps$'mod2-xcol'` and `r ps$'mod2-ycol'`.
A random sample of `r ps$'mod3-sample_num'` records
were collected, with sample sizes shown in the pie
chart below:

```{r}
pie_data <- table(ps$returns$data2$subset_data$cluster)
pie(
  x = pie_data,
  labels = as.character(pie_data),
  col = rainbow(length(pie_data)),
  main = "Number of random samples by cluster"
)
legend(
  x = "topright",
  legend = names(pie_data),
  fill = rainbow(length(pie_data))
)

```

Some things to note about this analysis are: `r ps$notes`

Respectfully submitted,

Gandalf
```

Reports may be run within the deployed version of *shinymgr* (e.g., left menu of Figure 6), or may be run directly in R by opening the Rmd file and navigating to the RDS as a file input. Users who run a report can download it to their local machine as a HTML, PDF, or Word file, where they can further customize the output.

To summarize this section, users of *shinymgr* "apps" created with the *shinymgr* framework are presented with a series of *shiny* tabs that establish an analysis workflow. Users can save their inputs and outputs as an RDS file to ensure full reproducibility. Further, the RDS file may be loaded into an R Markdown (Rmd) or Quarto (qmd) template for rapid reporting.

5 Tutorials and cheatsheet

with the package. Below is a list of current tutorials, intended to be worked through in order:

```
Available tutorials:
* shinymgr
  - intro            : "shinymgr-01: Introduction"
  - shiny            : "shinymgr-02: Shiny"
  - modules          : "shinymgr-03: Modules"
  - app_modules      : "shinymgr-04: App modules"
  - tests            : "shinymgr-05: Tests"
  - shinymgr         : "shinymgr-06: shinymgr"
  - database         : "shinymgr-07: Database"
  - shinymgr_modules : "shinymgr-08: shinymgr_modules "
  - apps             : "shinymgr-09: Apps"
```

```
  - analyses         : "shinymgr-10: Analyses"
  - reports          : "shinymgr-11: Reports"
  - deployment       : "shinymgr-12: Deployment"
```

The "intro" tutorial gives a general overview. Tutorials 2-5 are aimed at developers who are new to *shiny*, while tutorials 6 – 12 focus on the *shinymgr* package.

Launch a tutorial with the *learnr* `run_tutorial()` function, providing the name of the module to launch. The tutorial should launch in a browser, which has the benefit of being able to print the tutorial to PDF upon completion:

```
learnr::run_tutorial(
  name = "modules",
  package = "shinymgr")
```

Additionally, the package cheatsheet can be found with:

```
browseURL(paste0(find.package("shinymgr"), "/extdata/shinymgr_cheatsheet.pdf"))
```

Contributions are welcome from the community. Questions can be asked on the issues page at https://code.usgs.gov/vtcfwru/shinymgr/issues.

6 Acknowledgments

We thank Cathleen Balantic and Jim Hines for feedback on the overall package and package tutorials. *shinymgr* was prototyped by Therese Donovan at a *shiny* workshop taught by Chris Dorich and Matthew Ross at Colorado State University in 2020 (pre-pandemic). We thank the instructors for feedback and initial coding assistance. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government. The Vermont Cooperative Fish and Wildlife Research Unit is jointly supported by the U.S. Geological Survey, University of Vermont, Vermont Fish and Wildlife Department, and Wildlife Management Institute.

7 Bibliography

8 Appendix A

Entity relationship diagram for the *shinymgr* database, which tracks all components of an apps and modules (Figure 8). The database consists of 11 tables. Primary keys are referenced with a "pk" prefix, while foreign keys are referenced with an "fk" prefix. A full description of the database is contained in the "database" *learnr* tutorial that comes with the *shinymgr* package

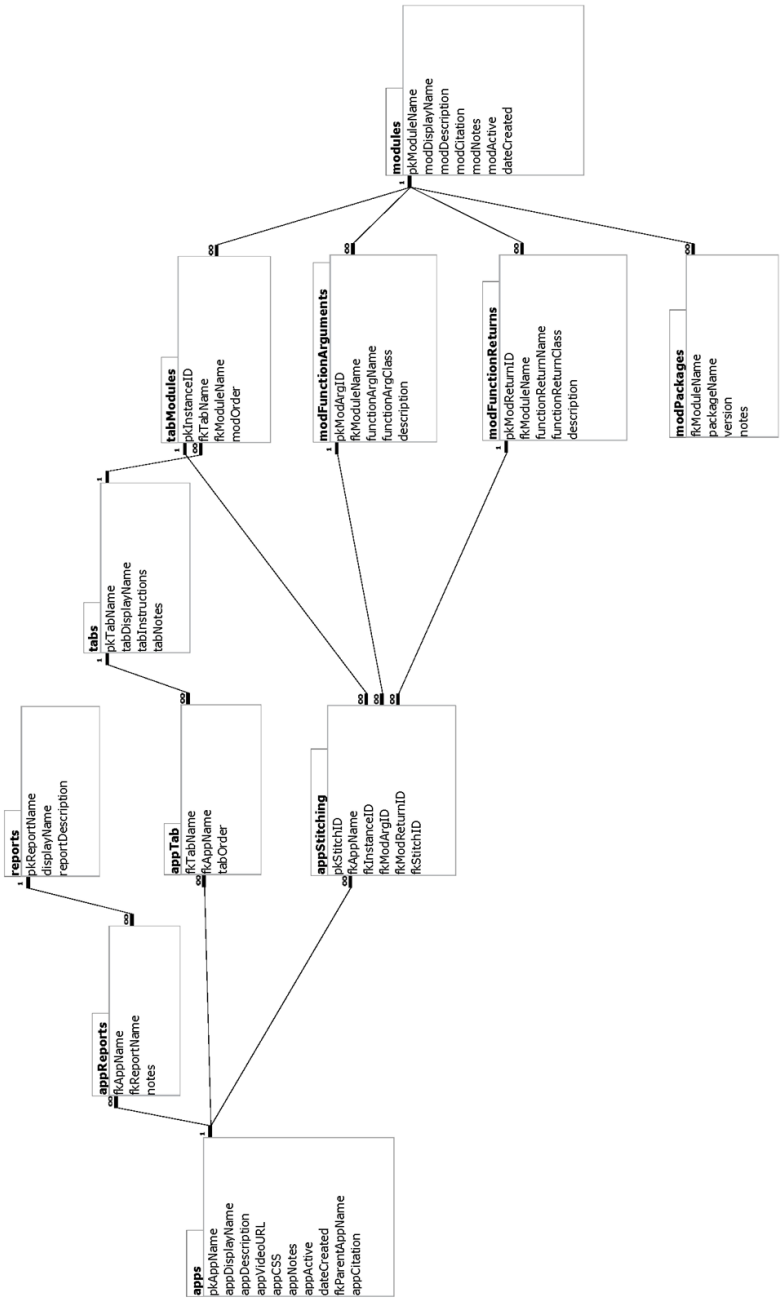


Figure 8: Entity relationship diagram for the shinymgr database, which tracks all components of an apps and modules. The database consists of 11 tables. Primary keys are referenced with a 'pk' prefix, while foreign keys are referenced with an 'fk' prefix. A full description of the database is contained in the 'database' learnr tutorial that comes with the shinymgr package.

9 Appendix B

Modules in *shinymgr* are written by developers for their own purposes. The `shinymgr::mod_init()` function creates a template for module development. The header is a series of key-value pairs that the developer fills out (typically after the module code is written and tested). The "iris_cluster" module is presented below as an example. The module consists of two paired functions: here, `iris_cluster_ui(id)` and `iris_cluster_server()`. The UI is a function with an argument called id, which is turned into module's "namespace" with the `NS()` function. A namespace is simply the module's identifier and ensures that function and object names within a given module do not conflict with function and object names in other modules. The Id's for each input and output in the UI must be wrapped in a `ns()` function call to make explicit that these inputs are assigned to the module's namespace. All UI elements are wrapped in a `tagList()` function, where a `tagList` allows one to combine multiple UI elements into a single R object. Readers should consult the "modules," "tests," and "shinymgr_modules" tutorials for additional information.

```
#!! ModName = iris_cluster
#!! ModDisplayName = Iris K-Means Clustering
#!! ModDescription = Clusters iris data based on 2 attributes
#!! ModCitation = Baggins, Bilbo.  (2022). iris_cluster. [Source code].
#!! ModNotes =
#!! ModActive = 1
#!! FunctionReturn = returndf !! selected attributes and their assigned clusters !! data.frame

iris_cluster_ui <- function(id){
  # create the module's namespace
  ns <- NS(id)

  tagList(
    sidebarLayout(
      sidebarPanel(
        # add the dropdown for the X variable
        selectInput(
          ns("xcol"),
          label = "X Variable",
          choices = c(
            "Sepal.Length",
            "Sepal.Width",
            "Petal.Length",
            "Petal.Width"
          ),
          selected = "Sepal.Length"
        ),

        # add the dropdown for the Y variable
        selectInput(
          ns("ycol"),
          label = "Y Variable",
          choices = c(
            "Sepal.Length",
            "Sepal.Width",
            "Petal.Length",
            "Petal.Width"
          ),
          selected = "Sepal.Width"
        ),
        # add input box for the cluster number

        numericInput(
          ns("clusters"),
          label = "Cluster count",
          value = 3,
          min = 1,
          max = 9
        )
```

```
      ), # end of sidebarPanel

      mainPanel(
        # create outputs
        plotOutput(
          ns("plot1")
        )
      ) # end of mainPanel
    ) # end of sidebarLayout
  ) # end of tagList
} # end of UI function

iris_cluster_server <- function(id) {

  moduleServer(id, function(input, output, session) {

    # combine variables into new data frame
    selectedData <- reactive({
      iris[, c(input$xcol, input$ycol)]
    })

    # run kmeans algorithm
    clusters <- reactive({
      kmeans(
        x = selectedData(),
        centers = input$clusters
      )
    })

    output$plot1 <- renderPlot({
      par(mar = c(5.1, 4.1, 0, 1))
      plot(
        selectedData(),
        col = clusters()$cluster,
        pch = 20,
        cex = 3
      )
    })

    return(
      reactiveValues(
        returndf = reactive({
          cbind(
            selectedData(),
            cluster = clusters()$cluster
          )
        })
      )
    )

  }) # end of moduleServer function

} # end of irisCluster function
```

References

Modularizing shiny app code, 2020. URL https://shiny.posit.co/r/articles/improve/modules/. Accessed: 2010-09-30. [p157]

J. M. Alston and J. A. Rick. A beginner's guide to conducting reproducible research. *The Bulletin of the Ecological Society of America*, 102(2):e01801, 2021. doi: https://doi.org/10.1002/bes2.1801. URL https://esajournals.onlinelibrary.wiley.com/doi/abs/10.1002/bes2.1801. [p157]

D. Attali. *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds*, 2021. URL https://CRAN.R-project.org/package=shinyjs. R package version 2.1.0. [p159]

C. Balantic and T. Donovan. Ammonitor: Remote monitoring of biodiversity in an adaptive framework with r. *Methods in Ecology and Evolution*, 11(7):869–877, 2020. doi: https://doi.org/10.1111/2041-210X.13397. [p159]

C. Brett and I. Neuhaus. *periscope: Enterprise Streamlined 'Shiny' Application Framework*, 2022. URL https://CRAN.R-project.org/package=periscope. R package version 1.0.1. [p157]

P. Campbell. *shinyauthr: 'Shiny' Authentication Modules*, 2021. URL https://CRAN.R-project.org/package=shinyauthr. R package version 1.0.0. [p157]

W. Chang and B. Borges Ribeiro. *shinydashboard: Create Dashboards with 'Shiny'*, 2021. URL https://CRAN.R-project.org/package=shinydashboard. R package version 0.7.2. [p159]

W. Chang, G. Csárdi, and H. Wickham. *shinytest: Test Shiny Apps*, 2021. URL https://CRAN.R-project.org/package=shinytest. R package version 1.5.1. [p161]

W. Chang, J. Cheng, J. Allaire, C. Sievert, B. Schloerke, Y. Xie, J. Allen, J. McPherson, A. Dipert, and B. Borges. *shiny: Web Application Framework for R*, 2022. URL https://CRAN.R-project.org/package=shiny. R package version 1.7.3. [p157, 159]

L. Clarfeld, C. Tang, and T. Donovan. *shinymgr: A framework for building, managing, and stitching shiny modules into reproducible workflows.*, 2024. R package version 1.1.0. [p159]

C. Fay and S. Rochette. *shinipsum: Lorem-Ipsum Helper Function for 'shiny' Prototyping*, 2020. URL https://cran.r-project.org/web/packages/shinipsum/index.html. R package version 0.1.0. [p157]

C. Fay, S. Rochette, V. Guyader, and C. Girard. *Engineering Production-Grade Shiny Apps*. Chapman and Hall/CRC, 2021. doi: https://doi.org/10.1201/9781003029878. [p157]

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2): 179–188, 1936. doi: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x. [p158]

R. Gentleman and D. T. Lang. Statistical analyses and reproducible research. *Journal of Computational and Graphical Statistics*, 16(1):1–23, 2007. doi: 10.1198/106186007X178663. URL https://doi.org/10.1198/106186007X178663. [p157]

R. D. Hipp. SQLite, 2020. URL https://www.sqlite.org/index.html. [p159]

J. Kim and H. Lee. *jsmodule: 'RStudio' Addins and 'Shiny' Modules for Medical Research*, 2022. URL https://CRAN.R-project.org/package=jsmodule. R package version 1.3.0. [p158]

M. Kosinski. *shiny.reglog: Optional Login and Registration Module System for ShinyApps*, 2022. URL https://statismike.github.io/shiny.reglog/. R package version 0.5.2. [p157]

C. Larman. *Agile and iterative development: a manager's guide*. Addison-Wesley Professional, 2004. [p157]

G. Lin. *reactable: Interactive Data Tables Based on 'React Table'*, 2022. URL https://CRAN.R-project.org/package=reactable. R package version 0.3.0. [p159]

K. Müller, H. Wickham, D. A. James, and S. Falcon. *RSQLite: SQLite Interface for R*, 2022. URL https://CRAN.R-project.org/package=RSQLite. R package version 2.2.14. [p159]

R. D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011. doi: 10.1126/science.1213847. URL https://www.science.org/doi/abs/10.1126/science.1213847. [p157]

V. Perrier, F. Meyer, and Z. S. Abeer. *datamods: Modules to Import and Manipulate Data in 'Shiny'*, 2022. URL https://CRAN.R-project.org/package=datamods. R package version 1.3.3. [p157]

R Special Interest Group on Databases (R-SIG-DB), H. Wickham, and K. Müller. *DBI: R Database Interface*, 2022. URL https://CRAN.R-project.org/package=DBI. R package version 1.1.3. [p159]

B. Schloerke, J. Allaire, and B. Borges. *learnr: Interactive Tutorials for R*, 2020. URL https://CRAN.R-project.org/package=learnr. R package version 0.10.1. [p159]

S. Stoudt, V. N. Vásquez, and C. C. Martinez. Principles for data analysis workflows. *PLOS Computational Biology*, 17(3):e1008770, 2021. doi: https://doi.org/10.1371/journal.pcbi.1008770. [p157]

K. Ushey. *renv: Project Environments*, 2023. URL https://rstudio.github.io/renv/. R package version 0.17.3. [p159]

H. Wickham. testthat: Get started with testing. *The R Journal*, 3:5–10, 2011. URL https://journal.r-project.org/archive/2011-1/RJournal_2011-1_Wickham.pdf. [p161]

K. Żyła, J. Nowicki, L. Siemiński, M. Rogala, R. Vibal, and T. Makowski. *rhino: A Framework for Enterprise Shiny Applications*, 2023. https://appsilon.github.io/rhino/, https://github.com/Appsilon/rhino. [p157]

Laurence A. Clarfeld
Vermont Cooperative Fish and Wildlife Research Unit
302 Aiken Center, University of Vermont
Burlington, VT 05405 USA
ORCiD: 0000-0002-3927-9411
laurence.clarfeld@uvm.edu

Caroline Tang
Queen's University
Biology Department
116 Barrie St, Kingston, ON K7L 3N6
ORCiD: 0000-0001-7966-5854
17ct24@queensu.ca

Therese Donovan
U.S. Geological Survey, Vermont Cooperative Fish and Wildlife Research Unit
302 Aiken Center, University of Vermont
Burlington, VT 05405 USA
ORCiD: 0000-0001-8124-9251
tdonovan@uvm.edu

Bayesian Model Selection with Latent Group-Based Effects and Variances with the R Package slgf

by Thomas A. Metzger and Christopher T. Franck

Abstract Linear modeling is ubiquitous, but performance can suffer when the model is misspecified. We have recently demonstrated that latent groupings in the levels of categorical predictors can complicate inference in a variety of fields including bioinformatics, agriculture, industry, engineering, and medicine. Here we present the R package slgf which enables the user to easily implement our recently-developed approach to detect group-based regression effects, latent interactions, and/or heteroscedastic error variance through Bayesian model selection. We focus on the scenario in which the levels of a categorical predictor exhibit two latent groups. We treat the detection of this grouping structure as an unsupervised learning problem by searching the space of possible groupings of factor levels. First we review the suspected latent grouping factor (SLGF) method. Next, using both observational and experimental data, we illustrate the usage of slgf in the context of several common linear model layouts: one-way analysis of variance (ANOVA), analysis of covariance (ANCOVA), a two-way replicated layout, and a two-way unreplicated layout. We have selected data that reveal the shortcomings of classical analyses to emphasize the advantage our method can provide when a latent grouping structure is present.

1 Introduction

Linear models with categorical predictors (i.e., factors) are pervasive in the social, natural, and engineering sciences, among other fields. Conventional approaches to fit these models may fail to account for subtle latent structures, including latent regression effects, interactions, and heteroscedasticity within the data. These latent structures are frequently governed by the levels of a factor. Several examples of such datasets can be found in Franck et al. (2013), Franck and Osborne (2016), Kharrati-Kopaei and Sadooghi-Alvandi (2007), and Metzger and Franck (2021). Our recent work (Metzger and Franck, 2021) developed latent grouping factor-based methodology to detect latent structures using Bayesian model selection. The current work provides an overview of the **slgf** package that enables users to easily implement the suspected latent grouping factor (SLGF) methodology, and expands on the previous work by allowing for more flexible model specification.

Consider Figure 1, which illustrates four relevant data sets analyzed in this paper. In each panel, the levels of a user-specified factor are found to exhibit a latent grouping structure that partitions the data into two groups with distinct regression effects (indicated by color-coding) and/or error variances (filled and open geometry). With the **slgf** package, the user specifies the factor suspected of governing this latent structure. The package protects the user against detecting spurious latent grouping structures since it can accommodate non-grouped candidate models. It can also accommodate additional linear model terms of interest. The **slgf** package then assesses the plausibility of each model and the corresponding structures via Bayesian model selection. An overview of **slgf** functionality for these data follows and full details of each analysis (including candidate models) appear in Section **Using the slgf package**. The **slgf** package focuses on assessing the plausibility of two-group structures in linear models with categorical predictors using fractional Bayes factors. A discussion comparing **slgf** and other R packages that address latent group models is in Section **Conclusion**.

The top left panel of Figure 1 represents a one-way analysis of variance (ANOVA) study where a continuous measurement of olfactory function (vertical axis) is modeled as a function of age, where age is a factor represented in five categories (horizontal axis) (O'Brien and Heft, 1995). We find the highest posterior model probability (61%) for the model where levels 1, 2, and 3 of the SLGF age have distinct mean effects and error variances from levels 4 and 5. We call this the `smell` data set.

The top right panel shows an analysis of covariance (ANCOVA), where the breaking strength of a starch film (vertical axis) is measured as a function of the SLGF (starch type) and a continuous measurement of film thickness (horizontal axis) (Furry, 1939). We find the highest posterior model probability (59%) for the model where potato starch (unshaded gray squares) have a larger error variance than the shaded points, and, the red points (canna and corn starch) have a distinct slope from the gray points. We call this the `textile` data set.

The bottom left panel shows the example described by Meek and Ozgur (1991), where the torque required to tighten a locknut (vertical axis) was measured as a function of a plating process and a threading technique. The plating processes analyzed included treatments with cadmium and wax

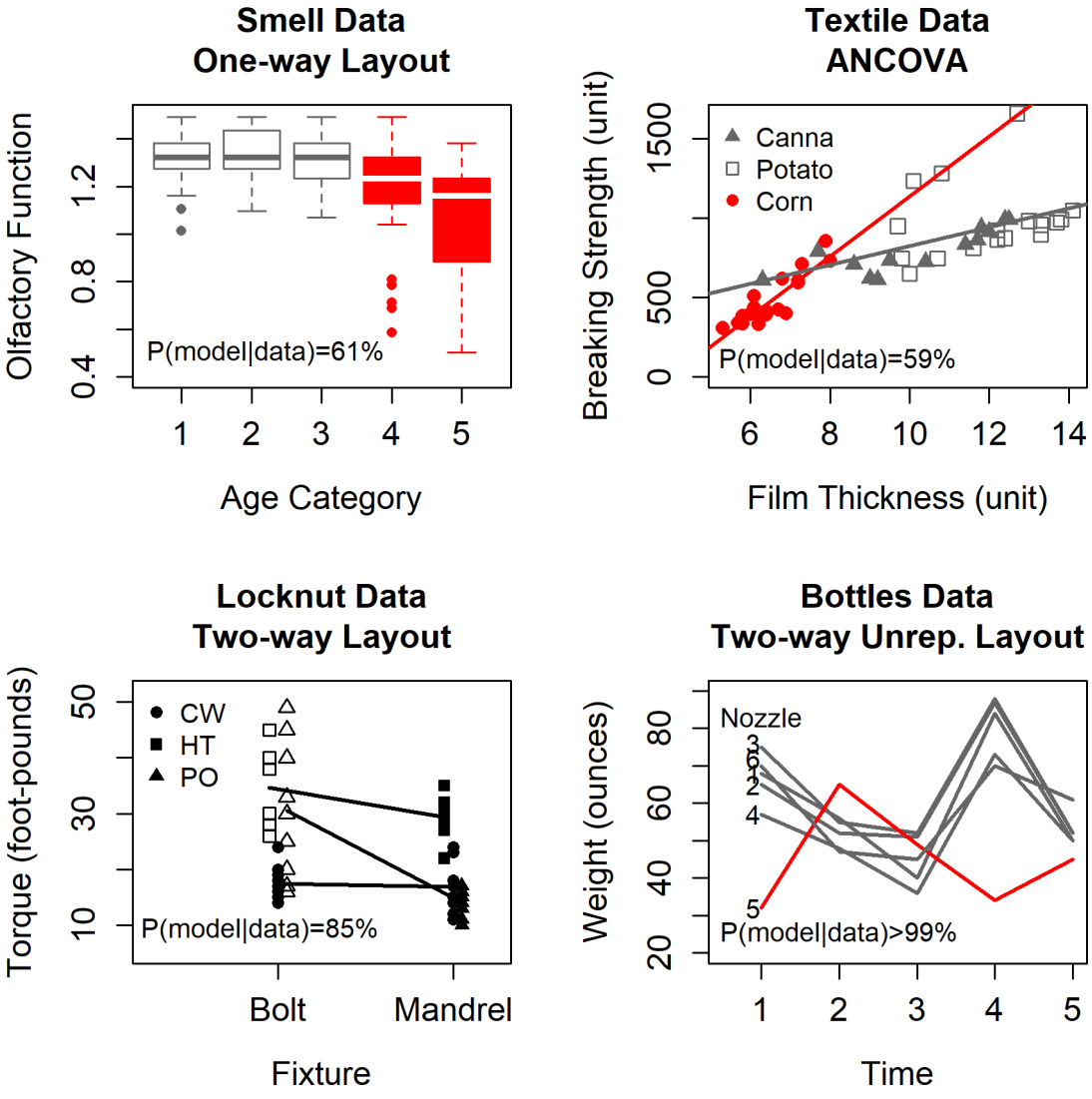


Figure 1: Smell data (O'Brien and Heft, 1995, top left), textile data (Furry, 1939, top right), locknut data (Meek and Ozgur, 1991, bottom left), and bottles data (Ott and Snee, 1973, bottom right). Color (red/gray) shows latent grouping structure (i.e., group-based regression effects) for smell, textile, and bottles data, and fill (solid/open geometry) shows group-based variances for smell, textile, and locknut data.

(CW), heat treating (HT), and phosphate and oil (PO). The threading techniques studied include bolt and mandrel, the types of fixture on which each locknut was affixed to conduct the test. We find the highest posterior model probability (85%) for the model where bolt by HT and bolt by PO measurements have a larger error variance than those from bolt by CW, mandrel by HT, mandrel by PO, and mandrel by CW. We call this the `locknut` data.

Finally, in the bottom right panel, the data set of Ott and Snee (1973) represents an unreplicated two-way layout where six machine nozzles were used to fill bottles on five occasions (horizontal axis). The weight of each bottle (vertical axis) was measured, and we find the highest posterior model probability for the structure where nozzle 5 is found to be out of alignment from the others ($> 99\%$). We call this the `bottles` data.

The **slgf** package implements a combinatoric approach that evaluates all possible assignments of SLGF levels into two groups. We refer to each these assignments as *schemes*. For example, in the `smell` data, the scheme that is visualized assigns age levels 1, 2, and 3 into one group and levels 4 and 5 into the other, denoted {1,2,3}{4,5}. More details on how schemes are established can be found in Subsection Grouping schemes and model classes.

The user may specify an SLGF for regression effects, another SLGF for error variances, require them to be the same, or specify no SLGF for one or both of these. For example, the `smell` data has age as the SLGF for both. In Subsection **Case study 2: textile data**, we analyze a data set with distinct regression and error variance SLGFs.

In this paper, we provide an overview of the **slgf** package that enables analysis of data sets like those in Figure 1 via Bayesian model selection. In Section **SLGF methodology**, we briefly review the SLGF methodology. In Section **Using the slgf package**, we illustrate the package functionality for the four data sets illustrated in Figure 1. For each data set, we will demonstrate the relevant code and package functionality along with a comparison between the results of a classical approach and our approach. In Section **Conclusion**, we summarize the package and its functionality.

2 SLGF methodology

2.1 Model specification

For a thorough review of the SLGF model specification see Metzger and Franck (2021). First consider the linear model

$$\boldsymbol{Y} = \mathbf{1}^T\alpha + \boldsymbol{W\nu} + \boldsymbol{V\tau} + \boldsymbol{U\rho} + \boldsymbol{\varepsilon}, \tag{1}$$

where 1^T is an $N \times 1$ vector of 1s, α is an intercept, $\boldsymbol{\nu}$ represents the full SLGF effect with K degrees of freedom, $\boldsymbol{\tau}$ represents the regression effects that do not arise from latent groupings (i.e., all other regression effects of interest), and the $\boldsymbol{\rho}$ terms indicate statistical interactions between SLGF and other regression effects; W, V, and U partition the overall model matrix into model matrices corresponding to the SLGF effects $\boldsymbol{\rho}$, additional effects $\boldsymbol{\tau}$, and SLGF interactions, respectively; and finally $\boldsymbol{\varepsilon}$ represents an $N \times 1$ vector of errors where $\boldsymbol{\varepsilon} \overset{\text{iid}}{\sim} N(0, \Sigma)$ for $\Sigma = \sigma^2 I$ where I is an $N \times N$ identity matrix.

Because a central goal of the SLGF methodology is to compare models with and without latent grouping structures, we next develop notation to indicate whether model terms in Equation (1) involve groupings of factor levels or not. If a model contains a one degree of freedom group effect instead of the full K degree of freedom SLGF effect, we denote the effect $\tilde{\nu}$ instead, with corresponding $\tilde{W}$ to ensure they remain conformable. Similarly, if the interaction $\boldsymbol{\rho}$ is with the group effect rather than the full SLGF effect, we denote it $\tilde{\boldsymbol{\rho}}$. When there are group-based error variances, we let $\tilde{\varepsilon}$ denote the vector of heteroscedastic errors, where the elements of $\tilde{\varepsilon}$ are either $N(0, \sigma_1^2)$ or $N(0, \sigma_2^2)$ depending on their membership in group 1 or 2, respectively.

For example, for the `smell` data in the top left panel of Figure 1, the most probable model can be represented as $Y = 1^T\alpha + \tilde{W}\tilde{\nu} + \tilde{\varepsilon}$, with a 1 degree of freedom group effect $\tilde{\nu}$ (color-coding) and heteroscedastic error term $\tilde{\varepsilon}$ (shading). This model (posterior model probability 0.65) was found to be far more probable than the ordinary one way analysis of variance model $Y = 1^T\alpha + W\boldsymbol{\nu} + \boldsymbol{\varepsilon}$ (posterior model probability less than 0.0001), the model with a 4 degree of freedom mean effect $\boldsymbol{\nu}$ and homoscedastic errors $\boldsymbol{\varepsilon}$. Similarly, the bottles data (bottom right panel) most probable model is $Y = 1^T\alpha + W\boldsymbol{\nu} + \tilde{U}\tilde{\boldsymbol{\rho}} + \boldsymbol{\varepsilon}$ with a 4 degree of freedom nozzle effect $\boldsymbol{\nu}$, an 8 degree of freedom group-by-nozzle interaction $\tilde{\boldsymbol{\rho}}$, and homoscedastic errors $\boldsymbol{\varepsilon}$.

2.2 Grouping schemes and model classes

Recall schemes are the possible assignments of factor levels to two latent groups. While the schemes shown in Figure 1 may seem visually obvious, the **slgf** package considers all possible such assignments of factor levels into two groups. This (i) obviates the need for the user to specify specific schemes, and (ii) apportions prior model probabilities commensurately with the actual number of models corresponding to a SLGF to prevent detection of spurious latent grouping structure. Problems will differ in the number of schemes under consideration. The package **slgf** automatically determines the schemes once the set of candidate models has been established by the user. The minimum number of levels that can comprise a grouping scheme can be adjusted by the user to lower the number of candidate models or to avoid creating model effects with too few degrees of freedom to be estimated. The user may specify the SLGF for regression effects and/or error variances, or neither. These SLGFs may or may not be different factors. If they are the same, the user may require that the grouping schemes must be equal or that they may be distinct. For example, in the textile data in the top right panel of Figure 1, the SLGF is starch for both regression effects and error variances, but the user should allow for distinct schemes since the variance scheme appears to be {potato}{canna,corn} and the regression effect scheme appears to be {corn}{canna,potato}.

A model *class* describes the structure of the model including specification of effects related to the hidden groups. Model classes may include, for example, the set of models with group-based regression effects but no group-based variances; or, a single model with no group-based regression effects or variances. For example, in the `smell` data represented in top left panel of Figure 1, we consider the following 62 models comprising six model classes:

1. A single model with a 1 degree of freedom global mean effect and homoscedastic error variance;
2. A single model with a 4 degree of freedom mean effect and homoscedastic error variance;
3. 15 models (corresponding to the 15 possible grouping schemes) with a 1 degree of freedom global mean effect and group-based heteroscedastic error variances;
4. 15 models with a 4 degree of freedom mean effect and group-based heteroscedastic error variances;
5. 15 models with a 1 degree of freedom group-based mean effect and homoscedastic error variance;
6. 15 models with a 1 degree of freedom group-based mean effect and group-based error variances.

For our analysis, we specified that the regression effect and variance grouping schemes must be equivalent, and that one level of the age factor could comprise a group. The user can relax these specifications as desired.

2.3 Parameter priors

With **slgf**, the user can choose to implement noninformative priors on the regression effects (default), or the Zellner-Siow mixture of g-priors on these effects. We first enumerate the noninformative priors. Let $\boldsymbol{\beta}$ represent the full set of regression effects. For simplicity, we parametrize on the precision scale where $\varphi = \frac{1}{\sigma^2}$ and the corresponding precision matrix $\varphi I_{n\times n}$ is denoted Φ. For a model m_s^c where c indexes class and s indexes grouping scheme, **slgf** imposes

$$P(\boldsymbol{\beta}, \varphi | m_s^c) \propto \varphi \tag{2}$$

for homoscedastic models, and

$$P(\boldsymbol{\beta}, \varphi_1, \varphi_2 | m_s^c) \propto \varphi_1 \cdot \varphi_2 \tag{3}$$

for heteroscedastic models.

Alternatively, in contexts with limited data, such as the two-way unreplicated `bottles` data in the bottom right panel of Figure 1, we recommend employing the Zellner-Siow mixture of g-prior (Zellner and Siow, 1980; Zellner, 1986; Liang et al., 2008), which reduces the minimal training sample size necessary for the computation of the fractional Bayes factor (see Subsection **Fractional Bayes factors and posterior model probabilities** for further detail). We have generally found that in cases where the number of data points is close to the number of parameters in some of the larger candidate models (e.g., case study 4, bottles data), the mixture of g-priors approach outperforms the noninformative priors due to the drastic reduction in the required proportion of the data needed to implement the fractional Bayes factor approach. For homoscedastic models, recall $\Phi = \phi I$ where I is an $N \times N$ identity matrix. Let

$$P(\alpha, \varphi | m_s^c) \propto \varphi \tag{4}$$

and

$$\boldsymbol{\beta}_{-\alpha}|\Phi, g, m_s^c \sim N(\mathbf{0},\, g(X^T\Phi^{-1}X)^{-1}). \tag{5}$$

Next, for heteroscedastic models, first denote $\tilde{\Phi}$ as a diagonal precision matrix where the ith diagonal element is either φ_1 or φ_2, depending upon the grouping membership of the ith observation. Let

$$P(\alpha, \varphi_1, \varphi_2|m_s^c) \propto \varphi_1 \cdot \varphi_2 \tag{6}$$

and

$$\boldsymbol{\beta}_{-\alpha}|\tilde{\Phi}, g, m_s^c \sim N(\mathbf{0},\, g(X^T\tilde{\Phi}^{-1}X)^{-1}); \tag{7}$$

In both homoscedastic and heteroscedastic cases,

$$g \sim \text{InvGamma}(\frac{1}{2},\, \frac{N}{2}). \tag{8}$$

Thus for homoscedastic models, the full prior on all parameters is the product of Equations (4), (5), and (8). For heteroscedastic models, it is the product of Equations (6), (7), and (8).

2.4 Fractional Bayes factors and posterior model probabilities

Note that if we form a standard Bayes factor for models using improper priors on parameters, the unspecified proportionality constants associated with the improper priors (Equations 2, 3, 4, and 6) would not cancel one another and the Bayes factor would be defined only up to an unspecified constant. Thus we invoke a fractional Bayes factor approach (O'Hagan, 1995) to compute well-defined posterior model probabilities for each model. More details follow.

The **slgf** package obtains posterior model probabilities through the use of fractional Bayes factors. Briefly, a Bayes factor is defined as the ratio of two models' integrated likelihoods. The integrated likelihood is obtained by integrating parameters out of the joint distribution of data and parameters. In some cases, this integration is analytic, but in others, it is undertaken with a Laplace approximation; the corresponding simplified expressions and methods used to optimize them are described in detail later in this section. In the SLGF context, let $\mathcal{M}$ represent the full set of models under consideration, representing all classes and grouping schemes of interest. Denote $\boldsymbol{\theta}$ as the full set of unknown parameters associated with a model $m_s^c \in \mathcal{M}$ and $\pi(\boldsymbol{\theta}|m_s^c)$ as the prior on these parameters given model m_s^c. The parameter vector $\boldsymbol{\theta}$ depends on class and scheme of model m_s^c. The integrated likelihood is

$$P(\boldsymbol{Y}|m_s^c) = \int_{\boldsymbol{\Theta}} P(\boldsymbol{Y}|\boldsymbol{\theta}, m_s^c)\pi(\boldsymbol{\theta}|m_s^c)d\boldsymbol{\theta},$$

with Bayes factor comparing models m_s^c and $m_{s'}^{c'}$

$$BF = \frac{P(\boldsymbol{Y}|m_s^c)}{P(\boldsymbol{Y}|m_{s'}^{c'})}.$$

Since the priors used by the **slgf** package are improper, $\pi(\boldsymbol{\theta}|m_s^c)$ is defined only up to an unspecified constant. Thus, BF is defined only up to a ratio of unspecified constants. To overcome this issue and enable improper priors on parameters to be used in the course of Bayesian model selection, the fractional Bayes factor (O'Hagan, 1995) was developed. A fractional Bayes factor is a ratio of two fractional marginal model likelihoods, where a fractional marginal likelihood is defined as

$$q^b(\boldsymbol{Y}|m_s^c) = \frac{\int P(\boldsymbol{Y}|\boldsymbol{\theta}, m_s^c)\pi(\boldsymbol{\theta}|m_s^c)d\boldsymbol{\theta}}{\int P(\boldsymbol{Y}|\boldsymbol{\theta}, m_s^c)^b\pi(\boldsymbol{\theta}|m_s^c)d\boldsymbol{\theta}}. \tag{9}$$

The $q^b(\boldsymbol{Y}|m_s^c)$ quantity in Equation (9) is the integrated likelihood based on the $1-b$ fraction of the data where the improper prior has been updated to become proper with b fraction of the data. Thus all normalizing constants are specified. The fractional Bayes factor is thus

$$FBF = \frac{q^b(\boldsymbol{Y}|m_s^c)}{q^b(\boldsymbol{Y}|m_{s'}^{c'})}.$$

for some fractional exponent $0 < b < 1$. Thus we must compute the integrals $\int P(\boldsymbol{Y}|\boldsymbol{\theta}, m_s^c)\pi(\boldsymbol{\theta}|m_s^c)d\boldsymbol{\theta}$ and $\int P(\boldsymbol{Y}|\boldsymbol{\theta}, m_s^c)^b\pi(\boldsymbol{\theta}|m_s^c)d\boldsymbol{\theta}$, the numerator and denominator of Equation (9), respectively, for all $m_s^c \in \mathcal{M}$. Although O'Hagan (1995) provides several recommendations for choice of b, **slgf** exclusively

implements $b = \frac{m_0}{N}$ where m_0 is the minimal training sample size required for the denominator of Equation (9) to be proper for all models. If m_0 is too small, then the denominator of Equation (9) diverges. The user must specify m_0; if their choice is too low, then **slgf** increases it until all relevant integrals converge. For further details, see O'Hagan (1995), p. 101; for recommendations on choosing m_0 in practice, see Subsection **Choice of m_0**.

Next we discuss the technical details on how these integrals are computed via Laplace approximation. Specifically, we will describe how $\log\left(q^b(\boldsymbol{Y}|m_s^c)\right)$ is computed in each case. In the case of noninformative regression priors for homoscedastic models, $\boldsymbol{\beta}$ and σ^2 are integrated analytically. Let $\hat{\boldsymbol{Y}}$ represent the fitted values of m_s^c and SSResid the residual sum of squares of this model. We obtain

$$\log\left(q^b(\boldsymbol{Y}|m_s^c)\right) = \left(-\frac{N(1-b)}{2}\right)(\log\pi + \log(\text{SSResid})) + \left(\frac{Nb-1}{2}\right)\log b + \log\left(\frac{\Gamma\left(\frac{N-P}{2}\right)}{\Gamma\left(\frac{Nb-P}{2}\right)}\right) \tag{10}$$

In the case of noninformative regression priors for heteroscedastic models, both the numerator and denominator integrals of Equation (9)must be approximated with a Laplace approximation because although $\boldsymbol{\beta}$ can be integrated analytically, σ_1^2 and σ_2^2 cannot be. The integrals are computed on the log-scale for numeric stability. Equation (9) on the log-scale simplifies to:

$$\log\left(q^b(\boldsymbol{Y}|m_s^c)\right) = \frac{N(b-1)}{2}\log(2\pi) + \frac{P+1}{2}\log b + \frac{1}{2}\log\left(\frac{|H_b^\star|}{|H^\star|}\right) + \log\left(\frac{P(\boldsymbol{Y}|\boldsymbol{\theta}^\star)\pi(\boldsymbol{\theta}^\star|m_s^c)}{P(\boldsymbol{Y}|\boldsymbol{\theta}_b^\star)^b\pi(\boldsymbol{\theta}_b^\star|m_s^c)}\right) \tag{11}$$

where $\boldsymbol{\theta}^\star$ and $H^\star$ denote the mode and Hessian of $P(\boldsymbol{Y}|\boldsymbol{\theta}, m_s^c)\pi(\boldsymbol{\theta}|m_s^c)$, and $\boldsymbol{\theta}_b^\star$ and $H_b^\star$ denote the mode and Hessian of $P(\boldsymbol{Y}|\boldsymbol{\theta}, m_s^c)^b\pi(\boldsymbol{\theta}|m_s^c)$. These modes and Hessians are computed with **optim** using the Nelder-Mead algorithm.

In the Zellner-Siow mixture of g-prior case, α and $\boldsymbol{\beta}_{-\alpha}$ are integrated analytically. For homoscedastic models, σ^2 is as well, and only g is integrated with a Laplace approximation. Again marginal model likelihoods are computed on the log-scale. The log of the mode of $P(\boldsymbol{Y}|g, m_s^c)^b\pi(\boldsymbol{\theta}|m_s^c)$, denoted $g_b^\star$, is found by solving the closed-form equation $\frac{(Nb-1-P)}{2}\log(1+bg) + \frac{Nb-1}{2}\log(1+bg(1-R^2)) - \frac{3}{2}\log g - \frac{N}{2g} := 0$ with the base **R** function **uniroot** where R^2 is the coefficient of determination for m_s^c. The Hessian is then evaluated at this solution $g_b^\star$; the closed-form Hessian of $P(\boldsymbol{Y}|g, m_s^c)^b\pi(\boldsymbol{\theta}|m_s^c)$ evaluated at $g^\star$ is given by $H_b^\star = \frac{1}{2}\left(\frac{((Nb-1)b^2(1-R^2)^2}{(1+bg^\star)(1-R^2)^2} - \frac{(Nb-P-1)b^2}{(1+bg^\star)^2} + \frac{3}{g^{\star 2}} - \frac{2N}{g^{\star 3}}\right)$. For $b = 1$, this expression describes the numerator of Equation (9); see Liang et al. (2008) for further mathematical details. The Laplace approximation for Equation (9) on the log-scale then is given by:

$$\begin{aligned}\log\left(q^b(\boldsymbol{Y}|m_s^c)\right) = &\log\left(\frac{\Gamma\left(\frac{N-1}{2}\right)}{\Gamma\left(\frac{Nb-1}{2}\right)}\right) + \frac{Nb-1}{2}(\log(\text{SSTotal}) + \log\pi) + \frac{1}{2}\log\left(\frac{|H_b^\star|}{|H^\star|}\right) + \\ &\log\left(\frac{P(\boldsymbol{Y}|\boldsymbol{\theta}^\star)\pi(\boldsymbol{\theta}^\star|m_s^c)}{P(\boldsymbol{Y}|\boldsymbol{\theta}_b^\star)^b\pi(\boldsymbol{\theta}_b^\star|m_s^c)}\right).\end{aligned} \tag{12}$$

For heteroscedastic models, a three-dimensional Laplace approximation is used to integrate σ_1^2, σ_2^2, and g. To obtain $\boldsymbol{\theta}_b^\star$ and $\boldsymbol{\theta}^\star$, we first transform $\gamma_1 = \log\left(\frac{1}{\sigma_1^2}\right)$ and $\gamma_2 = \log\left(\frac{1}{\sigma_2^2}\right)$ to stabilize the optimization. We optimize $\log P(\boldsymbol{Y}|g, \sigma_1^2, \sigma_2^2)^b\pi(\sigma_1^2, \sigma_2^2, g) = \frac{n_1 b}{2}\log\gamma_1 + \frac{n_2 b}{2}\gamma_2 - \frac{P}{2}\log g + \frac{1}{2}|X^T\tilde{\Sigma}X| - \frac{1}{2}\log|\frac{bg+1}{bg}X^T(\tilde{\Sigma} - Z_{\tilde{\Sigma}})X| - \frac{b}{2}\boldsymbol{Y}^T\left(\tilde{\Sigma} - Z_{\tilde{\Sigma}} - (\tilde{\Sigma} - Z_{\tilde{\Sigma}})X\left(\frac{bg+1}{bg}X^T\tilde{\Sigma}X - X^T Z_{\tilde{\Sigma}}X\right)^{-1}X^T(\tilde{\Sigma} - Z_{\tilde{\Sigma}})\right)\boldsymbol{Y} - \frac{3}{2}\log(g) - \frac{N}{2g} + \log(J)$ using the Nelder-Mead method from **optim** where $Z_{\tilde{\Sigma}} = \tilde{\Sigma}Z(Z^T\tilde{\Sigma}Z)^{-1}Z^T\tilde{\Sigma}$, $Z = \mathbf{1}^T$, and $\log(J) = -(\gamma_1 + \gamma_2)$ represents the determinant of the log-precision transformation. For $b = 1$ these equations yield integrand of the numerator of (9).

With the modes computed, the Hessians of $\log P(\boldsymbol{Y}|g, \sigma_1^2, \sigma_2^2)^b\pi(\sigma_1^2, \sigma_2^2, g)$ are calculated with the function **Hessian** from the package **numDeriv**. Finally with the modes and Hessians computed, the Laplace approximation for Equation (9) is given by:

$$\log\left(q^b(\boldsymbol{Y}|m_s^c)\right) = \frac{Nb-1}{2}\log(2\pi) + \frac{P+1}{2}\log(b) + \frac{1}{2}\log\left(\frac{|H_b^\star|}{|H^\star|}\right) + \log\left(\frac{P(\boldsymbol{Y}|\boldsymbol{\theta}^\star)\pi(\boldsymbol{\theta}^\star|m_s^c)}{P(\boldsymbol{Y}|\boldsymbol{\theta}_b^\star)^b\pi(\boldsymbol{\theta}_b^\star|m_s^c)}\right). \tag{13}$$

For the sake of consistency, all models, even with fully tractable marginal model likelihoods, are computed with a FBF. Once log-fractional marginal likelihoods have been computed for all models, we subtract the maximum from this set so that the set of log-fractional marginal likelihoods has been rescaled to have a maximum of 0. Each value is exponentiated to obtain a set of fractional marginal likelihoods with maximum 1. This adjustment helps to avoid numerical underflow when computing posterior model probabilities.

2.5 Choice of $\mathbf{m_0}$

The user must specify the argument m_0, the minimal training sample size such that all marginal model likelihoods are well-defined. If `prior="flat"`, then we recommend that the user begins by letting `m0` equal the dimension of the improper priors: that is, the number of coefficients in most complex model under consideration plus the number of variances under consideration. If `prior="zs"`, then `m0` can generally be much smaller (in practice, we have found that `m0=2` performs well) as the prior on the regression effects is proper. If the user's choice is too low, then `ms_slgf` will incrementally increase it by 1 until all marginal model probabilities are numerically stable. If `m0` reaches n, corresponding to 100% of data used for training, `ms_slgf` will terminate and the user should specify a different set of models.

2.6 Model priors

With this adjusted set of fractional marginal likelihoods, we next consider the priors for the model space. The function `ms_slgf` imposes a uniform prior by model class, and for classes containing multiple models, the prior on each class is uniformly divided among the models it contains. We finally compute posterior model probabilities for each model:

$$P(m'|\boldsymbol{Y}) = \frac{P(\boldsymbol{Y}|m')P(m')}{\sum_{\mathcal{M}} P(\boldsymbol{Y}|m)P(m)}. \tag{14}$$

The prior probability placed on each model can be found in the `models$ModPrior` vector in output from `ms_slgf`.

2.7 Parameter estimation

Our approach provides maximum *a posteriori* (MAP) estimates for all relevant quantities: $\hat{\boldsymbol{\beta}}$, $\hat{\sigma}^2 = \{\hat{\sigma}^2\}$ or $\hat{\boldsymbol{\sigma}}^2 = \{\hat{\sigma}_1^2, \hat{\sigma}_2^2\}$ in the homoscedasitc and heteroscedastic cases respectively, and g in the Zellner-Siow mixture of g-prior case.

Because the prior on $\boldsymbol{\beta}$ is either flat or centered at 0, the MAP estimator is simply the usual maximum likelihood estimator:

$$\hat{\boldsymbol{\beta}} = \arg\max_{\boldsymbol{\beta}} P(\boldsymbol{Y}|X, \boldsymbol{\beta}, \Sigma) \tag{15}$$

so that $\hat{\boldsymbol{\beta}} = (X^TX)^{-1}X^T\boldsymbol{Y}$. The variance(s) and g were computed via the base **R** function **optim** during the Laplace approximation stage. For computational efficiency, $\boldsymbol{\beta}$ is integrated out of $P(\boldsymbol{Y}|X, \boldsymbol{\theta})P(\boldsymbol{\theta})$ and the variances are estimated on the log-scale, so we let $\hat{\boldsymbol{\lambda}} := \{\hat{\lambda}\}$ in homoscedastic models or $\{\hat{\lambda}_1, \hat{\lambda}_2\}$ in heteroscedastic models. Then

$$\hat{\boldsymbol{\lambda}} = \arg\max_{\boldsymbol{\lambda}} \int P(\boldsymbol{Y}|X, \boldsymbol{\beta}, \Sigma)P(\boldsymbol{\beta})P(\Sigma)d\boldsymbol{\beta} \tag{16}$$

or,

$$\{\hat{\boldsymbol{\lambda}}, \hat{g}\} = \arg\max_{\boldsymbol{\lambda}, g} \int P(\boldsymbol{Y}|X, \boldsymbol{\beta}, \Sigma, g)P(\alpha)P(\boldsymbol{\beta}_{-\alpha}|\Sigma)P(g)d\boldsymbol{\beta}. \tag{17}$$

Then, $\hat{\sigma}^2 = \exp\{\hat{\boldsymbol{\lambda}}\}$ for $\hat{\sigma}^2 = \{\hat{\sigma}^2\}$ or $\hat{\boldsymbol{\sigma}}^2 = \{\hat{\sigma}_1^2, \hat{\sigma}_2^2\}$. The output values `coefficients`, `variances`, and `gs` (only if `prior="zs"`) are lists where each element contains the estimates for each model's $\hat{\boldsymbol{\beta}}$, $\hat{\boldsymbol{\sigma}}^2$, and $\hat{\mathbf{g}}$, respectively.

3 Using the slgf package

The function `ms_slgf()` is the main function of **slgf** that implements the methodology we have described. Each argument of `ms_slgf()` and its output will be illustrated in the case studies found in Subsections **Case study 1: smell data**, **Case study 2: textile data**, **Case study 3: locknut data**, and **Case study 4: bottles data**. The `ms_slgf()` function requires several inputs to compute and output posterior model probabilities for all models, schemes, and model classes of interest. The user begins with a `data.frame` containing a continuous response, at least one categorical predictor, and any other covariates of interest. The `data.frame` cannot contain column names with the character string `group`, because `ms_slgf()` will search for this string when fitting group-based models. The user must first identify an SLGF for the fixed effects and/or the variance. The user indicates, via the arguments `response`, `slgf_beta`, and `slgf_Sigma`, character strings corresponding to the response, the suspected latent fixed effect grouping factor, and the suspected latent variance grouping factors, respectively. If no latent regression effect structure or variance structure is to be considered, the user may specify `slgf_beta=NA`, `slgf_Sigma=NA`, or both. We note that if the user does not specify any SLGFs, the model selection is still undertaken through fractional Bayes factors as described previously. If the user chooses the same categorical variable for both latent grouping factors, the argument `same_scheme`, which defaults to `FALSE`, can indicate whether the grouping schemes for the regression effect and variance structures must be equivalent.

Next the user determines the model classes they wish to evaluate. The argument `usermodels` is a `list` where each element contains a string of R class `formula` or `character`. The user also specifies which classes should also be considered in a heteroscedastic context via the argument `het`, which is a vector of the same length as `usermodels`, containing an indicator 1 or 0 corresponding to each model class specified in `usermodels` where 1 indicates the model will be considered with group-based variances and 0 indicates it will not. Together the arguments `usermodels` and `het` indicate which fixed effect structures are of interest, and which should be further considered for heteroscedasticity, thus implicitly creating the full set of model classes considered.

Next the user chooses a prior to place on the regression effects. As described in Subsection **Parameter priors**, `prior="flat"` (the default) implements the noninformative prior and `prior="zs"` imposes the Zellner-Siow mixture of g-prior.

Finally the user must specify the minimum number of levels of the SLGF that can comprise a group, via the arguments `min_levels_beta` and `min_levels_Sigma`, which default to 1. The number of possible grouping schemes increases with the number of levels of the SLGF. To speed up the computation, the user can increase these arguments and thus reduce the number of candidate models. Because we partition into two groups, note these arguments may not exceed half the number of levels of the SLGF. Additionally, when considering data with limited degrees of freedom, increasing `min_levels_beta` and/or `min_levels_Sigma` may be necessary to ensure effects can be computed.

3.1 Case Study 1: smell data

First we revisit the smell data set analyzed by O'Brien and Heft (1995). They measured olfactory acuity (denoted `olf`) on a continuous scale as a function of age (`agecat`), where age groups were divided into five categorical levels. See Figure 2. We note that levels 4 and 5 of `agecat` appear to have larger variance than levels 1, 2, and 3, but standard analysis of variance models assume homoscedasticity. We first demonstrate how a classical analysis might misrepresent the data. A usual one-way ANOVA analysis compares the null model, with a single mean, against the alternative model, with 4 degrees of freedom for the mean effects, with homoscedastic error variance.

```
 % remove smell null model
> smell$agecat <- as.factor(smell$agecat) # coerce agecat to a factor variable
> smell_null <- lm(olf~1, data=smell) # fit a null model with a single mean
> smell_full <- lm(olf~agecat, data=smell) # fit a full model with a 4 agecat effects
> print(smell_null)
Call:
lm(formula = olf ~ 1, data = smell)

Coefficients:
(Intercept)
      1.234
> print(smell_full)
Call:
lm(formula = olf ~ agecat, data = smell)
```

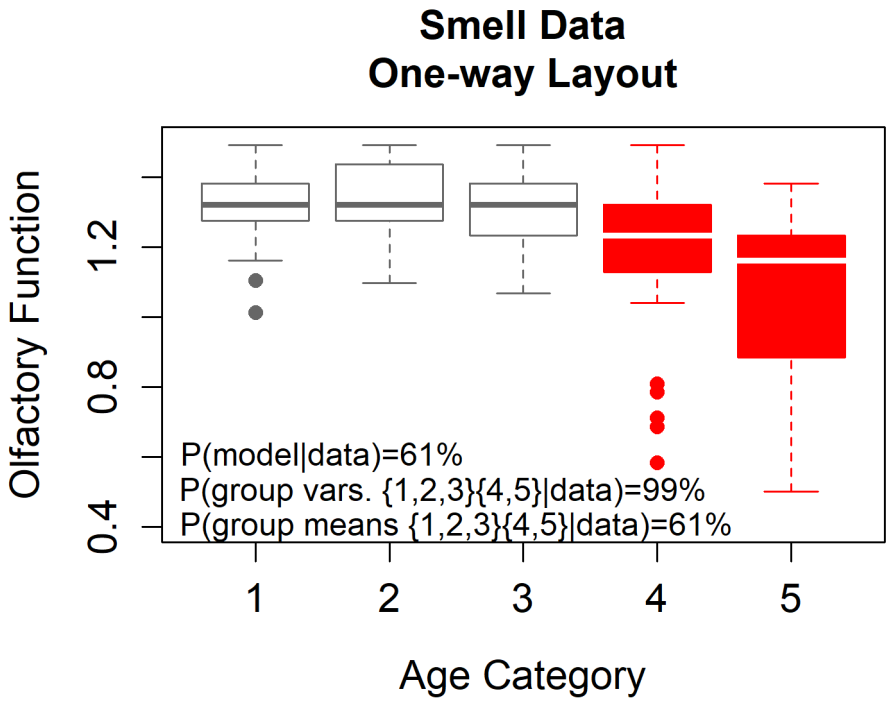


Figure 2: The smell data (O'Brien and Heft, 1995) is analyzed for group-based means and variances. We find posterior model probability of 61% for the model with group-based means and variances with scheme {1,2,3}{4,5}. We also find overall posterior probability of grouping scheme

.

```
Coefficients:
(Intercept)      agecat2      agecat3      agecat4      agecat5
    1.31689      0.02824     -0.01075     -0.11580     -0.25728
> anova(smell_null, smell_full) # compare the null and full models
Analysis of Variance Table

Model 1: olf ~ 1
Model 2: olf ~ agecat
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1    179 7.7585
2    175 5.6197  4    2.1388 16.651 1.395e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
> summary(smell_null)$sigma^2
0.04334349
> summary(smell_full)$sigma^2
0.03211259
```

This approach, which assumes all levels of `agecat` have equal error variance, favors the model with a 4 degree of freedom `agecat` effect. Note we obtain maximum likelihood estimates for the error variance of $\hat{\sigma}^2_{\text{full}} = 0.03211$. Based on Figure 2, we suspect this value may overestimate the error variance for levels 1, 2, and 3, while underestimating that of levels 4 and 5. We also suspect that the full model may be overly complex, as the means for levels 1, 2, and 3 appear to be plausibly equivalent. That is, the apparent latent grouping scheme for both regression effects and error variances is {1,2,3}{4,5}, or equivalently, {4,5}{1,2,3}.

Next, consider the **slgf** approach. We will consider the classes of models with group-based means, group-based variances, and both group-based means and variances. We specify `dataf=smell` and `response="olf"`, along with `slgf_beta="agecat"` and `slgf_Sigma="agecat"` as the suspected latent grouping factor for both regression effects and variances. We set the minimum number of levels for a group to 1 with `min_levels_beta=1` and `min_levels_Sigma=1`. Note that fewer grouping

schemes would be considered if we let these arguments equal 2. For simplicity, since the mean and variance grouping schemes both visually appear to be {1,2,3}{4,5}, we will restrict the schemes to be equivalent with `same_scheme=TRUE`. Via the `usermodels` argument, we will consider the null model class `olf~1`, the full model class `olf~agecat`, and the group-means model class `olf~group`, which will automatically consider all possible grouping schemes. Similarly, we will consider each of these formulations with the class of both homoscedastic and group-based variances via the argument `het=c(1,1,1)`. With a relatively large amount of data, we will use the uninformative `prior="flat"`. Finally we specify a minimal training sample size of `m0=9`, although if we specify this value to be too small, `ms_slgf()` will automatically increase it to the smallest value for which the relevant integrals converge and/or the necessary optimizations can be performed. We run `ms_slgf` to obtain the posterior model probabilties for all 62 models under consideration. We inspect the two most probable models, with indices 62 and 32, which comprise over 99% of the posterior probability over the model space considered:

```
> smell_out <- ms_slgf(dataf=smell, response="olf", lgf_beta="agecat",
                       min_levels_beta=1, lgf_Sigma="agecat",
                       min_levels_Sigma=1, same_scheme=TRUE,
                       usermodels=list("olf~1", "olf~agecat", "olf~group"),
                       het=c(1,1,1), prior="flat", m0=9)
> smell_out$models[c(1,2),c(1,2,3,5,7)]
        Model  Scheme.beta Scheme.Sigma  FModProb Cumulative
62  olf~group {4,5}{1,2,3} {4,5}{1,2,3} 0.6054935  0.6054935
32 olf~agecat         None {4,5}{1,2,3} 0.3878754  0.9933688
```

The most probable model, as suspected, is `olf~group`, indicating group-based means where `Scheme.beta` is {4,5}{1,2,3}. Note also `Scheme.Sigma` indicates group-based heteroscedasticity with the same scheme. This model received posterior probability of approximately 61%. The next most probable model also has group-based heteroscedasticity with scheme {4,5}{1,2,3}, but note the model is `olf~agecat`, containing the full model not with group-based mean effects, but rather 4 degrees of freedom for the `agecat` effect. By inspecting `smell_out$scheme_probabilities_Sigma`, we see that models with variance grouping scheme {4,5}{1,2,3} comprise over 99% of the posterior probability. By contrast, the models with fixed effect grouping scheme {4,5}{1,2,3} (that is, the homoscedastic and heteroscedastic versions) comprise 61% of the posterior probability. We find these posterior probabilities intuitive, easy to interpret quantifications of uncertainty.

The output fields `coefficients` and `variances` contain lists with the coefficients and variance(s) associated with each model. The output field `model_fits` contains the output from a linear model fit to the model specification in question, containing the , and Note the most probable model has index `62`, so we inspect the 62nd elements of the coefficient and variance lists `smell_out$coefficients` and `smell_out$variances`, which contain the MAP estimates for each model's regression effects and variance(s), respectively. The group-based variance estimates are $\hat{\sigma}^2_{\{4,5\}} = 0.0587$ and $\hat{\sigma}^2_{\{1,2,3\}} = 0.0121$. We contrast these variances against the estimate $\hat{\sigma}^2_{\text{full}} = 0.032$, which appears to have overestimated the variance of levels 1, 2, and 3, while simultaneously underestimating that of levels 4 and 5.

```
> smell_out$coefficients[[62]]
(Intercept)  group{4,5}
  1.3252211  -0.1940328
> smell_out$variances[[62]]
     {4,5}    {1,2,3}
0.05868885 0.01211084
```

3.2 Case study 2: textile data

We reanalyze the breaking strength data set of Furry (1939), also investigated by Metzger and Franck (2021), to illustrate the additional flexibility of **slgf** beyond the original work. The breaking strength of a starch film `strength` (measured in grams) is analyzed according to the thickness of the film, denoted `film` (measured in 10^{-4} inches), and the type of starch `starch` used to create the film (canna, corn, or potato). As usual, we begin by plotting the data to ascertain whether there is a latent grouping factor present. By inspection we note that the potato films, represented by squares in Figure 3, appear to have a higher variability than the corn (filled red circles) and canna (filled gray triangles) films.

We first illustrate a typical ANCOVA approach, in which three parallel lines for each level of starch are fit with a common error variance. This model leads to the fit shown in the center panel of Figure 3. Note only the film thickness effect is statistically significant according to a traditional hypothesis testing approach with $\alpha = 0.05$. The residual standard error of this model is $\hat{\sigma}^2_{\text{ANCOVA}} = 27126.09$.

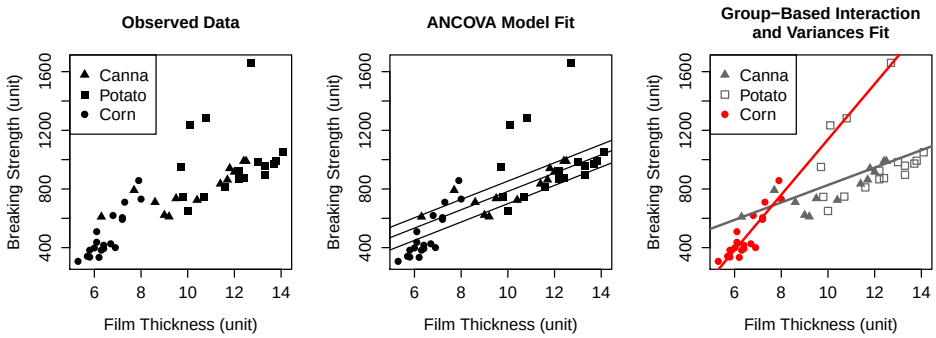


Figure 3: The breaking strength data set from Furry (1939) represents the breaking strength of starch films depending on the thickness of a film coating and the type of starch used to make the film. The left panel shows the data. The center panel shows an ANCOVA model. The right panel shows the most probable model ($P(m|\boldsymbol{Y}) \approx 66\%$) containing a latent group-based interaction between groups {canna, potato}{corn} (gray points vs. red points) and film thickness, as well as distinct variances between groups {canna, corn}{potato} (filled points vs. open points).

```
> textile_ancova <- lm(strength~film+starch, data=textile)
> summary(textile_ancova)

Call:
lm(formula = strength ~ film + starch, data = textile)

Residuals:
    Min      1Q  Median      3Q     Max
-203.63  -99.45  -57.84   56.72  637.61

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    158.26     179.78   0.880 0.383360
film            62.50      17.06   3.664 0.000653 ***
starchcorn     -83.67      86.10  -0.972 0.336351
starchpotato    70.36      67.78   1.038 0.304795
```

We contrast these findings against our methodology with **slgf**. The following arguments are input: `dataf=textile` specifies the data frame; `response="strength"` specifies the column of `textile` that contains the response variable; `slgf_beta="starch"` and `slgf_Sigma="starch"` indicate that the categorical variable `starch` should be used as the latent grouping factor for both regression effects and variances; `same_scheme=FALSE` indicates that the latent regression effect and variance grouping structures do not need to be partitioned by the same levels of `starch`; `min_levels_beta=1` and `min_levels_Sigma=1` indicate that a latent group can contain only one level of `starch`; the user-models argument indicates that we will consider main effects models `strength~film+starch` and `strength~film+starch+film*starch`, and models with group-based regression effects including `strength~film+group` and `strength~film+group+film*group`; the argument `het=c(1,1,1,1)` indicates that each of these four model specifications will also be considered with group-based variances; `prior="flat"` places a flat prior on the regression effects; and `m0=8` specifies the minimal training sample size.

```
> data(textile)
> out_textile <- ms_slgf(dataf = textile, response = "strength",
              lgf_beta = "starch", lgf_Sigma = "starch",
              same_scheme=FALSE, min_levels_beta=1, min_levels_Sigma=1,
              usermodels = list("strength~film+starch", "strength~film*starch",
                                "strength~film+group", "strength~film*group"),
              het=c(1,1,1,1), prior="flat", m0=8)
> out_textile$models[1:5,c(1,2,3,5)]
                  Model          Scheme.beta         Scheme.Sigma      FModProb
31  strength~film*group {corn}{canna,potato} {potato}{canna,corn}  0.6596667376
8  strength~film*starch                 None {potato}{canna,corn}  0.3337588991
30  strength~film*group {canna}{corn,potato} {potato}{canna,corn}  0.0018692078
```

```
28  strength~film*group {corn}{canna,potato} {corn}{canna,potato}  0.0010854755
7  strength~film*starch                 None {corn}{canna,potato}  0.0006831597
```

Refer to code and output above, where we provide the five most probable models. Note the three most probable models all have the latent variance grouping scheme {potato}{canna, corn}; again over 99% of the posterior model probability is accounted for by this variance scheme. This visually agrees with the plot, which shows that the potato starch films seem to have higher variability than the canna and corn starch films. The regression effect structure is less clear: the most probable model selects the `film*group` model, which contains main effects for `film` and `group` as well as their interaction, with scheme {canna}{corn, potato}. We plot this model in the right panel of Figure 3 to illustrate its plausibility. It does appear that the slope for corn is steeper than that of potato and canna, which can be contracted into a single level to simplify the model. However, the error variance for potato appears larger than that of canna and potato, as evidenced by the large spread of square potato points around the gray line. Thus we assert that the most probable model under our methodology is reasonable and appropriate. The group standard errors are $\sigma^2_{\{\text{potato}\}} = 57734.046$ and $\sigma^2_{\{\text{canna,corn}\}} = 5791.713$, indicating the standard ANCOVA model underestimates the error variance of the potato observations, and overestimates those of the canna and corn observations.

Finally we illustrate the output `scheme_probabilities_beta` and `scheme_probabilities_Sigma`, which sum up the probabilities for all model specifications associated with each possible grouping scheme. We see moderately high cumulative probability for models with regression grouping scheme {corn}{canna,potato}, followed closely be models with no grouping scheme for regression effects:

```
> out_textile$scheme_probabilities_beta
           Scheme.beta  Cumulative
2 {corn}{canna,potato} 0.592860983
4                 None 0.403632744
1 {canna}{corn,potato} 0.002502435
3 {potato}{canna,corn} 0.001003838
```

Intuitively, based on the wider spread of the square potato points in Figure 3, we see high cumulative probability for the variance grouping scheme {potato}{canna,corn}:

```
> out_textile$scheme_probabilities_Sigma
          Scheme.Sigma   Cumulative
3 {potato}{canna,corn} 9.975853e-01
2 {corn}{canna,potato} 2.184257e-03
1 {canna}{corn,potato} 2.304323e-04
4                 None 1.632320e-08
```

3.3 Case study 3: locknut data

We consider the two-way replicated layout of Meek and Ozgur (1991), where the torque (`torque`) required to tighten a locknut was measured as a function of a plating process (`plating`) and a threading method (`fixture`).

A two-way analysis with an interaction yields the following ANOVA table. The fixture and plating main effects, along with fixture by plating interaction, are all statistically significant at level $\alpha = 0.005$. Additionally, we find $\hat{\sigma}^2_{\text{Full}} = 36.58$:

```
> anova(lm(Torque~Fixture+Plating+Fixture*Plating, data=locknut))
Analysis of Variance Table

Response: Torque
               Df Sum Sq Mean Sq F value    Pr(>F)
Fixture         1  821.4  821.40 22.4563 1.604e-05 ***
Plating         2 2290.6 1145.32 31.3118 9.363e-10 ***
Fixture:Plating  2  665.1  332.55  9.0916 0.0003952 ***
Residuals       54 1975.2   36.58
```

Upon inspection of Figure 4, we suspect that two latent characteristics are at play. First, based on the non-parallel lines representing the plating effects, there may be a group-by-plating interaction, so we will consider `slgf_beta="Plating"`. Note since fixture has only two levels, it is not feasible to consider group-based effects based on fixture since the one degree of freedom fixture effect would be equivalent to a group effect.

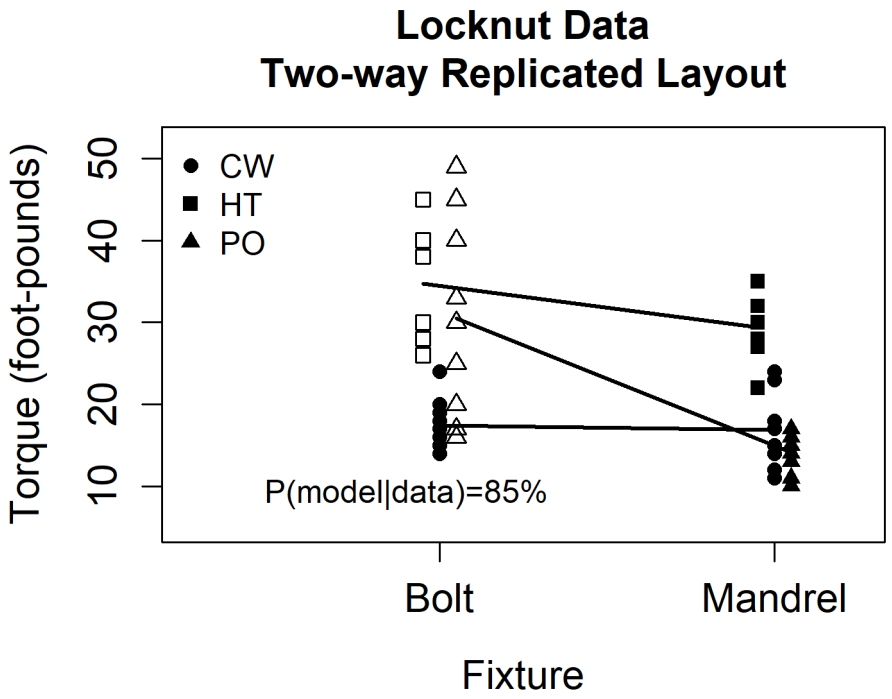


Figure 4: The most probable model ($P(m|Y) \approx 85\%$) contains a full fixture by plating interaction effect with no grouping structure, and group-based variances based on the levels of this interaction with scheme {bolt*CW, mandrel*CW, mandrel*HT, mandrel*PO}{bolt*HT, bolt*PO} (filled points vs. open points).

Regarding the variance structure, the variance of the torque amount at levels PO and HT appears higher, but only for the bolt fixture. This suggests that the levels of the interaction govern the variance groups; that is, `slgf_Sigma="Fixture*Plating"`. Since this specific variable header does not appear in the `locknut` data set, we manually create a new variable with each interaction level by pasting together the main effect variables:

```
locknut$Interaction <- paste0(locknut$Fixture, "*", locknut$Plating)
```

Thus we consider the following model specifications. Liang et al. (2008) (p. 420) note that the Zellner-Siow mixture of g-prior provides a fully Bayesian, consistent model selection procedure for small n along with relatively straightforward expressions for the marginal model probabilities. This approach is implemented by the user with the argument `prior="zs"`:

```
> data(locknut)
> locknut$Interaction <- paste0(locknut$Fixture, "*", locknut$Plating)
> out_locknut <- ms_slgf(dataf=locknut, response="Torque", same_scheme=FALSE,
                       lgf_beta="Plating", min_levels_beta=1,
                       lgf_Sigma="Interaction", min_levels_Sigma=1,
                       usermodels=list("Torque~Fixture+Plating+Fixture*Plating",
                                       "Torque~Fixture+group+Fixture*group"),
                       het=c(1,1), prior="zs", m0=2)
```

This formulation favors the same main and interaction effects favors by the standard model. However, **slgf** favors group-based variances with scheme {bolt*HT, bolt*PO}{bolt*CW, mandrel*CW, mandrel*HT, mandrel*PO} with posterior probability of approximately 85%. This variance structure was expected based on the relatively larger spread of the open points in Figure 4. As we have noted previously, the group variance estimates show that the heteroscedastic model overestimates the variance for some levels of fixture and plating, and underestimates it for others. Since model '13' was the model probable model, we print these variances, obtaining $\hat{\sigma}^2_{\text{bolt*HT,bolt*PO}} \approx 85.0$ and $\hat{\sigma}^2_{\text{bolt*CW,mandrel*CW,mandrel*HT,mandrel*PO}} \approx 11.6$:

```
> out_locknut$variances[[13]]
{bolt*HT,bolt*PO} {bolt*CW,mandrel*CW,mandrel*HT,mandrel*PO}
         85.00448                                   11.58652
```

3.4 Case study 4: bottles data

Finally, we consider the data of Ott and Snee (1973), where a machine with six heads (`head`) is designed to fill bottles (`weight`). The weight of each bottle is measured once over five time points (`time`) as a two-way unreplicated layout. A visual inspection of the data (Figure 5, left panel) indicates that one of the filling heads is behaving distinctly than the other five. There appears to be an interaction between head and time, but we lack the degrees of freedom to fit such a model. If we were to fit the standard main effects model, we obtain the clearly inappropriate model fit in the center panel of Figure 5.

Since it appears that head {5} is out of calibration in some way as compared to heads {1,2,3,4,6}, we instead consider the group-based interaction model `weight~time+group:time` where 'head' is the regression effect SLGF. For this illustration, we consider only homoscedastic models. In this data-poor context, we recommend the use of the Zellner-Siow mixture of g-prior by specifying `prior="zs"` in the `ms_slgf` function. The minimal training sample size can be much lower, as this prior is proper. We inspect the posterior model probabilities of the most probable model and the additive main effects model:

```
> bottles_me <- lm(weight~time+heads, data=bottles)
> bottles2 <- data.frame(weight=bottles$weight, time=as.factor(bottles$time),
                         heads=as.factor(bottles$heads))
> bottles_out <- ms_slgf(dataf=bottles2, response="weight", lgf_beta="heads",
                  min_levels_beta=1, lgf_Sigma=NA, min_levels_Sigma=NA, same_scheme=FALSE,
                  usermodels=list("weight~time+group:time",  "weight~time+heads"),
                  het=c(0,0), prior="zs", m0=2)
> bottles_out$models[1:2,c(1,2,4,5)]
                    Model    Scheme.beta Log-Marginal  FModProb
5 weight~time+group:time {5}{1,2,3,4,6}     -103.168    0.9991932
32     weight~heads+time           None     -114.726 0.0002158313
```

The group-based approach overwhelmingly favors the model with a main effect for 'time' along with the group-based interaction 'group:time' with scheme {5}{1,2,3,4,6}. We also note that the error variance for the main effects model is $\hat{\sigma}^2_{\text{ME}} = 130.1233$, while the estimate for the group-based interaction model is $\hat{\sigma}^2_{\{5\}\{1,2,3,4,6\}} = 39.76$, suggesting the main effects model seriously overestimates the error variance and thus may lead to misleading inference on regression effects.

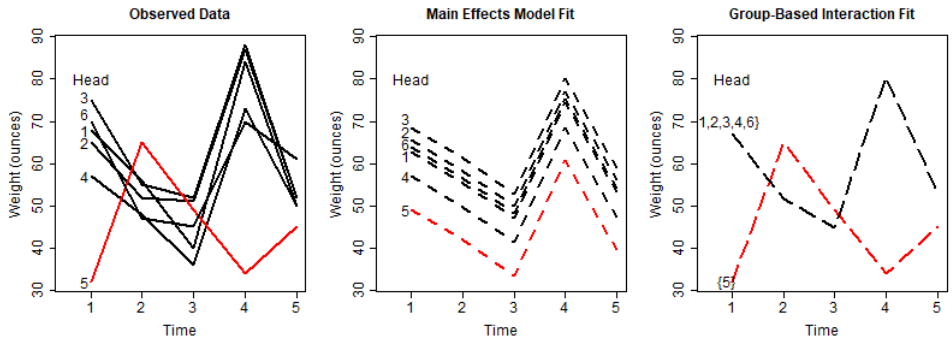


Figure 5: The bottles data set from Ott and Snee (1973) represents fill weights by six machine heads over five time points. The left panel shows the data, with head 5 appearing to be out of calibration. The center panel shows a main effects model, with a realistic fit for heads 1, 2, 3, 4, and 6, but not 5. The right panel shows the most probable group-based interaction ($P(m|\boldsymbol{Y}) > 99.9\%$) with main effects for time and a group-by-time interaction with scheme {5}{1,2,3,4,6}.

We note that there will be a linear dependency between the group-by-time interaction and the time main effect for time 5. The `NA` values can be seen by inspecting the coefficients of the corresponding model. These effects are not counted in the dimensionality of the model when computing $q^b(\boldsymbol{Y}|m)$.

```
> bottles_out3$coefficients[[5]]
          (Intercept)              heads2              heads3              heads4
                53.24                1.80                4.80               -6.80
              heads5              heads6  group{1,2,3,4,6}:time1   group{5}:time1
               -8.24               -1.00               14.00              -13.00
group{1,2,3,4,6}:time2   group{5}:time2  group{1,2,3,4,6}:time3   group{5}:time3
```

```
                -1.40             20.00                  -8.20             4.00
group{1,2,3,4,6}:time4  group{5}:time4  group{1,2,3,4,6}:time5  group{5}:time5
                27.40            -11.00                     NA               NA
```

4 Conclusion

This manuscript has provided an overview of the **slgf** package in R, which is available from the Comprehensive R Archive Network. Source code can be found on Github at https://github.com/metzger181osu/slgf. The **slgf** package allows the user to determine whether latent groupings of categorical predictor's levels provide a better characterization of the response variable compared with ordinary linear models that do not account for the suspected latent groupings. This is accomplished through the *suspected latent grouping factor* methodology of Metzger and Franck (2021). The methodology allows for formal comparisons between ordinary linear models and latent grouping models, which protects the user from automatically selecting a spurious clustering structure that is not well supported by the data. We illustrate the ability to detect the lack of a grouping structure in the simulation studies of Metzger and Franck (2021).

The **slgf** package allows the user to (i) explore different grouping schemes for fixed effects and error variances, and (ii) specify entirely separate latent grouping factors for fixed effects and variances. We illustrate (i) in Case Study 2: Textile data, where the top model shows a different regression line for corn compared to canna and potato, but the error variance for potato is different from canna and corn (see Figure 3). To show (ii), we considered the locknut example of Subsection **Case study 3: locknut data**, where we considered whether fixture (bolt, mandrel) exhibited a fixed effect latent grouping structure, and whether interaction (bolt*CW, bolt*HT, bolt*PO, mandrel*CW, mandrel*HT, mandrel*PO) exhibited a variance latent grouping structure. As described in Subsection **Case study 3: locknut data**, we found no latent grouping structure for fixed effects, but torque error variance for bolt*HT and bolt*PO differ from the other interaction levels. The analysis supported no latent grouping structure for plating.

The **slgf** package provides functionality to detect plausible underlying cluster structures among levels of categorical predictors in linear models. This exercise in cluster detection is in some ways similar to considering a finite mixture model. R packages already exist to fit finite mixture models using the EM algorithm, such as **mixtools** (Benaglia et al., 2009). The **flexmix** package (Gruen and Leisch, 2023) in particular is notable for its ability to fit mixture models to regression data (including Gaussian, binomial, and Poisson models). Additionally, the package **MultiLCIRT** also considers latent variables for the item response theory setting; see Bartolucci et al. (2014), who use BIC for model selection rather than fractional Bayes factors.

In contrast to fitting finite mixture models for the purpose of parameter estimation and inference, **slgf** assesses the plausibility of cluster structures for small to medium-sized data sets via model selection. Additionally, **slgf** can avoid problems with convergence in the EM algorithm that may arise in small-sample scenarios, particularly when the number of data points is relatively low and the model being fit (e.g., a two component mixture model) is larger than the actual model generating the data (e.g., a one component mixture model with no cluster structure).

By contrast, **slgf** circumvents convergence issues by considering all possible groupings of points within the user-specified model classes, obtaining integrated likelihoods and posterior model probabilities for each model, and quantifying overall probability of a cluster structure as the sum of all posterior probabilities for models with two groups by the law of total probability. The **slgf** package thus excels in smaller-data settings where assessing the plausibility of a cluster structure is the core goal, and packages like **flexmix** will excel in cases where the main goal is to fit specified mixture models and conduct inference on parameters.

In addition to the basic **slgf** demonstration shown in Case Study 1: Smell data, we illustrate **slgf** functionality for mixtures of g-priors (Liang et al., 2008) and a two way unreplicated layout in Case Study 4: Bottles data. Mixtures of g-priors have been shown to work well with fractional Bayes factor methods to reduce the training fraction when sample size is small relative to the number of model parameters (Metzger and Franck, 2021).

Finally, although the methodology described here and in Metzger and Franck (2021) exclusively handles two latent groups, we call on any readers with a compelling data set that may exhibit more than two latent groups to contact the authors so that we might explore a generalization of our method to more than two groups.

We have provided an overview of functionality that we hope will enable scientists from diverse fields to access the SLGF methodology of Metzger and Franck (2021) via the **slgf** package to detect hidden groupings in the levels of categorical predictors that might impact outcomes of interest across a wide range of human endeavors.

References

F. Bartolucci, S. Bacci, and M. Gnaldi. Multilcirt: An r package for multidimensional latent class item response models. *Computational Statistics & Data Analysis*, 71:971–985, 2014. ISSN 0167-9473. doi: https://doi.org/10.1016/j.csda.2013.05.018. URL https://www.sciencedirect.com/science/article/pii/S0167947313002053. [p189]

T. Benaglia, D. Chauveau, D. R. Hunter, and D. Young. mixtools: An R package for analyzing finite mixture models. *Journal of Statistical Software*, 32(6):1–29, 2009. URL https://www.jstatsoft.org/v32/i06/. [p189]

C. T. Franck and J. A. Osborne. Exploring Interaction Effects in Two-Factor Studies using the hiddenf Package in R. *The R Journal*, 8(1):159–172, 2016. URL https://journal.r-project.org/archive/2016/RJ-2016-011/index.html. [p175]

C. T. Franck, D. M. Nielsen, and J. A. Osborne. A method for detecting hidden additivity in two-factor unreplicated experiments. *Computational Statistics & Data Analysis*, 67(Supplement C):95 – 104, 2013. ISSN 0167-9473. doi: https://doi.org/10.1016/j.csda.2013.05.002. URL http://www.sciencedirect.com/science/article/pii/S0167947313001618. [p175]

M. S. Furry. Breaking strength, elongation and folding endurance of films of starches and gelatin used in textile sizing. *Technical Bulletin (United States Department of Agriculture)*, 674:1–36, 1939. [p175, 176, 184, 185]

B. Gruen and F. Leisch. *flexmix: Flexible Mixture Modeling*, 2023. URL https://CRAN.R-project.org/package=flexmix. R package version 2.3-19. [p189]

M. Kharrati-Kopaei and S. M. Sadooghi-Alvandi. A new method for testing interaction in unreplicated two-way analysis of variance. *Communications in Statistics - Theory and Methods*, 36(15):2787–2803, 2007. doi: 10.1080/03610920701386851. URL http://dx.doi.org/10.1080/03610920701386851. [p175]

F. Liang, R. Paulo, G. Molina, M. A. Clyde, and J. O. Berger. Mixtures of g priors for bayesian variable selection. *Journal of the American Statistical Association*, 103(481):410–423, 2008. doi: 10.1198/016214507000001337. URL https://doi.org/10.1198/016214507000001337. [p178, 180, 187, 189]

G. Meek and C. Ozgur. Torque variation analysis. *Journal of the Industrial Mathematics Society*, 41:1–16, 1991. [p175, 176, 186]

T. A. Metzger and C. T. Franck. Detection of latent heteroscedasticity and group-based regression effects in linear models via bayesian model selection. *Technometrics*, 63(1):116–126, 2021. doi: 10.1080/00401706.2020.1739561. URL https://doi.org/10.1080/00401706.2020.1739561. [p175, 177, 184, 189]

R. G. O'Brien and M. W. Heft. New discrimination indexes and models for studying sensory functioning in aging. *Journal of Applied Statistics*, 22:9–27, 1995. [p175, 176, 182, 183]

A. O'Hagan. Fractional bayes factors for model comparison. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):99–138, 1995. ISSN 00359246. URL http://www.jstor.org/stable/2346088. [p179, 180]

E. R. Ott and R. D. Snee. Identifying useful differences in a multiple-head machine. *Journal of Quality Technology*, 5(2):47–57, 1973. [p176, 177, 188]

A. Zellner. On assessing prior distributions and bayesian regression analysis with g-prior distributions. In P. Goel and A. Zellner, editors, *Bayesian Inference and Decision Techniques: Essays in Honor of Bruno de Finetti*, pages 233–243. Elsevier Science Publishers, Inc., 1986. [p178]

A. Zellner and A. Siow. Posterior odds ratios for selected regression hypotheses. *Trabajos de Estadistica y de Investigacion Operativa*, 31(1):585–603, Feb 1980. ISSN 0041-0241. doi: 10.1007/BF02888369. URL https://doi.org/10.1007/BF02888369. [p178]

Thomas A. Metzger
Department of Statistics, The Ohio State University
1958 Neil Avenue, Columbus, Ohio 43210
United States of America

ORCiD: 0000-0003-3620-1405
metzger.181@osu.edu

Christopher T. Franck
Department of Statistics, Virginia Tech
403E Hutcheson Hall, Blacksburg, VA 24060
United States
(ORCiD: 0000-0003-1251-4378)
chfranck@vt.edu

BMRMM: An R Package for Bayesian Markov (Renewal) Mixed Models

by Yutong Wu and Abhra Sarkar

Abstract We introduce the **BMRMM** package implementing Bayesian inference for a class of Markov renewal mixed models which can characterize the stochastic dynamics of a collection of sequences, each comprising alternative instances of categorical states and associated continuous duration times, while being influenced by a set of exogenous factors as well as a 'random' individual. The default setting flexibly models the state transition probabilities using mixtures of Dirichlet distributions and the duration times using mixtures of gamma kernels while also allowing variable selection for both. Modeling such data using simpler Markov mixed models also remains an option, either by ignoring the duration times altogether or by replacing them with instances of an additional category obtained by discretizing them by a user-specified unit. The option is also useful when data on duration times may not be available in the first place. We demonstrate the package's utility using two data sets.

1 Introduction

Markov models (MMs) are widely used for modeling the transition dynamics of categorical state sequences. Classical Markov renewal models (MRMs) additionally model the state duration times, when available, where the state transitions follow Markov dynamics and the state duration times follow a continuous distribution that depends on the immediately preceding and following states (Figure 1). M(R)Ms have been widely used in different variations (Phelan, 1990; Eichelsbacher and Ganesh, 2002; Muliere et al., 2003; Alvarez, 2005; Diaconis and Rolles, 2006; Bulla and Muliere, 2007; Etterson et al., 2007; Bacallado et al., 2009; Li, 2009; Epifani et al., 2014; Siebert and Söding, 2016; Holsclaw et al., 2017; Sesia et al., 2019). There are also some sparse works on covariate-dependent Markov models (Muenz and Rubinstein, 1985; Gradner, 1990; Alioum et al., 1998; Islam and Chowdhury, 2006).

The existing literature however focuses very heavily on modeling single sequences. Sarkar et al. (2018) developed a highly flexible and computationally efficient class of Bayesian Markov mixed models (BMMMs) for jointly modeling a collection of categorical sequences, each one associated with an individual as well as a set of time-invariant external covariates (e.g., the sex and genotype of the associated individual, an experimental condition under which the sequence was generated, etc.). BMMMs characterize the state transition probabilities using a convex combination of a fixed covariate-dependent component and a random individual-level component, both being Dirichlet distributed. They further allow covariate levels with similar effects to be probabilistically clustered together, allowing automatic selection of the significant covariates, and providing a sophisticated framework for analyzing data sets having the aforementioned structure.

BMMMs however do not model duration times of the states which are often additionally available in real-world applications. Recently, Wu et al. (2023) extended BMMMs to Bayesian Markov renewal mixed models (BMRMMs), allowing for the additional analysis of continuous duration times which, depending on the application, can either be the duration for which a state persists or the duration between two consecutive states, i.e., inter-state intervals (ISIs). Specifically, they modeled the duration times using mixtures of gamma kernels with mixture probabilities being a convex combination a covariate-dependent effect and an individual-level effect, similar to BMMMs. Covariate levels with similar influences on mixture probabilities are clustered together in BMRMMs as well, allowing the selection of the significant covariates. BMRMMs thus holistically model both state transitions and continuous duration times, painting a comprehensive picture of the underlying stochastic dynamics.

In this article, we describe the R package **BMRMM** which implements BMMMs and BMRMMs, collectively referred to henceforth as BM(R)MMs. The **BMRMM** package runs posterior inference for categorical state transitions and continuous duration times, if available, via a Markov chain Monte Carlo (MCMC) algorithm, returning an object containing comprehensive inference results. The package also includes a suite of plotting functions to display the results graphically. Specifically for continuous duration times, when available, the package provides users with three different options: (i) ignore the duration times and model the state transitions alone as a BMMM; (ii) introduce an additional category by discretizing the continuous duration times by a user-specified unit, and analyze the appended state transitions as a BMMM; (iii) model the duration as a mixture of gamma kernels using a BMRMM, as proposed in Wu et al. (2023). Additionally, users can choose to turn off one or both of the fixed covariate effects and the random individual effects. Overall, the **BMRMM** package thus gives users a lot of flexibility in handling their data sets, providing inferences for both Bayesian Markov renewal or non-renewal models as needed.

The **BMRMM** package conveniently includes a synthetic `foxp2` data set that describes the laboratory study on the role of the FoxP2 gene implicated in speech deficiencies for adult mice, which is also the motivating application for the methodology of BMMMs and BMRMMs. Mutations in the FoxP2 gene have long been associated with severe deficits in vocal communication for mammals (MacDermot et al., 2005). Mice with and without the mutation singing under various "social contexts" have thus been studied in many experiments (Fujita et al., 2008; Castellucci et al., 2016; Gaub et al., 2016; Chabout et al., 2016). The FoxP2 data set (Chabout et al., 2016), e.g., comprises the sequences of syllables making up the songs as well as the lengths of inter-syllable intervals (ISIs). The data set `foxp2` included in the **BMRMM** package is taken from the simulation study of Wu et al. (2023). It is much shorter than the real FoxP2 data set but closely mimics its other aspects and is used in this paper to demonstrate how to obtain detailed inferences for both syllable transitions and ISI dynamics for a comprehensive analysis of the vocal repertoire in mice with and without the FoxP2 mutation.

The utility of the **BMRMM** package goes well beyond the FoxP2 data set. As described above, the package is designed for scenarios where the data set consists of categorical state sequences associated with an individual as well as a number of observed covariates where additional data on continuous duration times may or may not be available. Data sets with such structures are frequently observed in different areas of scientific research and can potentially benefit from the **BMRMM** package. For example, Islam and Chowdhury (2006) analyzed the transitions of different rainfall orders in three districts of Bangladesh under three covariates, wind speed, humidity, and maximum temperature. In an education assessment study, Zhang et al. (2019) recorded sequences of writing states, characterized by keystroke logs, for 257 eighth graders of various genders, races, and socioeconomic statuses. Combescure et al. (2003) estimated the control states of 371 asthma patients with different body mass indices (BMIs) and disease severity over a four-year-long study and produced the asthma control data set, which we will use as an additional example to demonstrate the utility of our package in this paper. For such data sets, the **BMRMM** package provides a flexible, sophisticated, and principled way to model fixed effects of the covariates and random effects of the individuals in both the state transition dynamics and the distribution of the ISIs.

Other computer programs for Markov models with covariates include MARKOV (Marshall et al., 1995) and MKVPCI (Alioum and Commenges, 2001). R packages for analyzing discrete Markov models include **markovchain** (Spedicato, 2017) and **msm** (Jackson, 2011). **SemiMarkov** (Listwon and Saint-Pierre, 2015) and **SMM** (Barbu et al., 2018) provide functions for the simulation and estimation of traditional semi-Markov models. Some R packages provide the inference of hidden semi-Markov models, such as **mhsmm** (O'Connell and Højsgaard, 2011) and **hhsmm** (Amini et al., 2022). Ferguson et al. (2012) built the **msSurv** package which provides a nonparametric estimation of semi-Markov models but does not consider covariates. There are also R packages implementing MRPs in specific application areas. For example, Kharrat et al. (2019) introduced the **Countr** package for flexible regression models based on MRPs. Pustejovsky (2021) developed the **ARPobservation** for simulating behavior streams based on alternating renewal processes. Other R packages for categorical data analysis include **catdap** (Katsura, 1980) and **vcd** (Meyer et al., 2022). To our knowledge, there has not been an R package that implements flexible Bayesian M(R)MMs.

In the following section, we summarize the technical details of BMRMMs. Documentation for the functions of the **BMRMM** package is then provided. Next, we demonstrate the usage of our package in analyzing two different data sets. The final section contains some concluding remarks.

2 The Bayesian Markov (renewal) mixed models

We briefly describe the BM(R)MM methodologies here – more details can be found in Sarkar et al. (2018) and Wu et al. (2023). Consider specifically a sequence s comprising T_s state instances and let $y_{s,t}$ denote the state at time t in sequence s. The states $y_{s,t}$'s come from a set $\mathcal{Y} = \{1, 2, \ldots, d_0\}$. Within a sequence s, there are $T_s - 1$ duration times (state persistence times or inter-state intervals), denoted by $\{\tau_{s,t}\}_{s=1,t=2}^{s_0,T_s}$, where $\tau_{s,t}$ is the duration between the $(t-1)^{th}$ and t^{th} states in sequence s, and s_0 represents the total number of sequences. Figure 1 presents a graphical summary of the data structure. Each sequence s is associated with p categorical covariates or factors, denoted by $x_{s,j} \in \mathcal{X}_j = \{1, 2, \ldots, d_j\}$, and an individual, denoted by i_s. Without loss of generality, we assume that the analyses of the transition probabilities and the duration times distributions both include all p covariates. Moreover, the analysis of duration times counts the previous state $y_{s,t-1}$ as an additional $(p+1)^{th}$ covariate. In the **BMRMM** package, users have the flexibility to select particular covariates for each analysis and exclude the previous state from the analysis of duration times. In their original definition in Pyke (1961), the duration time $\tau_{s,t}$ in an MRP was allowed to depend on both the preceding state $y_{s,t-1}$ and the following state $y_{s,t}$. To keep the notation simple and the methodology easy to understand for a broad audience, we however only include the preceding state $y_{s,t-1}$ as a predictor of $\tau_{s,t}$ in this paper. This analysis can be easily modified to have the pair $(y_{s,t-1}, y_{s,t})$ as a predictor instead of just $y_{s,t-1}$, as was actually done in Wu et al. (2023).

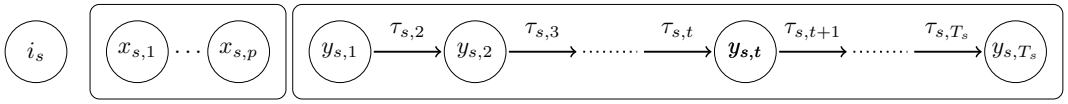


Figure 1: Graphical model showing the data structure: $y_{s,t}$ denotes the observed state at the t^{th} time location in the s^{th} sequence; $\tau_{s,t}$ denotes the observed duration times (either state persistence times or inter-state intervals) between the states $y_{s,t-1}$ and $y_{s,t}$; each sequence s is also associated with an individual i_s and a set of exogenous time-invariant covariates $x_{s,1}, \ldots, x_{s,p}$. The Markov mixed model considered in this article analyzes the state transitions $y_{s,t}$ in a collection of sequences; the Markov renewal mixed model additionally analyzes the duration times $\tau_{s,t}$; both models accommodate fixed effects of the covariates $x_{s,1}, \ldots, x_{s,p}$ and random effects of the individuals i_s.

2.1 Model for state transitions

For a sequence s associated with individual i and covariate levels $x_1, \ldots, x_p$, the transition probabilities $\Pr(y_{s,t} = y_t \mid i_s = i, x_{s,1} = x_1, \ldots, x_{s,p} = x_p, y_{s,t-1} = y_{t-1}) = P^{(i)}_{trans,x_1,\ldots,x_p}(y_t \mid y_{t-1})$ are defined as a convex combination of a fixed covariate effect component $\boldsymbol{\lambda}_{trans,x_1,\ldots,x_p}(\cdot \mid y_{t-1})$ and a random effect component $\boldsymbol{\lambda}^{(i)}_{trans}$:

$$P^{(i)}_{trans,x_1,\ldots,x_p}(y_t \mid y_{t-1}) = \pi^{(i)}_{trans,0}(y_{t-1})\lambda_{trans,x_1,\ldots,x_p}(y_t \mid y_{t-1}) + \pi^{(i)}_{trans,1}(y_{t-1})\lambda^{(i)}_{trans}(y_t \mid y_{t-1}). \quad (1)$$

The coefficients of the convex combination, namely, $\{\pi^{(i)}_{trans,0}(y_{t-1}), \pi^{(i)}_{trans,1}(y_{t-1})\}$ are individual-specific and satisfy $\pi^{(i)}_{trans,1}(y_{t-1}) = 1 - \pi^{(i)}_{trans,0}(y_{t-1})$.

For each covariate $j = 1, \ldots, p$, it is possible that some covariate levels exert a similar effect on the transition dynamics. For example, the components $\boldsymbol{\lambda}_{trans,x_1=1,x_2,\ldots,x_p}(\cdot \mid y_{t-1})$ and $\boldsymbol{\lambda}_{trans,x_1=2,x_2,\ldots,x_p}(\cdot \mid y_{t-1})$ would be equal if levels 1 and 2 of covariate 1 have similar influences on transition dynamics for fixed levels for covariates $2, \ldots, p$. A clustering mechanism for covariate levels allows the fixed component $\boldsymbol{\lambda}_{trans,x_1,\ldots,x_p}(\cdot \mid y_{t-1})$ to be the same for all levels with a similar influence. In particular, for covariate j, we construct the partition $\mathcal{C}^{(j)}_{trans} = \{\mathcal{C}^{(j)}_{trans,h_j}\}_{h_j=1}^{k_{trans,j}}$ of its levels, where $k_{trans,j}$ is the number of clusters for covariate j and h_j represents the cluster index. We introduce latent variables $\{z_{trans,j,\ell}\}_{j=1,\ell=1}^{p,d_j}$ that indicate the cluster index for the ℓ^{th} label of covariate j. Two levels of the covariate j, $\ell_1, \ell_2 \in \mathcal{X}_j = \{1, \ldots, d_j\}$, are clustered together if and only if $z_{trans,j,\ell_1} = z_{trans,j,\ell_2}$. For the fixed effects, we then replace the covariate levels $x_1, \ldots, x_p$'s with cluster indices $h_1, \ldots, h_p$'s and present the fixed effect as $\boldsymbol{\lambda}_{trans,h_1,\ldots,h_p}(\cdot \mid y_{t-1})$.

We set Dirichlet priors for both fixed and individual effect components and let them center around the same mean vector $\boldsymbol{\lambda}_{trans,0}$ to facilitate posterior computation. The probability vector $\boldsymbol{\lambda}_{trans,0}$ is also given a Dirichlet prior with mean $\boldsymbol{\lambda}_{trans,00}$ to capture the natural preferences of certain states in $\mathcal{Y}$:

$$\begin{aligned}
&\boldsymbol{\lambda}_{trans,h_1,\ldots,h_p}(\cdot \mid y_{t-1}) \sim \text{Dir}\left\{\alpha_{trans,0}\lambda_{trans,0}(1 \mid y_{t-1}), \ldots, \alpha_{trans,0}\lambda_{trans,0}(d_0 \mid y_{t-1})\right\},\\
&\boldsymbol{\lambda}^{(i)}_{trans}(\cdot \mid y_{t-1}) \sim \text{Dir}\left\{\alpha^{(0)}_{trans}\lambda_{trans,0}(1 \mid y_{t-1}), \ldots, \alpha^{(0)}_{trans}\lambda_{trans,0}(d_0 \mid y_{t-1})\right\},\\
&\boldsymbol{\lambda}_{trans,0}(\cdot \mid y_{t-1}) \sim \text{Dir}\left\{\alpha_{trans,00}\lambda_{trans,00}(1), \ldots, \alpha_{trans,00}\lambda_{trans,00}(d_0)\right\}.
\end{aligned}$$

We present the complete Bayesian hierarchical model for the transition dynamics as

$$\begin{aligned}
&(y_{s,t} \mid y_{s,t-1} = y_{t-1}, i_s = i, z_{trans,1,x_{s,1}} = h_1, \ldots, z_{trans,p,x_{s,p}} = h_p) \sim\\
&\qquad \text{Mult}\left\{P^{(i)}_{trans,h_1,\ldots,h_p}(1 \mid y_{t-1}), \ldots, P^{(i)}_{trans,h_1,\ldots,h_p}(d_0 \mid y_{t-1})\right\}, \text{ where}\\
&\mathbf{P}^{(i)}_{trans,h_1,\ldots,h_p}(\cdot \mid y_{t-1}) = \pi^{(i)}_{trans,0}(y_{t-1})\boldsymbol{\lambda}_{trans,h_1,\ldots,h_p}(\cdot \mid y_{t-1}) + \pi^{(i)}_{trans,1}(y_{t-1})\boldsymbol{\lambda}^{(i)}_{trans}(\cdot \mid y_{t-1}),\\
&z_{trans,j,\ell} \sim \text{Mult}\left\{\mu_{trans,j}(1), \ldots, \mu_{trans,j}(d_j)\right\}, \quad \boldsymbol{\mu}_{trans,j} \sim \text{Dir}(\alpha_{trans,j}, \ldots, \alpha_{trans,j}),\\
&\boldsymbol{\lambda}_{trans,h_1,\ldots,h_p}(\cdot \mid y_{t-1}) \sim \text{Dir}\left\{\alpha_{trans,0}\lambda_{trans,0}(1 \mid y_{t-1}), \ldots, \alpha_{trans,0}\lambda_{trans,0}(d_0 \mid y_{t-1})\right\},\\
&\boldsymbol{\lambda}^{(i)}_{trans}(\cdot \mid y_{t-1}) \sim \text{Dir}\left\{\alpha^{(0)}_{trans}\lambda_{trans,0}(1 \mid y_{t-1}), \ldots, \alpha^{(0)}_{trans}\lambda_{trans,0}(d_0 \mid y_{t-1})\right\},\\
&\boldsymbol{\lambda}_{trans,0}(\cdot \mid y_{t-1}) \sim \text{Dir}\left\{\alpha_{trans,00}\lambda_{trans,00}(1), \ldots, \alpha_{trans,00}\lambda_{trans,00}(d_0)\right\},\\
&\pi^{(i)}_{trans,0}(y_{t-1}) \sim \text{Beta}(a_{trans,0}, a_{trans,1}),\\
&\alpha_{trans,0} \sim \text{Ga}(a_{trans,0}, b_{trans,0}), \qquad \alpha^{(0)}_{trans} \sim \text{Ga}(a^{(0)}_{trans}, b^{(0)}_{trans}).
\end{aligned}$$

2.2 Model for continuous duration times

The **BMRMM** package provides three options for analyzing the duration times: (i) ignore the durations altogether and only model the transition probabilities of the existing states, (ii) treat the durations as blocks of a new special category, with a discretization unit specified by users, (iii) model the durations as a continuous random variable with a flexible mixture of gamma distributions. For the first two options, we only need to apply the model described in the previous subsection. For the third option, we need to conduct a separate analysis of the duration times as described below.

Let K denote the number of gamma mixture components in the model for the duration times. We let $\mathbf{P}^{(i)}_{dur}(\cdot \mid x_1, \ldots, x_p, y_{s,t-1})$ denote the mixture probability vector given individual i, covariate levels $x_1, \ldots, x_p$, and preceding syllable $y_{s,t-1}$. The preceding state $y_{s,t-1}$ can be removed from the formula if the users do not wish to consider its influence in the inference of the duration times. The distribution of the continuous duration times, $\{\tau_{s,t}\}^{s_0,T_s}_{s=1,t=2}$, is then modeled as

$$\begin{aligned} &f(\tau_{s,t} \mid i_s = i, x_{s,1} = x_1, \ldots, x_{s,p} = x_p, y_{s,t-1} = y_{t-1}) \\ &= \textstyle\sum_{k=1}^{K} P^{(i)}_{dur}(k \mid x_1, \ldots, x_p, y_{t-1}) \mathrm{Ga}(\tau_{s,t} \mid \alpha_k, \beta_k), \end{aligned}$$

where α_k and β_k denote the shape and rate parameters of the k^{th} gamma mixture component, respectively. We introduce a set of latent variables $\{z_{dur,s,t}\}^{s_0,T_s}_{s=1,t=2}$ that represents the index of the mixture component. If $z_{dur,s,t}$ equals to k, then $\tau_{s,t}$ follows $\mathrm{Ga}(\alpha_k, \beta_k)$ distribution, i.e.,

$$\begin{aligned} f(\tau_{s,t} \mid z_{dur,s,t} = k) &\sim \mathrm{Ga}(\tau_{s,t} \mid \alpha_k, \beta_k), \\ \Pr(z_{dur,s,t} = k \mid i_s = i, x_{s,1} = x_1, \ldots, x_{s,p} = x_p, y_{s,t-1} = y_{t-1}) &= P^{(i)}_{dur}(k \mid x_1, \ldots, x_p, y_{t-1}). \end{aligned}$$

Similar to the model for the transition probabilities, the mixture probabilities are a convex combination of a fixed population-level effect and a random individual-level effect:

$$\mathbf{P}^{(i)}_{dur}(\cdot \mid x_1, \ldots, x_p, y_{t-1}) = \pi^{(i)}_{dur,0}(\cdot) \boldsymbol{\lambda}_{dur,x_1,\ldots,x_p,y_{t-1}}(\cdot) + \pi^{(i)}_{dur,1}(\cdot) \boldsymbol{\lambda}^{(i)}_{dur}(\cdot),$$

where $\boldsymbol{\lambda}_{dur,x_1,\ldots,x_p,y_{t-1}}(\cdot)$ is the baseline component, $\boldsymbol{\lambda}^{(i)}_{dur}(\cdot)$ is the random individual effect, and $\{\pi^{(i)}_{dur,0}(k), \pi^{(i)}_{dur,1}(k)\}^{K}_{k=1}$ are individual-specific coefficients such that $\pi^{(i)}_{dur,1}(k) = 1 - \pi^{(i)}_{dur,0}(k)$. Again, for each covariate $r = 1, \ldots, p, p+1$ (where the $(p+1)^{th}$ covariate is the preceding state y_{t-1}), we construct the partition $\mathcal{C}^{(r)}_{dur} = \{\mathcal{C}^{(r)}_{dur,g_r}\}^{k_{dur,r}}_{g_r=1}$ of its levels, where $k_{dur,r}$ is the number of clusters for covariate r and g_r represents the cluster index. We introduce latent variables $\{z_{dur,r,w}\}^{p+1,d_r}_{r=1,w=1}$ that indicate the cluster index for the w^{th} label of the r^{th} covariate. We now replace the population-level effect $\boldsymbol{\lambda}_{dur,x_1,\ldots,x_p,y_{t-1}}(\cdot)$ with $\boldsymbol{\lambda}_{dur,g_1,\ldots,g_p,g_{p+1}}(\cdot)$.

The mixture probability vectors are given Dirichlet priors with the mean vector $\boldsymbol{\lambda}_{dur,0}$, which itself centers around a global vector $\boldsymbol{\lambda}_{dur,00}$:

$$\begin{aligned} &\boldsymbol{\lambda}_{dur,g_1,\ldots,g_{p+1}}(\cdot) \sim \mathrm{Dir}\left\{\alpha_{dur,0}\lambda_{dur,0}(1), \ldots, \alpha_{dur,0}\lambda_{dur,0}(K)\right\}, \\ &\boldsymbol{\lambda}^{(i)}_{dur}(\cdot) \sim \mathrm{Dir}\left\{\alpha^{(0)}_{dur}\lambda_{dur,0}(1), \ldots, \alpha^{(0)}_{dur}\lambda_{dur,0}(K)\right\}, \\ &\boldsymbol{\lambda}_{dur,0}(\cdot) \sim \mathrm{Dir}\left\{\alpha_{dur,00}\lambda_{dur,00}(1), \ldots, \alpha_{dur,00}\lambda_{dur,00}(K)\right\}. \end{aligned}$$

We present the complete Bayesian hierarchical model for the continuous duration times as

$$\begin{aligned} &(\tau_{s,t} \mid z_{dur,s,t} = k) \sim \mathrm{Ga}(\tau_{s,t} \mid \alpha_k, \beta_k), \\ &(z_{dur,s,t} \mid i_s = i, z_{dur,1,x_{s,1}} = g_1, \ldots, z_{dur,p,x_{s,p}} = g_p, z_{dur,p+1,y_{s,t-1}} = g_{p+1}) \sim \\ &\qquad \mathrm{Mult}\left\{P^{(i)}_{dur,g_1,\ldots,g_{p+1}}(1), \ldots, P^{(i)}_{dur,g_1,\ldots,g_{p+1}}(K)\right\}, \text{ where} \\ &P^{(i)}_{dur,g_1,\ldots,g_{p+1}}(k) = \pi^{(i)}_{dur,0}(k)\lambda_{dur,g_1,\ldots,g_{p+1}}(k) + \pi^{(i)}_{dur,1}(k)\lambda^{(i)}_{dur}(k), \\ &\boldsymbol{\lambda}_{dur,g_1,\ldots,g_{p+1}}(\cdot) \sim \mathrm{Dir}\left\{\alpha_{dur,0}\lambda_{dur,0}(1), \ldots, \alpha_{dur,0}\lambda_{dur,0}(K)\right\}, \quad \alpha_{dur,0} \sim \mathrm{Ga}(a_{dur,0}, b_{dur,0}), \\ &\boldsymbol{\lambda}^{(i)}_{dur}(\cdot) \sim \mathrm{Dir}\left\{\alpha^{(0)}_{dur}\lambda_{dur,0}(1), \ldots, \alpha^{(0)}_{dur}\lambda_{dur,0}(K)\right\}, \quad \alpha^{(0)}_{dur} \sim \mathrm{Ga}(a^{(0)}_{dur}, b^{(0)}_{dur}), \\ &\boldsymbol{\lambda}_{dur,0}(\cdot) \sim \mathrm{Dir}\left\{\alpha_{dur,00}\lambda_{dur,00}(1), \ldots, \alpha_{dur,00}\lambda_{dur,00}(K)\right\}, \\ &\pi^{(i)}_{dur,0}(k) \sim \mathrm{Beta}(a_{dur,0}, a_{dur,1}), \\ &\alpha_k \sim \mathrm{Ga}(a_{dur,0}, b_{dur,0}), \quad \beta_k \sim \mathrm{Ga}(a_{dur,0}, b_{dur,0}). \end{aligned}$$

Inference is based on samples drawn from the posterior using a Metropolis-Hastings-within-Gibbs MCMC algorithm. Most full conditionals are available in closed form and can be directly sampled from. A Metropolis-Hastings step is however used for updating the discrete valued cluster configurations. There is, however, no conjugate prior for gamma distributions with unknown shape parameters (Damsleth, 1975). Recently, Miller (2019) designed a procedure that efficiently approximates the posterior full conditionals of gamma shape parameters under a gamma prior with another gamma density. We adopt this approximation in our MCMC algorithm.

3 The BMRMM R package

3.1 Package description

The **BMRMM** package is developed to implement Bayesian Markov (renewal) mixed models. The main function `BMRMM` of the package carries out detailed analyses of the state transitions and their duration times (if applicable) as described in the previous section. Moreover, the package includes a number of supplementary functions that use the results of the main function to produce numerical summaries, visualizations, and diagnostics. Table 1 provides a brief description of all functions.

Function	Description
`BMRMM`	Creates a `BMRMM` object.
`summary.BMRMM`	Summary for an object of class `BMRMM` and create a `BMRMMsummary` object.
`plot.BMRMMsummary`	Visualization of a `BMRMMsummary` object.
`hist.BMRMM`	Returns histograms of duration times for a `BMRMM` object.
`diag.BMRMM`	Provides MCMC diagnostic plots for a `BMRMM` object.
`model.selection.scores`	Returns the LPML and WAIC scores of the mixture gamma model.

Table 1: Summary of functions in the **BMRMM** package.

3.2 The main function BMRMM

The main function is `BMRMM` which implements the inference for both the state transition probabilities and the duration times. We summarize the parameters in Table 3 and present the function as follows.

```
BMRMM(data, num.cov, cov.labels = NULL, state.labels = NULL,
      random.effect = TRUE, fixed.effect = TRUE,
      trans.cov.index = 1:num.cov, duration.cov.index = 1:num.cov,
      duration.distr = NULL, duration.incl.prev.state = TRUE,
      simsize = 10000, burnin = simsize/2)
```

The parameter `data` specifies the target data set and needs to follow a certain structure. The first column should list the individual IDs i_s, followed by p columns for the values of the p associated covariates $x_{s,j}$, then two columns for the values of the previous state $y_{s,t-1}$, the current state $y_{s,t}$, and finally a column for duration times $\tau_{s,t}$. The package supports one to five categorical covariates that take on values $1, 2, \ldots$. The duration times column is optional if the user would like to use BMMM instead of BMRMM to analyze just the state transitions. This is shown in Table 2. The users can look at the included simulated data set `foxp2` as an example.

Id	Covariate 1	...	Covariate p	Previous State	Current State	State Durations/ISI

Table 2: Columns of the desired input data set.

The number of covariates in the data set is specified by the argument `num.cov`. The argument `cov.labels` is a list of vectors giving the names of covariate levels in the covariate order that is presented in `data` while the parameter `state.labels` is a vector providing the names of the transition states. The default labels are Arabic numerals.

The `random.effect` parameter gives users the option to exclude the random individual effects. If `random.effect` is set to `FALSE`, the transition probabilities (and the mixture probabilities for duration times, if applicable) will only consider the influence of the covariate levels. Similarly, the `fixed.effect` parameter allows users to exclude the fixed population effects. The default values for `random.effect` and `fixed.effect` are both `TRUE`. The covariate indices for the two analyses can be specified by setting

trans.cov.index and duration.cov.index. We note that indices specified by trans.cov.index and duration.cov.index refer to the index of the covariate when the first covariate is given index 1, thus different from its index in data.

Users can define duration.distr in the following three ways.

1. If users set duration.distr to be NULL, which is the default setting, then the duration times will be ignored and not modeled at all. The BMMM described will be implemented to analyze the existing state transitions alone.
2. If duration.distr is set as list(`mixDirichlet',unit), the duration times will be used to construct a new state `dur.state', which will be analyzed along with the original set of states. The additional argument unit must be defined and acts both as a threshold and as a block size for duration times. For example, if the unit is set to 5, then for each duration value greater than 5 units, each block of 5 unit in it will be treated as an instance of a new 'dur.state' state. If there is a state transition from state `a' to `b' with a duration time of 15 seconds and the unit is specified at 5 seconds, then the updated Markov sequence will contain three consecutive `dur.state' states, i.e., (`a',`dur.state',`dur.state',`dur.state',`b'). Since we adopt the floor operation, a duration time of say 17.68 seconds will also be replaced by three consecutive instances of 'dur.state' states in this example. The BMMM model will then be implemented to analyze the resulting appended state transitions.

   These first two options may naturally result in loss of information and is therefore not recommended when a detailed analysis of the distribution of the duration times is warranted.
3. If duration.distr is set to be list(`mixgamma',shape,rate), the duration times are modeled as a continuous random variable using a flexible mixture of gamma kernels, as described for a BMRMM model. In this case, users can specify the prior shape and rate parameters with the shape and rate arguments within the definition of duration.distr. We note that shape and rate must be numeric vectors of the same length.

By default, we consider the previous state $y_{s,t-1}$ as a covariate when we model the duration times as continuous variables, i.e., duration.incl.prev.state is set to TRUE. Users can set this parameter to FALSE if they wish to exclude the previous state when analyzing the duration times. The remaining parameters simsize and burnin denote the total number of MCMC iterations and the number of burn-ins, respectively.

Argument	Explanation	Default value
data	the data set to be used following the required format	
num.cov	an integer giving the number of observed covariates in data	
cov.labels	a list of vectors giving names of all covariate levels	NULL
state.labels	a vector giving names of the states	NULL
random.effect	TRUE if random individual effects are included	TRUE
fixed.effect	TRUE if fixed population effects are included	TRUE
trans.cov.index	selects the covariates to analyze for transition probabilities	1:num.cov
duration.cov.index	selects the covariates to analyze for duration times	1:num.cov
duration.distr	specifies the distribution for duration times	NULL
duration.incl.prev.state	TRUE if y_{t-1} acts as a covariate for the analysis of duration times	TRUE
simsize	number of MCMC iterations	10000
burnin	number of burnins of the MCMC algorithm	simsize/2

Table 3: Arguments to the BMRMM function.

The BMRMM function returns an object of class BMRMM, which either contains only results.trans or both of results.trans and results.duration if duration times follow a mixture gamma distribution. For the state transitions, the posterior mean transition probability matrices for each combination of the covariate levels and each individual are given by results.trans$tp.exgns.post.mean and results.trans$tp.anmls.post.mean, respectively. Additionally, results.trans$clusters stores cluster configurations for each covariate from each MCMC iteration. As for duration times, the fields results.duration$shape.samples and results.duration$rate.samples record the shape and rate parameters, for each mixture component in every MCMC iteration, respectively. Meanwhile, results.duration$comp.assignment gives the assignment of the mixture component for each data point in the last MCMC iteration. Similar to transition probabilities, results.duration$clusters gives the cluster configurations of the covariates. Other elements of results.trans and results.duration can be found in the detailed R function description.

3.3 Summarizing BMRMM results

The **BMRMM** package provides an S3 method for summarizing results of a BMRMM object as follows.

```
summary.BMRMM(object, delta = 0.02, digits = 2, ...)
```

The object must be of class BMRMM. The argument delta is associated with local tests for transition probabilities, which we will explain further. The digit parameter is an integer used for number formatting, as in the general summary function. The summary.BMRMM function returns an object of class BMRMMsummary with the following fields.

- trans.global and dur.global
  These two fields give the global test results from the inference of transition probabilities and duration times. Global tests show the significance of the covariates in affecting the state transitions and duration times. Specifically, for each covariate, the empirical distribution of the size of the clusters in the stored MCMC iterations is calculated. The null hypothesis that a covariate is not important is equivalent to the event that all its levels are in the same cluster, or, in other words, that the cluster size for the covariate is just one.
- trans.probs.mean and trans.probs.sd
  The two fields provide the mean and standard deviation for the posterior mean of each transition type under all combinations of covariate levels, respectively.
- trans.local.mean.diff and trans.local.null.test
  The BMRMMsummary object also contains local test results for transition probabilities. Local tests analyze the differences between the transition probabilities associated with two different levels of a covariate j, fixing the levels of the other covariates. For every pair of levels of covariate j, trans.local.mean.diff gives the absolute differences in transition probabilities for each transition type in the MCMC iterations. The local null hypothesis we test for each transition type is that this difference is at least the pre-specified value delta. Meanwhile, trans.local.null.test gives the probability of the null hypothesis that the difference between two covariate levels is not significant under each transition type.
- dur.mix.params and dur.mix.probs
  For each mixture component, dur.mix.params provides the estimates of the gamma shape and rate parameters from the last MCMC iteration. For every covariate level, users can obtain the mixture probabilities by calling the field dur.mix.probs, which can be further used to estimate the length of the duration times.

3.4 Visualizing results with BMRMM plotting functions

The main plotting function of the package, plot.BMRMMsummary, is an S3 method for class BMRMMsummary. It gives the barplots for global tests as well as heatmaps for the posterior mean and standard deviation for transition probabilities, local tests for transition probabilities, mixture parameters and probabilities for duration times. The parameters of plot.BMRMMsummary include x, which must be an object of class BMRMMsummary and type, which is a single string representing the field of x that needs to be plotted. The function also takes general plotting arguments such as xlab, ylab, etc.

```
plot.BMRMMsummary(x, type, xlab = NULL, ylab = NULL, main = NULL, col = NULL, ...)
```

When duration times are analyzed as continuous variables using mixture gamma distributions, the users can use the S3 method hist.BMRMM to generate histograms for duration times along with the estimated posterior distribution. The parameter x is an object of class BMRMM. The argument comp gives the specific mixture component that the user would like to investigate. When comp is NULL, which is the default setting, the histogram of all observed duration times is plotted and superimposed with the posterior mean of the fitted mixture gamma distribution. When comp is a specific integer, we will be looking at the last MCMC iteration. The histogram for duration times assigned with component comp will be presented alongside the mixture gamma distribution with the shape and rate parameters from the last MCMC iteration. Users can refer to the documentation of the general hist function to see the interpretation for the rest of the parameters.

```
hist.BMRMM(x, comp = NULL, xlim = NULL, breaks = NULL, main = NULL,
           col = 'gray', xlab = 'Duration times', ylab = 'Density', ...)
```

Finally, users can check the MCMC diagnostics with the traceplots and autocorrelation plots produced by the function diag.BMRMM. The object parameter should be an object of class BMRMM. For transition probabilities, users can specify the covariate levels as well as the state transitions they are interested in by defining cov.combs and transitions, respectively. For duration times, users can define components, a numeric vector, to obtain the diagnostic plots for shape and rate parameters of the specific component kernels.

```
diag.BMRMM(object, cov.combs = NULL, transitions = NULL, components = NULL)
```

3.5 Model selection scores for continuous duration times

When the duration times are modeled using mixtures of gamma distributions, model selection can be performed on the number of mixture components using the function `model.selection.scores`.

```
model.selection.scores(object)
```

The function takes an `object` as its input, which must be an object of class `BMRMM`. It returns a list consisting of the log pseudo marginal likelihood (LPML) (Geisser and Eddy, 1979) and the widely applicable information criterion (WAIC) (Watanabe and Opper, 2010) scores of the model. Larger values of LPML and smaller values of WAIC indicate better model fits. They are particularly suitable for complex Bayesian hierarchical models as they can be easily computed from the MCMC samples.

4 Illustrations on the synthetic FoxP2 data set

The FoxP2 data set records the songs sung by adult male mice of two genotypes, wild type or FoxP2, denoted by W and F, respectively (Chabout et al., 2016). The mice sang under three social contexts, U (fresh female urine on a cotton tip placed inside the male's cage), L (an awake and behaving adult female placed inside the cage), and A (an anesthetized female placed on the lid of the cage). Each song comprises a sequence of syllables and continuous inter-syllable intervals (ISIs). The data set can be used to analyze the effect of the FoxP2 gene on the vocal syntax of mice, in turn providing insights into the effects of the gene on human vocal communication abilities and related deficiencies. The real FoxP2 data set originates from the study by Chabout et al. (2016) and requires permission to use. Wu et al. (2023) simulated a data set that closely mimics the real one. For demonstrating the **BMRMM** package, we included in it a shortened version of this synthetic data set which we refer to as the `foxp2` data set.

The `foxp2` synthetic data set has 17391 rows and 6 columns, which are Id, Genotype, Context, Prev_State, Cur_State, and Transformed_ISI. The original FoxP2 data set records ISIs in seconds. In the simulated data set `foxp2`, following Wu et al. (2023), log(1+ISI) values are used which give a better model fit.

Id	Genotype	Context	Prev_State	Cur_State	Transformed_ISI
1	2	2	3	3	0.20197711
1	2	2	3	3	0.06972753
1	2	2	3	3	0.07211320
1	2	2	3	3	0.15790932
1	2	2	3	3	0.06781471
1	2	2	3	3	0.09426236

Table 4: Part of the simulated FoxP2 data set `foxp2`.

If we are only interested in analyzing the transition probabilities with the covariates genotype and social contexts, we would use the main function as follows.

```
R> res.fp2 <- BMRMM(foxp2, num.cov = 2)
```

If we would like to pick specific covariates for our analyses, we can define `trans.cov.index` and `duration.cov.index` accordingly. For example, if we only want to use context for transition probabilities and genotype for ISIs, we would run the following.

```
R> res.fp2 <- BMRMM(foxp2, num.cov = 2,
                    trans.cov.index = c(2), duration.cov.index = c(1))
```

If we would like to analyze the ISIs as part of the original state sequence following a mixture Dirichlet distribution, as was done by Sarkar et al. (2018), the ISIs are replaced by (possibly consecutive) "silent" states by dividing them into blocks of 250 milliseconds. The `BMRMM` function can do this by setting `duration.distr` as a list with the string `'mixDirichlet'` and the argument `unit` as $\log(0.25+1)$, based on the log transformation.

```
R> res.fp2 <- BMRMM(foxp2, num.cov = 2,
                    duration.distr = list('mixDirichlet', unit = log(0.25+1)))
```

In the next example, we would like to analyze the ISIs as continuous variables following a mixture gamma distribution. For syllable transitions, we use both genotype and context as covariates. For the ISIs, in addition to these two, we also use the preceding syllable as a covariate.

```
R> res.fp2 <- BMRMM(data = foxp2, num.cov = 2, state.labels = c('d', 'm', 's', 'u'),
                    cov.labels = list(c('F', 'W'), c('U', 'L', 'A')),
                    duration.distr = list('mixgamma', shape = rep(1, 4), rate = rep(1, 4)))
```

In what follows, we show the results for the last function call. The returned `res.fp2` have two parts, which are named `res.fp2$results.trans` and `res.fp2$results.duration`. Now we demonstrate how we print and visualize the results.

First, we obtain a `BMRMMsummary` object, `sm.fp2`, by calling the `summary.BMRMM` function on the returned results `res.fp2`. The global test results for identifying the significant covariates can be found by calling the fields `trans.global` and `dur.global`. The function `plot.BMRMMsummary` is called to visualize the global tests using barplots, as presented in Figure 2. We recall that a covariate is significant when its levels formed more than one cluster with very high posterior probability (the bar heights). Figure 2 and the printed results suggest that every covariate is significant for the ISIs but only the social context is significant for the transition probabilities.

```
R> sm.fp2 <- summary(res.fp2)
R> sm.fp2$trans.global
            label_data
cluster_data Context Genotype
           1       0        1
           3       1        0
R> sm.fp2$dur.global
            label_data
cluster_data Context Genotype prev_state
           2    0.00     1.00       0.10
           3    1.00     0.00       0.90
           4    0.00     0.00       0.01
R> plot(sm.fp2, 'trans.global')
R> plot(sm.fp2, 'dur.global')
```

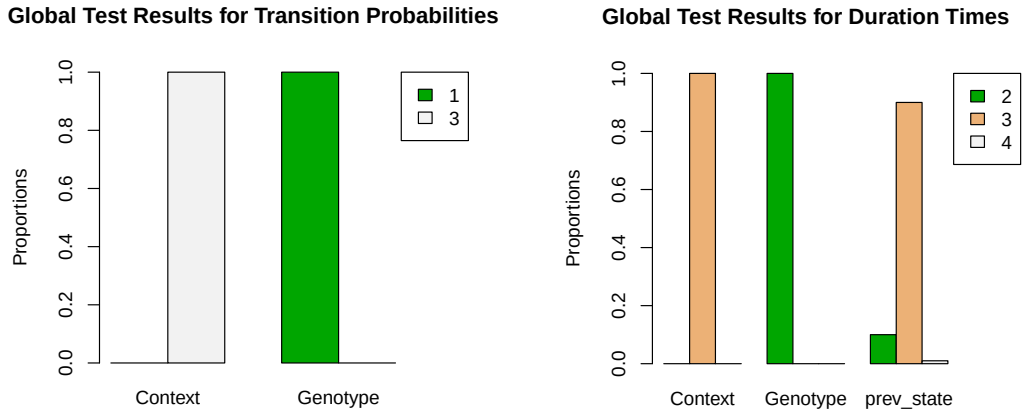


Figure 2: Results for the simulated `foxp2` data set showing the global tests of significance of the covariates for the state transitions (left) and the ISIs (right). The bars represent the estimated posterior probabilities of the number of clusters formed by the levels of each covariate.

The plotting function can be called to visualize the posterior transition probabilities under different combinations of the covariate levels. We show in Figure 3 the heatmaps for the posterior mean and standard deviation of the transition probabilities for each transition type for the following combinations of covariates: (F, A) and (W, L).

```
R> plot(sm.fp2, 'trans.probs.mean')
R> plot(sm.fp2, 'trans.probs.sd')
```

We also perform the local test to assess the influence of genotype on the transition probabilities by computing the absolute difference of the transition probabilities between F and W among the thinned MCMC samples after burn-ins, i.e., $|\Delta\lambda_{trans,\cdot,x_2}(y_t \mid y_{t-1})| = |\lambda_{trans,1,x_2}(y_t \mid y_{t-1}) - \lambda_{trans,2,x_2}(y_t \mid$

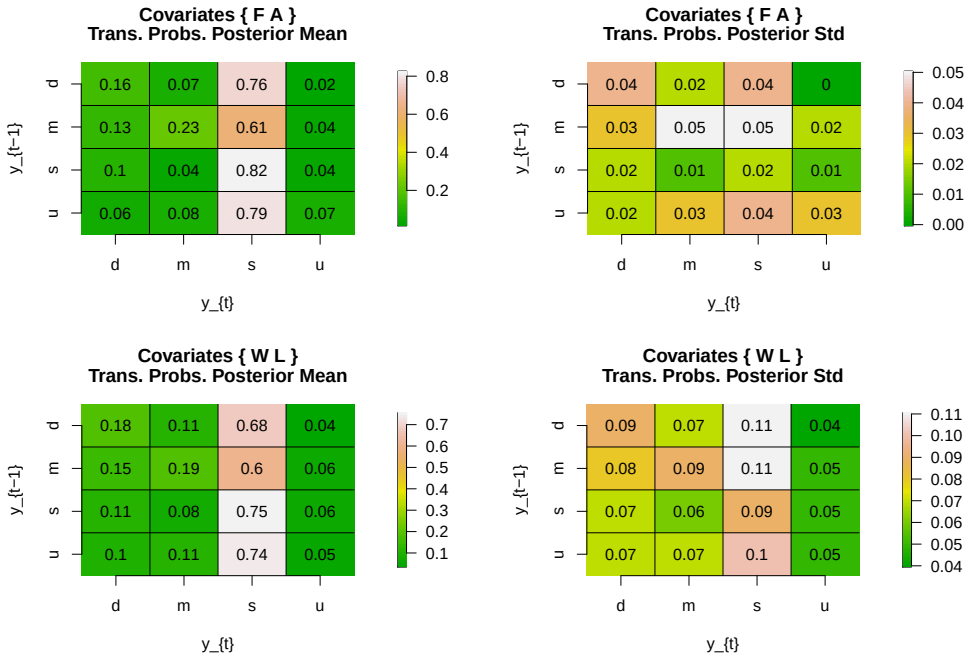


Figure 3: Results for the simulated `foxp2` data set showing the posterior mean and standard deviation for each transition type for selected covariate combinations, (F, A) (top) and (W, L) (bottom).

$y_{t-1})|$. The estimated posterior probability for the null hypothesis is therefore the proportion of times $|\Delta\lambda_{trans,\cdot,x_2}(y_t \mid y_{t-1}) \leq \delta|$ is observed in the MCMC samples, where x_2 is the social context and δ is the user-specific difference threshold `delta`. The plotting function `plot.BMRMMsummary` gives the plots for all local test results if we set the `type` to be `` `trans.local.mean.diff' `` or `` `trans.local.null.test' ``. Here, we show the results of local tests for the covariate 1 (i.e., genotype) with `delta` equaling the default value of 0.02, and present the plots in Figure 4. From the figure, we see that the posterior probabilities of the null hypotheses are generally large for most transition types (e.g., transitions to the syllable u) regardless of the social context, indicating that genotype does not have a strong influence on transition probabilities with a fixed context under these transition types.

```
R> plot(sm.fp2, 'trans.local.mean.diff')
R> plot(sm.fp2, 'trans.local.null.test')
```

Next, we turn our attention to the ISIs. We first check the fit of our estimated mixture gamma distribution presented in Figure 5a. We then look further into the shape of each mixture component in Figure 5b. From the histogram for each component, we see that components 2 and 4 represent longer ISIs while components 1 and 3 represent shorter ISIs.

```
R> hist(res.fp2, xlim = c(0,1))
R> for(comp in 1:4) {
     hist(res.fp2, comp = comp)
   }
```

We examine the values of mixture parameters and mixture probabilities for each covariate level in the last MCMC iteration, which provides insights into the influence of the covariate on ISI lengths.

```
R> sm.fp2$dur.mix.params
       shape.k rate.k
Comp 1   29.07 394.30
Comp 2    1.23   1.49
Comp 3    8.46 465.66
Comp 4    3.03  20.13

R> sm.fp2$dur.mix.probs
$Genotype
          F    W
Comp 1 0.46 0.48
Comp 2 0.19 0.13
```

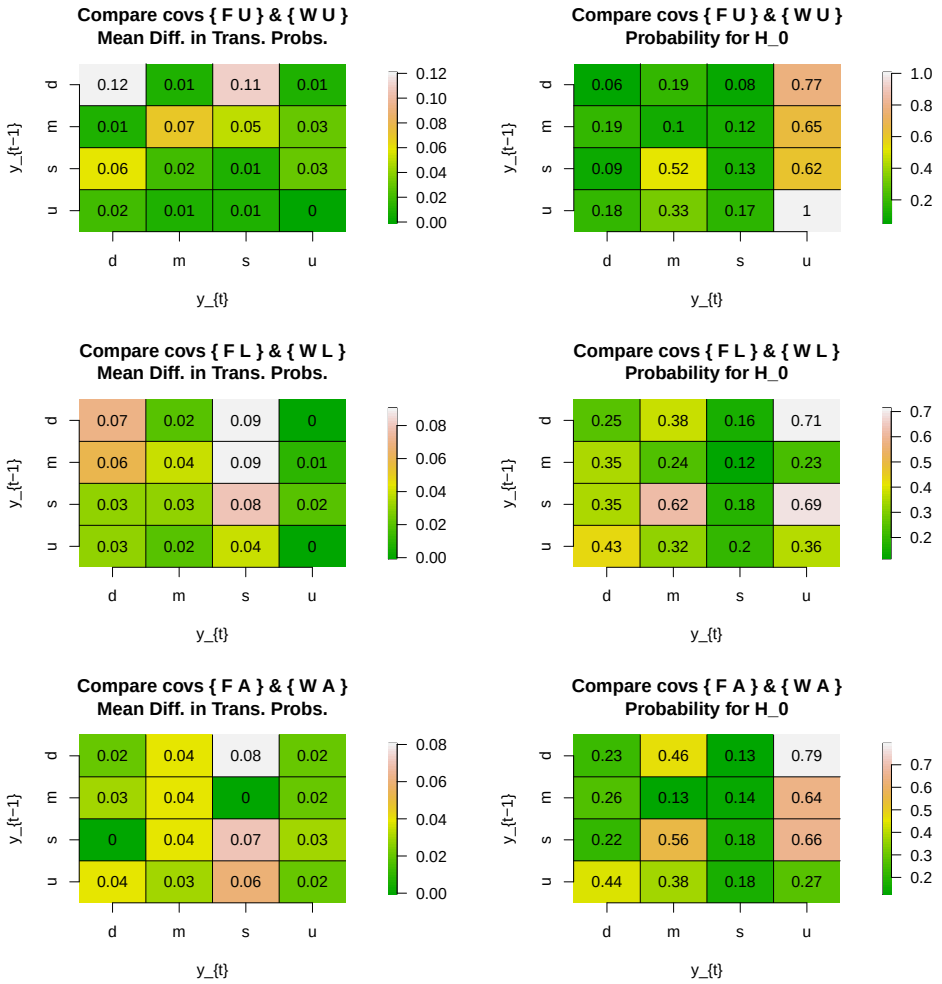
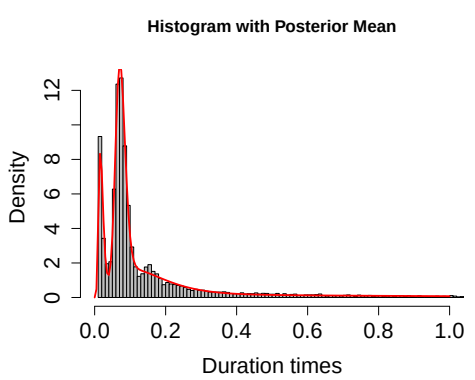
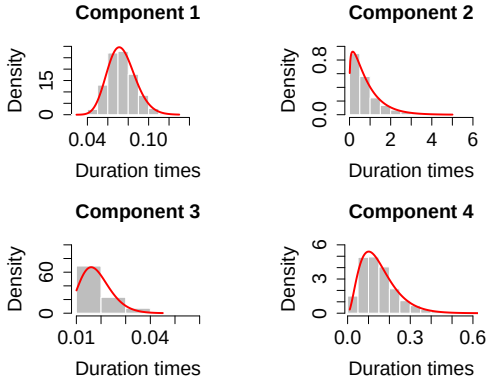


Figure 4: Results for the simulated `foxp2` data set showing local test results for genotypes fixing the social context, U (top), L (middle), and A (bottom). The averaged absolute difference in transition probabilities between F and W is presented on the left. The posterior probabilities of the corresponding null hypotheses are on the right.



(a) Histogram of ISIs with the estimated posterior mean (red line) of their marginal gamma mixture density averaged from recorded MCMC samples.



(b) Histograms of ISIs for each component of the gamma mixture model along with the component density (red lines) from the last MCMC iteration.

Figure 5: Results for the simulated `foxp2` data set showing the histograms of the ISIs with the estimated posterior gamma mixture density (left) and the histograms of the ISIs for each mixture component (right).

```
Comp 3 0.10 0.15
Comp 4 0.25 0.24

$Context
          U    L    A
Comp 1 0.58 0.35 0.48
Comp 2 0.14 0.16 0.18
Comp 3 0.08 0.21 0.08
Comp 4 0.20 0.28 0.25

$prev_state
          d    m    s    u
Comp 1 0.52 0.52 0.42 0.42
Comp 2 0.13 0.13 0.22 0.17
Comp 3 0.11 0.11 0.10 0.18
Comp 4 0.24 0.24 0.26 0.23
```

From the mixture probabilities, we see that mice with genotype F have a much higher mixture probability in component 2 compared to genotype W, which indicates mice with the FoxP2 mutation require a longer ISI between pronouncing two syllables, a reflection of vocal impairment.

Finally, we check the MCMC diagnostic plots and see if we had good mixing for the parameters. Here we focus on a specific transition type, $u \to m$, for covariate combination $\{F, U\}$ and a specific mixture component (component 2). We show these plots in Figure 6.

```
R> diag.BMRMM(res.fp2, cov.combs = list(c(1, 1)),
              transitions = list(c(4, 2)), components = c(2))
```

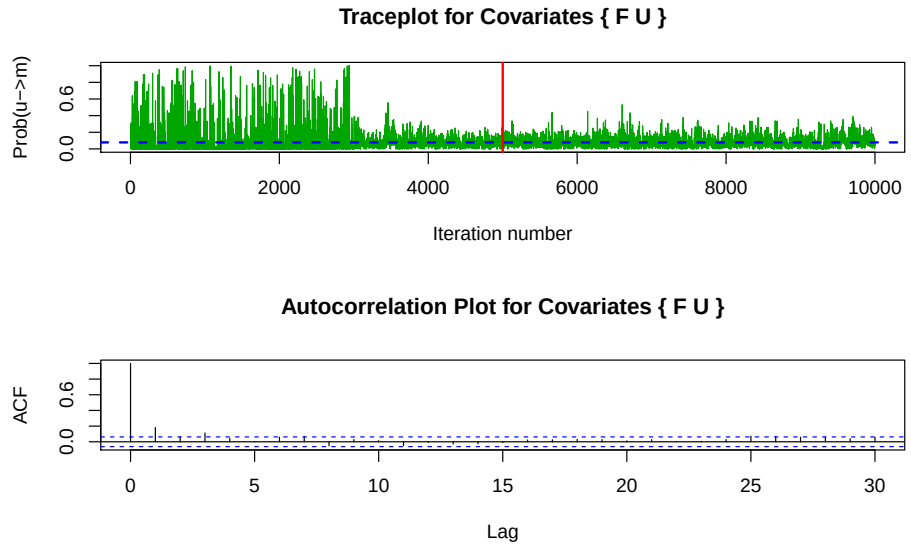
5 Illustrations on the asthma control data set

The **BMRMM** package is able to analyze duration times in detail which could either be the ISIs, as seen in the synthetic `foxp2` data set, or the state persistence times, as in a traditional semi-Markov model. To demonstrate the usage of our package in analyzing the state persistence times, we use the asthma control data set from the ARIA (Association pour la Recherche en Intelligence Artificielle) study of severe asthmatic patients (Combescure et al., 2003) in France between 1997 and 2001. At each visit, a chest physician graded the asthma control status of the patient using control scores (Juniper et al., 1999). The data set contains the sojourn time of the control states as well as three covariates: Asthma severity, sex, and the body mass index (BMI) of the patients. Saint-Pierre et al. (2003) used a Markov model with piece-wise constant intensities to model the asthma control evolution and proposed a regression model for analyzing the effect of covariates. Combescure et al. (2003) used the data set to assess the relationship between asthma severity and control of asthma. Listwon and Saint-Pierre (2015) fitted a semi-Markov model for the sojourn times using exponential and Weibull distributions and analyzed the effect of covariates individually due to complexity. Our **BMRMM** package is able to analyze the effect of the three covariates while also incorporating random individual effects exhibited by different patients on transition dynamics and state duration times.

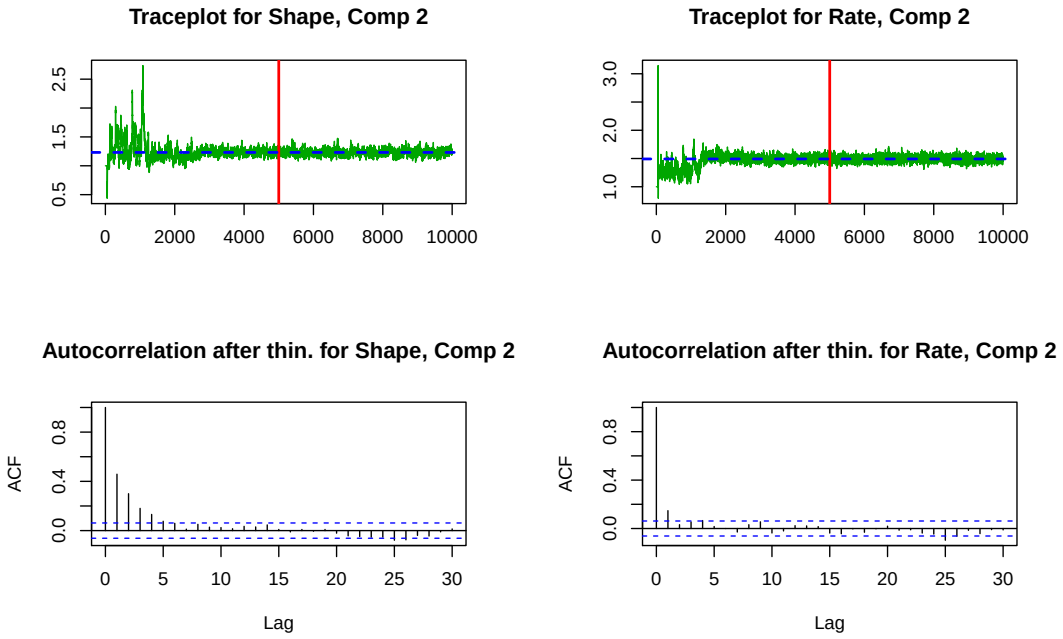
The `asthma` data set we use here is from the **SemiMarkov** package (Listwon and Saint-Pierre, 2015). We have renamed and reordered the columns such that the data set fits the required format. Specifically, the data set has 928 rows, recording the asthma control states of 371 patients, which is one of the following three transient states: Optimal control (State 1), sub-optimal control (State 2), and unacceptable control (State 3). Each state can transit to any other two states and the state duration times are recorded. The data set also contains three binary covariates of the asthma patients, including the disease severity (1 if mild-moderate and 2 if severe), BMI (body mass index, 1 if BMI < 25 and 2 otherwise), and sex (1 if women and 2 if men). We display part of the processed data in Table 5, where `Duration` is the sojourn time in `Prev_State`.

We investigate the transition dynamics and state persistence times of the `asthma` data set using the `BMRMM` function. We consider $K = 4$ mixture components for state persistence times. The choice of K is derived from running the BMRMM model several times with different K's and comparing the fitness of the models using the LPML and WAIC scores.

```
R> res.asm <- BMRMM(data = asthma, num.cov = 3, state.labels = c(1, 2, 3),
                    cov.labels = list(c('Mild-Moderate','Severe'),
                                      c('BMI<25','BMI>=25'),
                                      c('Women','Men')),
```



(a) For transition type $u \to m$ under covariates $\{F, U\}$.



(b) For gamma mixture component 2 shape and rate parameters.

Figure 6: Results for the simulated `foxp2` data set showing the MCMC diagnostic plots, including traceplots and autocorrelation plots.

```
duration.distr = list('mixgamma', shape = rep(1, 4),
                                  rate = rep(1, 4)))
```

We name the returned `BMRMM` object `res.asm` and obtain the `BMRMMsummary` object `sm.asm` by calling the `summary.BMRMM` function. As in the FoxP2 application, we first plot the global test results for both transition probabilities and state persistence times in Figure 7. For the transition probabilities, only the severity of asthma is significant while for duration times only the preceding state is significant. The BMI value and the sex are not significant for either transition dynamics or state durations.

```
R> sm.asm <- summary(res.asm)
R> sm.asm$trans.global
            label_data
cluster_data  BMI Severity  Sex
           1 0.96     0.00 1.00
           2 0.04     1.00 0.00
R> sm.asm$dur.global
```

Id	Severity	BMI	Sex	Prev_State	Cur_State	Duration
2	2	2	1	3	2	0.1533
2	2	2	1	2	2	4.1232
3	2	2	2	3	1	0.0958
3	2	2	2	1	3	0.2300
3	2	2	2	3	1	0.2656
3	2	2	2	1	1	5.4073

Table 5: Part of the `asthma` data set from the ARIA study of severe asthmatic patients.

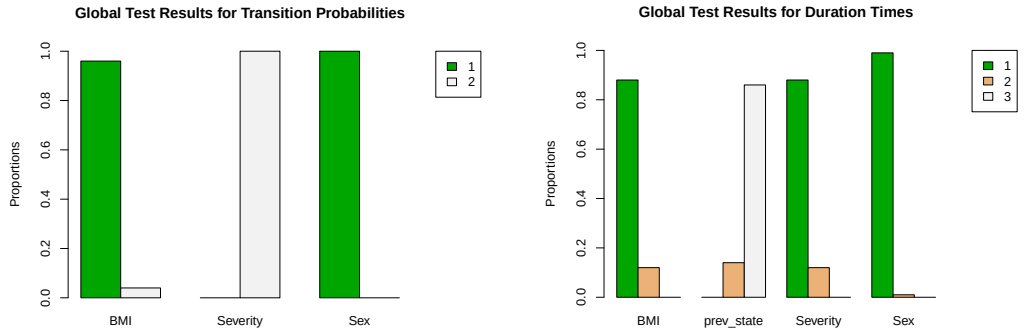


Figure 7: Results for the `asthma` data set showing the global tests of significance of the covariates for the state transitions (left) and the state persistence times (right). The bars represent the estimated posterior probabilities of the number of clusters formed by the levels of each covariate.

```
                    label_data
cluster_data  BMI prev_state Severity  Sex
           1 0.88       0.00     0.88 0.99
           2 0.12       0.14     0.12 0.01
           3 0.00       0.86     0.00 0.00
R> plot(sm.asm, 'trans.global')
R> plot(sm.asm, 'dur.global')
```

We show the posterior mean and standard deviations of the state transition probabilities for men and women with severe conditions and BMI $\geq$ 25 in Figure 8. We see that for severe patients with BMI $\geq$ 25, the transition probabilities are similar for men and women. We also take a look at the local test results for the BMI values fixing the severity of the patients in Figure 9. Though the absolute differences between the two covariate levels for BMI are small, the probabilities for the null hypotheses are also small, especially for transitions to state 1 and state 2. This suggests that even though the influence of BMI on state transitions is not significant globally, it is significant given the severity of asthma condition regardless of sex.

```
R> plot(sm.asm, 'trans.probs.mean')
R> plot(sm.asm, 'trans.probs.sd')
```

Figure 10a presents the histograms of the entire asthma data set, superimposed with the posterior mean of the mixture gamma distribution. With four components, the estimated mixture gamma fits the asthma data well. From the histogram for each component, we see from Figure 10b that component 1 and 2 represents shorter state persistence times while components 3 and 4 represent longer durations.

```
R> hist(res.asm, xlim = c(0,1))
R> for(comp in 1:4) {
     hist(res.asm, comp = comp)
   }
```

We investigate the covariates' influence by examining the mixture probabilities from the last MCMC iteration. An interesting discovery is that the mixture probabilities for both sexes, BMI levels, and severity levels are the same, indicating that the levels of these three covariates do not strongly influence the distributions of state durations. This matches the global test results in Figure 7. If the preceding state is state 1, which is optimal control for asthma, the state duration time is longer than that if the previous state is 2 or 3, as there is a lower weight in component 1 and higher weight in

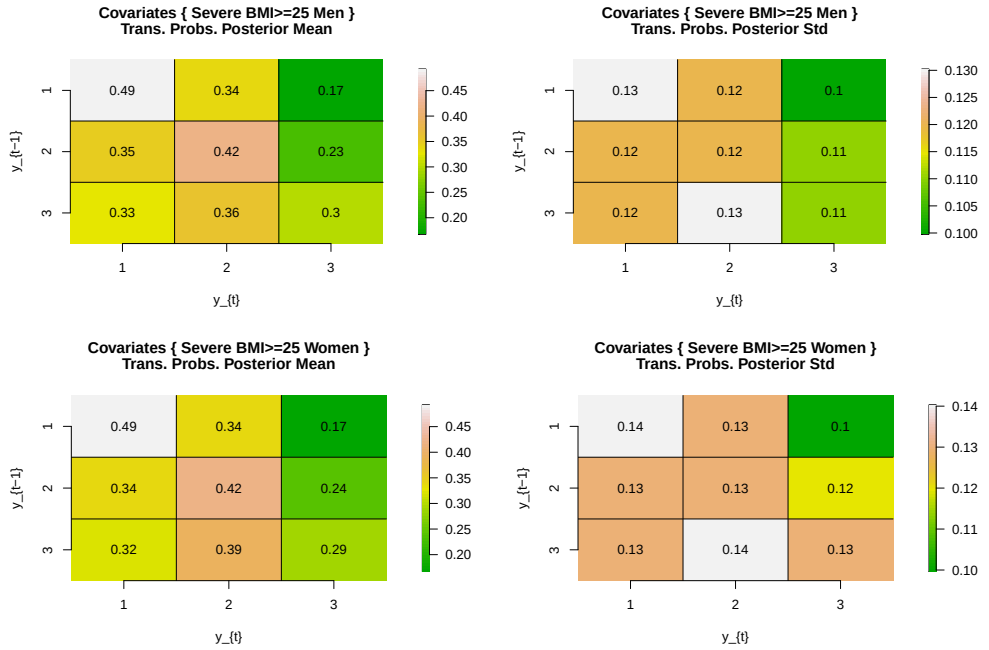


Figure 8: Results for the `asthma` data set showing the posterior mean and standard deviation for each transition type for selected covariate combinations, $\{\text{Severe}, \text{BMI} \geq 25, \text{Men}\}$ (top) and $\{\text{Severe}, \text{BMI} \geq 25, \text{Women}\}$ (bottom).

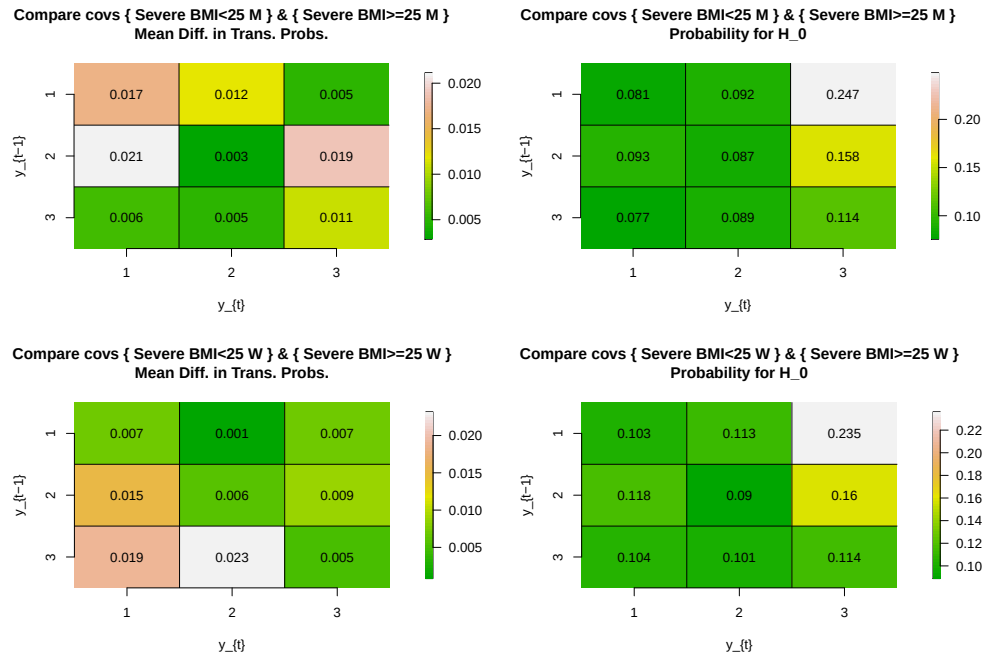
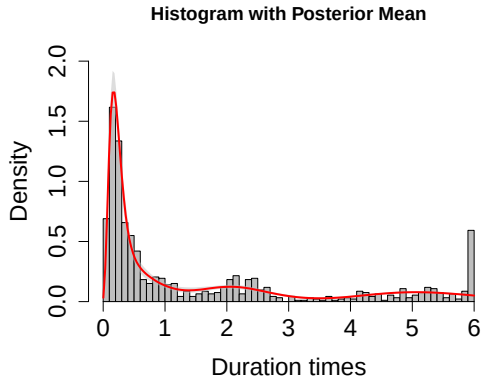


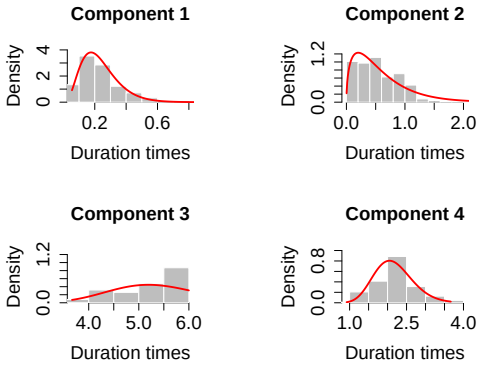
Figure 9: Results for the `asthma` data set showing local test results for BMI fixing asthma severity condition and sex, men (top) and women (bottom). The averaged absolute difference in transition probabilities between BMI < 25 and BMI ≥ 25 is presented on the left. The posterior probability of the null hypothesis is on the right.

components 3 and 4. On the other hand, if the preceding state is 3, which is unacceptable control, then the state duration time is much shorter, as the mixture probability in component 1 is much higher when the preceding state is state 3. We present some examples of the diagnostic plots for the `asthma` data set in Figure 11.

```
R> sm.asm$dur.mix.probs
$Severity
       Mild-Moderate Severe
Comp 1          0.37   0.37
```



(a) Histogram of ISIs with the estimated posterior mean (red line) of their marginal gamma mixture density averaged from recorded MCMC samples.



(b) Histograms of ISIs for each of the three components of the gamma mixture model along with the component density (red lines) from the last MCMC iteration.

Figure 10: Results for the `asthma` data set showing the histograms of ISIs with the estimated posterior gamma mixture density (left) and histograms of ISIs for each mixture component (right).

```
Comp 2          0.26   0.26
Comp 3          0.20   0.20
Comp 4          0.17   0.17

$BMI
       BMI<25 BMI>=25
Comp 1   0.37    0.37
Comp 2   0.26    0.26
Comp 3   0.20    0.20
Comp 4   0.17    0.17

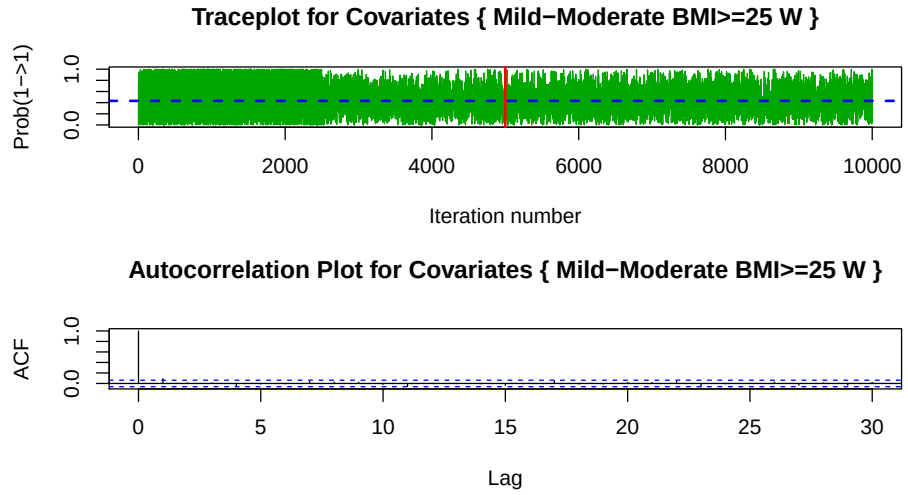
$Sex
       Women  Men
Comp 1  0.37 0.37
Comp 2  0.26 0.26
Comp 3  0.20 0.20
Comp 4  0.17 0.17

$prev_state
          1    2    3
Comp 1 0.27 0.33 0.51
Comp 2 0.23 0.33 0.23
Comp 3 0.31 0.20 0.09
Comp 4 0.19 0.15 0.17

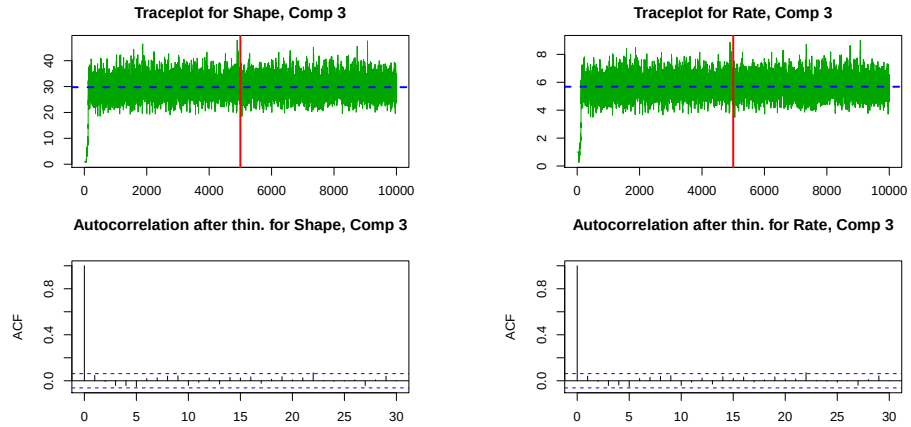
R> diag.BMRMM(res.fp2, cov.combs = list(c(1, 2, 1)),
              transitions = list(c(1, 1)), components = c(3))
```

6 Conclusion

We presented the **BMRMM** package which implements both Bayesian Markov mixed models (BMMM) for analyzing the state transitions and Bayesian Markov renewal mixed models (BMRMM) for additionally analyzing the duration times (being either state persistence times or inter-state intervals) in a collection of categorical sequences, using flexible Dirichlet and gamma mixtures, respectively. The BMRMM takes into account fixed effects of the associated covariates as well as random effects of the associated individuals while simultaneously selecting the significant covariates separately for the state transitions and the duration times. The package includes a synthetic `foxp2` data set to demonstrate the data framework and function usages. The package also provides a series of plotting functions for visualizing the results of the analyses, including various global and local hypotheses tests, MCMC diagnostics, etc. We are committed to maintaining and further developing the package in the future.



(a) For transition type $1 \to 1$ under covariates {Mild-Moderate, BMI ≥ 25, Women}.



(b) For gamma mixture component 3 shape and rate parameters.

Figure 11: Results for the `asthma` data set showing the MCMC diagnostic plots, including traceplots and autocorrelation plots.

Future improvements to the package may include more options for the distribution types of transition probabilities and duration times beyond the currently available mixture Dirichlet and mixture gamma distributions, respectively.

7 Acknowledgements

We thank two anonymous reviewers very much for their careful review of our work and their constructive comments that led to significant improvements to both the package and this paper.

References

A. Alioum and D. Commenges. MKVPCI: a computer program for Markov models with piecewise constant intensities and covariates. *Computer Methods and Programs in Biomedicine*, 64:109–119, 2001. URL https://doi.org/10.1016/s0169-2607(00)00094-8. [p193]

A. Alioum, V. Leroy, D. Commenges, F. Dabis, R. Salamon, G. d'Epidémiologie Clinique du SIDA en Aquitaine, et al. Effect of gender, age, transmission category, and antiretroviral therapy on the progression of human immunodeficiency virus infection using multistate Markov models. *Epidemiology*, 9:605–612, 1998. URL https://www.jstor.org/stable/3702781. [p192]

E. E. Alvarez. Estimation in stationary Markov renewal processes, with application to earthquake forecasting in Turkey. *Methodology and Computing in Applied Probability*, 7:119–130, 2005. URL https://doi.org/10.1007/s11009-005-6658-2. [p192]

M. Amini, A. Bayat, and R. Salehian. hhsmm: an R package for hidden hybrid Markov/semi-Markov models. *Computational Statistics*, 1:1–53, 2022. URL https://doi.org/10.1007/s00180-022-01248-x. [p193]

S. Bacallado, J. D. Chodera, and V. Pande. Bayesian comparison of Markov models of molecular dynamics with detailed balance constraint. *The Journal of Chemical Physics*, 131:1–10, 2009. URL https://doi.org/10.1063/1.3192309. [p192]

V. S. Barbu, C. Bérard, D. Cellier, M. Sautreuil, and N. Vergne. SMM: An R package for estimation and simulation of discrete-time semi-Markov models. *The R Journal*, 10:226–246, 2018. URL https://doi.org/10.32614/RJ-2018-050. [p193]

P. Bulla and P. Muliere. Bayesian nonparametric estimation for reinforced Markov renewal processes. *Statistical Inference for Stochastic Processes*, 10:283–303, 2007. URL https://doi.org/10.1007/s11203-006-9000-x. [p192]

G. A. Castellucci, M. J. McGinley, and D. A. McCormick. Knockout of Foxp2 disrupts vocal development in mice. *Nature Scientific Reports*, 6:1–14, 2016. URL https://doi.org/10.1038/srep23305. [p193]

J. Chabout, A. Sarkar, S. Patel, T. Raiden, D. B. Dunson, S. E. Fisher, and E. D. Jarvis. A Foxp2 mutation implicated in human speech deficits alters sequencing of ultrasonic vocalizations in adult male mice. *Frontiers in Behavioral Neuroscience*, 10:1–18, 2016. URL https://doi.org/10.3389/fnbeh.2016.00197. [p193, 199]

C. Combescure, P. Chanez, P. Saint-Pierre, J. P. Daures, H. Proudhon, P. Godard, et al. Assessment of variations in control of asthma over time. *European Respiratory Journal*, 22:298–304, 2003. URL https://doi.org/10.1183/09031936.03.00081102. [p193, 203]

E. Damsleth. Conjugate classes for gamma distributions. *Scandinavian Journal of Statistics*, 2:80–84, 1975. URL https://www.jstor.org/stable/4615580. [p196]

P. Diaconis and S. W. Rolles. Bayesian analysis for reversible Markov chains. *The Annals of Statistics*, 34:1270–1292, 2006. URL https://doi.org/10.1214/009053606000000290. [p192]

P. Eichelsbacher and A. Ganesh. Bayesian inference for Markov chains. *Journal of Applied Probability*, 39:91–99, 2002. URL https://www.jstor.org/stable/3215920. [p192]

I. Epifani, L. Ladelli, and A. Pievatolo. Bayesian estimation for a parametric Markov renewal model applied to seismic data. *Electronic Journal of Statistics*, 8:2264–2295, 2014. URL https://doi.org/10.1214/14-EJS952. [p192]

M. A. Etterson, B. Olsen, and R. S. Greenberg. The analysis of covariates in multi-fate Markov chain nest-failure models. *Studies in Avian Biology*, 34:55–64, 2007. URL https://sora.unm.edu/node/139709. [p192]

N. Ferguson, S. Datta, and G. Brock. msSurv: An R package for nonparametric estimation of multistate models. *Journal of Statistical Software*, 50:1–24, 2012. URL https://doi.org/10.18637/jss.v050.i14. [p193]

E. Fujita, Y. Tanabe, A. Shiota, M. Ueda, K. Suwa, M. Y. Momoi, and T. Momoi. Ultrasonic vocalization impairment of Foxp2 (R552H) knockin mice related to speech-language disorder and abnormality of Purkinje cells. *Proceedings of the National Academy of Sciences*, 105:3117–3122, 2008. URL https://doi.org/10.1073/pnas.0712298105. [p193]

S. Gaub, S. E. Fisher, and G. Ehret. Ultrasonic vocalizations of adult male Foxp2-mutant mice: behavioral contexts of arousal and emotion. *Genes, Brain and Behavior*, 15:243–259, 2016. URL https://doi.org/10.1111/gbb.12274. [p193]

S. Geisser and W. F. Eddy. A predictive approach to model selection. *Journal of the American Statistical Association*, 74:153–160, 1979. URL https://doi.org/10.2307/2286745. [p199]

W. Gradner. Analyzing sequential categorical data: Individual variation in Markov chains. *Psychometrika*, 55:263–275, 1990. URL https://doi.org/10.1007/BF02295287. [p192]

T. Holsclaw, A. M. Greene, A. W. Robertson, and P. Smyth. Bayesian nonhomogeneous markov models via pólya-gamma data augmentation with applications to rainfall modeling. *The Annals of Applied Statistics*, 11:393–426, 2017. URL https://doi.org/10.1214/16-AOAS1009. [p192]

M. A. Islam and R. I. Chowdhury. A higher order Markov model for analyzing covariate dependence. *Applied Mathematical Modelling*, 30:477–488, 2006. URL https://doi.org/10.1016/j.apm.2005.05.006. [p192, 193]

C. Jackson. Multi-state models for panel data: the msm package for R. *Journal of Statistical Software*, 38: 1–28, 2011. URL https://doi.org/10.18637/jss.v038.i08. [p193]

E. Juniper, P. O'byrne, G. Guyatt, P. Ferrie, and D. King. Development and validation of a questionnaire to measure asthma control. *European Respiratory Journal*, 14:902–907, 1999. URL https://doi.org/10.1034/j.1399-3003.1999.14d29.x. [p203]

K. Katsura. catdap, a categorical data analysis program package. *Computer Science Monograph*, 14, 1980. URL https://CRAN.R-project.org/package=catdap. [p193]

T. Kharrat, G. N. Boshnakov, I. McHale, and R. Baker. Flexible regression models for count data based on renewal processes: The Countr package. *Journal of Statistical Software*, 90:1–35, 2019. URL https://doi.org/10.18637/jss.v090.i13. [p193]

B. Li. Markov models for Bayesian analysis about transit route origin–destination matrices. *Transportation Research Part B: Methodological*, 43:301–310, 2009. URL https://doi.org/10.1016/j.trb.2008.07.001. [p192]

A. Listwon and P. Saint-Pierre. SemiMarkov: An R package for parametric estimation in multi-state semi-Markov models. *Journal of Statistical Software*, 66:1–16, 2015. URL https://doi.org/10.18637/jss.v066.i06. [p193, 203]

K. D. MacDermot, E. Bonora, N. Sykes, A.-M. Coupe, C. S. Lai, S. C. Vernes, F. Vargha-Khadem, F. McKenzie, R. L. Smith, A. P. Monaco, et al. Identification of FOXP2 truncation as a novel cause of developmental speech and language deficits. *The American Journal of Human Genetics*, 76:1074–1080, 2005. URL https://doi.org/10.1086/430841. [p193]

G. Marshall, W. Guo, and R. H. Jones. MARKOV: A computer program for multi-state Markov models with covariables. *Computer Methods and Programs in Biomedicine*, 47:147–156, 1995. URL https://doi.org/10.1016/0169-2607(95)01641-6. [p193]

D. Meyer, A. Zeileis, and K. Hornik. *vcd: Visualizing Categorical Data*, 2022. URL https://CRAN.R-project.org/package=vcd. R package version 1.4-10. [p193]

J. W. Miller. Fast and accurate approximation of the full conditional for gamma shape parameters. *Journal of Computational and Graphical Statistics*, 28:476–480, 2019. URL https://doi.org/10.1080/10618600.2018.1537929. [p196]

L. R. Muenz and L. V. Rubinstein. Markov models for covariate dependence of binary sequences. *Biometrics*, 41:91–101, 1985. URL https://doi.org/10.2307/2530646. [p192]

P. Muliere, P. Secchi, and S. G. Walker. Reinforced random processes in continuous time. *Stochastic Processes and their Applications*, 104:117–130, 2003. URL https://doi.org/10.1016/S0304-4149(02)00234-X. [p192]

J. O'Connell and S. Højsgaard. Hidden semi Markov models for multiple observation sequences: The mhsmm package for R. *Journal of Statistical Software*, 39:1–22, 2011. URL https://doi.org/10.18637/jss.v039.i04. [p193]

M. J. Phelan. Bayes estimation from a Markov renewal process. *The Annals of Statistics*, 18:603–616, 1990. URL https://doi.org/10.1214/aos/1176347618. [p192]

J. E. Pustejovsky. *ARPobservation: Simulating recording procedures for direct observation of behavior*, 2021. URL https://CRAN.R-project.org/package=ARPobservation. R package version 1.1. [p193]

R. Pyke. Markov renewal processes: definitions and preliminary properties. *The Annals of Mathematical Statistics*, 32:1231–1242, 1961. URL https://www.jstor.org/stable/2237923. [p193]

P. Saint-Pierre, C. Combescure, J. Daures, and P. Godard. The analysis of asthma control under a Markov assumption with use of covariates. *Statistics in Medicine*, 22:3755–3770, 2003. URL https://doi.org/10.1002/sim.1680. [p203]

A. Sarkar, J. Chabout, J. J. Macopson, E. D. Jarvis, and D. B. Dunson. Bayesian semiparametric mixed effects Markov models with application to vocalization syntax. *Journal of the American Statistical Association*, 113:1515–1527, 2018. URL https://doi.org/10.1080/01621459.2018.1423986. [p192, 193, 199]

M. Sesia, C. Sabatti, and E. J. Candès. Gene hunting with hidden Markov model knockoffs. *Biometrika*, 106:1–18, 2019. URL https://doi.org/10.1093/biomet/asy033. [p192]

M. Siebert and J. Söding. Bayesian Markov models consistently outperform PWMs at predicting motifs in nucleotide sequences. *Nucleic Acids Research*, 44:6055–6069, 2016. URL https://doi.org/10.1093/nar/gkw521. [p192]

G. A. Spedicato. Discrete time Markov chains with R. *The R Journal*, 9:84–104, 2017. doi: 10.32614/RJ-2017-036. URL https://doi.org/10.32614/RJ-2017-036. [p193]

S. Watanabe and M. Opper. Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research*, 11:3571–3594, 2010. URL https://doi.org/10.48550/arXiv.1004.2316. [p199]

Y. Wu, E. D. Jarvis, and A. Sarkar. Bayesian semiparametric Markov renewal mixed models for vocalization syntax. *Biostatistics*, 2023. URL https://doi.org/10.1093/biostatistics/kxac050. To appear. [p192, 193, 199]

M. Zhang, P. W. van Rijn, P. Deane, and R. E. Bennett. Scenario-based assessments in writing: An experimental study. *Educational Assessment*, 24:73–90, 2019. URL https://doi.org/10.1080/10627197.2018.1557515. [p193]

Yutong Wu
Department of Mechanical Engineering
The University of Texas at Austin
204 E Dean Keeton St C2200, Austin, TX 78712-1591
United States
ORCID: 0000-0001-7828-9981
yutong.wu@utexas.edu

Abhra Sarkar
Department of Statistics and Data Sciences
The University of Texas at Austin
2317 Speedway D9800, Austin, TX 78712-1823
United States
ORCID: 0000-0002-6924-8464
abhra.sarkar@utexas.edu

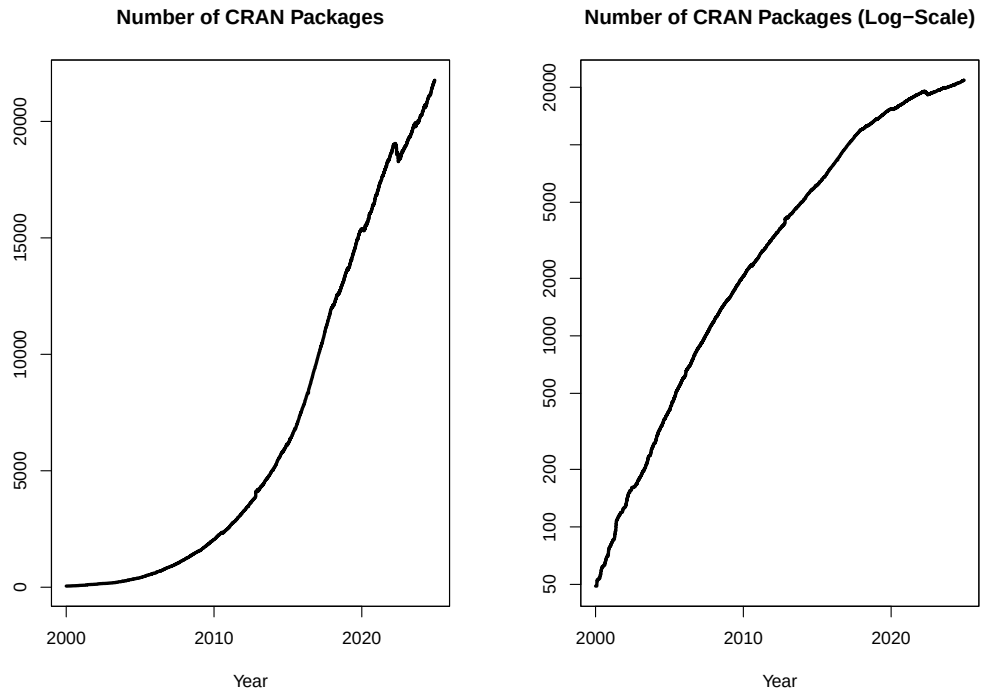
Changes on CRAN

2024-01-01 to 2024-06-30

by Kurt Hornik, Uwe Ligges, and Achim Zeileis

1 CRAN growth

In the past 6 months, 1060 new packages were added to the CRAN package repository. 405 packages were unarchived, 690 were archived and 7 had to be removed. The following shows the growth of the number of active packages in the CRAN package repository:



On 2024-06-30, the number of active packages was around 21018.

2 CRAN package submissions

From January 2024 to June 2024 CRAN received 14584 package submissions. For these, 23887 actions took place of which 16756 (70%) were auto processed actions and 7131 (30%) manual actions.

Minus some special cases, a summary of the auto-processed and manually triggered actions follows:

	archive	inspect	newbies	pending	pretest	publish	recheck	waiting
auto	4316	1674	3532	383	0	4588	1572	691
manual	2726	140	93	255	179	2881	638	219

These include the final decisions for the submissions which were

	archive	publish
auto	4083 (28.7%)	4045 (28.4%)
manual	2692 (18.9%)	3399 (23.9%)

where we only count those as *auto* processed whose publication or rejection happened automatically in all steps.

A new team member, Konstanze Lauseker, joined the CRAN submission team. Welcome, Konstanze. Unfortunately, Victoria Wimmer left the CRAN submission team after processing 4588 incoming submissions. Thanks a lot!

3 CRAN mirror security

Currently, there are 94 official CRAN mirrors, 73 of which provide both secure downloads via '`https`' *and* use secure mirroring from the CRAN master (via rsync through ssh tunnels). Since the R 3.4.0 release, `chooseCRANmirror()` offers these mirrors in preference to the others which are not fully secured (yet).

4 CRAN Task View Initiative

Currently there are 46 task views (see https://CRAN.R-project.org/web/views/), with median and mean numbers of CRAN packages covered 104 and 122, respectively. Overall, these task views cover 4711 CRAN packages, which is about 22% of all active CRAN packages.

Julia Piaskowski (University of Idaho) joined the team of CRAN Task View Editors, welcome!

Kurt Hornik
WU Wirtschaftsuniversität Wien
Austria
ORCiD: 0000-0003-4198-9911
Kurt.Hornik@R-project.org

Uwe Ligges
TU Dortmund
Germany
ORCiD: 0000-0001-5875-6167
Uwe.Ligges@R-project.org

Achim Zeileis
Universität Innsbruck
Austria
ORCiD: 0000-0003-0918-3766
Achim.Zeileis@R-project.org

R Foundation News

by Torsten Hothorn

1 Donations and members

Membership fees and donations received between 2024-04-12 and 2024-12-06.

1.1 Donations

Qualitas AG (Switzerland) Keith Chamberlain (United States) Lawrence Fredendall (United States) Thomas Hennequin (Netherlands) Calvin Hopper (United States) Emma Howard (Ireland) Roger Koenker (United Kingdom) Korea R User Group (Korea, Republic of) Plamen Vladkov Mirazchiyski (Slovenia) David Smith (United States) Tobias Strapatsas (Germany) Jason Wyse (Ireland)

1.2 Supporting benefactors

Zubin Dowlaty (United States)

1.3 Supporting institutions

Alfred Mueller Analytic Services, München (Germany) Digital Ecology Limited , Berkeley (United Kingdom) NIFU Nordic Institute for Studies in Innovation, Research and Education, Oslo (Norway) Roseburg Forest Products, Springfield (United States) The University of Auckland, Statistics Department, Auckland (New Zealand) University of Iowa, Iowa City (United States)

1.4 Supporting members

Douglas Adamoski (Brazil) Vedo Alagic (Austria) Tim Appelhans (Germany) Kristoffer Winther Balling (Denmark) Amit Behera (United States) Ashanka Beligaswatte (Australia) Nathan Bernhardt (United States) Chris Billingham (United Kingdom) Gordon Blunt (United Kingdom) Robert Carnell (United States) Ivan Maria Castellani (Italy) William Chiu (United States) Tom Clarke (United Kingdom) Giuseppe Corbelli (Italy) Rafael Costa (Brazil) Charles Cowens (United States) Terry Cox (United States) Alistair Cullum (United States) Robert Daly (Australia) Gergely Daroczi (Hungary) Ajit de Silva (United States) Elliott Deal (United States) Dubravko Dolic (Germany) Serban Dragne (United Kingdom) Mitch Eppley (United States) Guenter Faes (Germany) David Freedman (United States) Keita Fukasawa (Japan) Anne Catherine Gieshoff (Switzerland) Spencer Graves (United States) Susan Gruber (United States) Chris Hanretty (United Kingdom) James Harris (United States) Takehiko Hayashi (Japan) Kieran Healy (United States) ken ikeda (Japan) Knut Helge Jensen (Norway) Sebastian Jeworutzki (Germany) Brian Johnson (United States) Markus Kainu (Finland) Christian Kampichler (Netherlands) Katharina Kesy (Germany) An Khuc

(United States) Miha Kosmac (United Kingdom) Sebastian Krantz (Germany) Jan Herman Kuiper (United Kingdom) Teemu Daniel Laajala (Finland) Jindra Lacko (Czechia) Vishal Lama (United States) Bernardo Lares (Venezuela) Rory Lawless (United States) Thierry Lecerf (Switzerland) Seungdoe Lee (Korea, Republic of) Mauro Lepore (United States) Andrea Luciani (Italy) David Luckett (Australia) Sharon Machlis (United States) Mehrad Mahmoudian (Finland) Michal Majka (Austria) Amanuel Medhanie (United States) Bogdan-Alexandru Micu (Luxembourg) Igor Mikheenko (Russian Federation) harvey minnigh (Puerto Rico) Guido Möser (Germany) Markus Näpflin (Switzerland) Mark Niemann-Ross (United States) Jens Oehlschlägel (Germany) Dan Orsholits (Switzerland) George Ostrouchov (United States) Jaesung James Park (Korea, Republic of) Matt Parker (United States) josiah parry (United States) Elgin Perry (United States) Bill Pikounis (United States) Kelly Pisane (Netherlands) Paul Rayburn (Canada) Ramon Rodriguez-Santana (United States) David Romano (United States) Peter Ruckdeschel (Germany) Raoul Schorer (Switzerland) Dominic Schuhmacher (Germany) Dejan Schuster (Germany) Christian Seubert (Austria) Jagat Sheth (United States) Sindri Shtepani (Canada) David Sides (United States) Rachel Smith-Hunter (United States) Murray Sondergard (Canada) Matteo Starri (Italy) Marco Steenbergen (Switzerland) Berthold Stegemann (Germany) ROBERT Szabo (Sweden) Jan Tarabek (Czechia) Tim Taylor (United Kingdom) Chris Toney (United States) Nicholas Turner (United States) Philipp Upravitelev (Russian Federation) Mark van der Loo (Netherlands) Frans van Dunné (Costa Rica) Vincent van Hees (Netherlands) Marcus Vollmer (Germany) Jaap Walhout (Netherlands) Sandra Ware (Australia) Lim Zhong Hao (Singapore) 杨 (Yang) 胡 (Hu) (New Zealand)

Torsten Hothorn
Universität Zürich
Switzerland
ORCiD: 0000-0001-8301-0471
Torsten.Hothorn@R-project.org

Bioconductor Notes, March 2024

by Maria Doyle, Bioconductor Community Manager, and Bioconductor Core Developer Team

Abstract We discuss general project news.

1 Introduction

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. The project has entered its twentieth year, with funding for core development and infrastructure maintenance secured through 2025 (NIH NHGRI 2U24HG004059). Additional support is provided by NIH NCI, Chan-Zuckerberg Initiative, National Science Foundation, Microsoft, and Amazon. In this news report, we give some updates on core team and project activities.

2 Software

In October 2023, Bioconductor 3.18 was released*. It is compatible with R 4.3 and includes 2266 software packages, 429 experiment data packages, 920 up-to-date annotation packages, 30 workflows, and 4 books. Books are built regularly from source, ensuring full reproducibility; an example is the community-developed Orchestrating Single-Cell Analysis with Bioconductor.

**Note: Bioconductor 3.19 and 3.20 were subsequently released in May and October 2024, respectively. For details on the latest release, visit the Bioconductor website.*

3 Website Redesign

In January 2024, we unveiled the new Bioconductor.org, featuring a cleaner design, improved accessibility, and reorganized content. This redesign, shaped by community feedback, aims to better serve our global users. Looking ahead, we have identified the need to enhance search functionalities and improve how Bioconductor content is structured and integrated to support advanced tools, including AI. Planning is underway, with development expected to begin in 2025, subject to grant outcomes. See blog post for more details.

4 Community and Impact

4.1 Community Profile

At the end of 2023, the Center for Scientific Collaboration and Community Engagement (CSCCE) published a Bioconductor Community Profile. This report highlights the impact of Bioconductor's first year of CZI EOSS 4 funding, providing insights into our community's structure, challenges, and successes. Read the full profile here.

4.2 Outreachy Internships

Bioconductor participated in the Outreachy Internship program for the December 2023 – March 2024 cohort. Interns Chioma Onyido, Ester Afuape, and Peace Sandy from Nigeria contributed to curating microbiome studies for BugSigDB. They also shared their experiences working on these projects and engaging with the Bioconductor community in a blog post, which you can read here.

4.3 YERUN Open Science Award

In February 2024, the Bioconductor Community Advisory Board received the YERUN Open Science Award for advancing open-source software in biomedical research and promoting equitable access to genomic analysis tools. The €2,000 prize will fund a hackathon focused on AI-assisted translation of training materials. Learn more in the UL article.

4.4 Bioconductor Leaves Twitter/X

In December 2023, Bioconductor transitioned away from Twitter/X due to concerns about the platform's alignment with our Code of Conduct. The account is now archived, and we encourage the community to connect with us on platforms like Mastodon, LinkedIn, YouTube, Slack, and our mailing lists. Read the full announcement here.

5 Conferences

5.1 BioC2024 Announcement

The annual Bioconductor Conference, BioC2024, will take place in Grand Rapids, Michigan, from July 24–26, 2024. This event will feature keynote talks, workshops, and opportunities for community engagement. For more details, visit the conference website here.

5.2 EuroBioC2024 Announcement

The European Bioconductor Conference, EuroBioC2024, will be held in Oxford, UK, from September 4–6, 2024. Join us for discussions, tutorials, and networking with the Bioconductor community in Europe. More information is available here.

6 Boards and Working Groups Updates

If you are interested in becoming involved with any Bioconductor working group please contact the group leader(s).

6.1 EDAM Working Group Announcement

Bioconductor has launched an EDAM Working Group in collaboration with EDAM-bio.tools to improve the discoverability of Bioconductor packages through the EDAM ontology, a widely used bioinformatics vocabulary and classification system. This effort supports greater integration with communities beyond R and platforms like Galaxy and aligns with Bioconductor's mission of accessibility and interoperability. The group is submitting a proposal for the ELIXIR BioHackathon 2024, and invites interested contributors to join the discussion on the Bioconductor Slack in the #edam-collaboration channel.

7 Using Bioconductor

Start using Bioconductor by installing the most recent version of R and evaluating the commands

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install()
```

Install additional packages and dependencies, e.g., SingleCellExperiment, with

```
BiocManager::install("SingleCellExperiment")
```

Docker images provides a very effective on-ramp for power users to rapidly obtain access to standardized and scalable computing environments. Key resources include:

- bioconductor.org to install, learn, use, and develop Bioconductor packages.
- A list of available software linking to pages describing each package.
- A question-and-answer style user support site and developer-oriented mailing list.
- A community slack workspace (sign up) for extended technical discussion.
- The F1000Research Bioconductor gateway for peer-reviewed Bioconductor workflows as well as conference contributions.
- The Bioconductor YouTube channel includes recordings of keynote and talks from recent conferences, in addition to video recordings of training courses.

- Our package submission repository for open technical review of new packages.

Upcoming and recently completed events are browsable at our events page.

The Technical and and Community Advisory Boards provide guidance to ensure that the project addresses leading-edge biological problems with advanced technical approaches, and adopts practices (such as a project-wide Code of Conduct) that encourages all to participate. We look forward to welcoming you!

We welcome your feedback on these updates and invite you to connect with us through the Bioconductor Slack workspace or by emailing community@bioconductor.org.

Maria Doyle, Bioconductor Community Manager
University of Limerick

Bioconductor Core Developer Team
Dana-Farber Cancer Institute, Roswell Park Comprehensive Cancer Center, City University of New York, Fred Hutchinson Cancer Research Center, Mass General Brigham