Mohammed Saqr
Sonsoles López-Pernas *Editors*

Learning Analytics Methods and Tutorials

A Practical Guide Using R

OPEN ACCESS

Springer

Learning Analytics Methods and Tutorials

Mohammed Saqr • Sonsoles López-Pernas
Editors

Learning Analytics Methods and Tutorials

A Practical Guide Using R

Springer

Editors
Mohammed Saqr
School of Computing
University of Eastern Finland
Joensuu, Finland

Sonsoles López-Pernas
School of Computing
University of Eastern Finland
Joensuu, Finland

https://github.com/sn-code-inside/labook-code

ISBN 978-3-031-54463-7 ISBN 978-3-031-54464-4 (eBook)
https://doi.org/10.1007/978-3-031-54464-4

This work was supported by University of Eastern Finland.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To our families, friends, and loved ones.

A special dedication from Mohammed to his daughters, the exquisite roses, Carmen and Layla.

Foreword

Learning analytics and its sister community of educational data mining have, over the years, matured into a field where an increasing community of scholars and practitioners work together to understand the best methods for exploring the high-volume data available about learners and their learning. This has led to an increasing number of handbooks and online courses. However, none of these prior resources had focused on providing step-by-step tutorials for how to carry out the various types of analyses conducted in learning analytics. This book, by Saqr and López-Pernas, and their colleagues, fills this important gap, offering clear and concise step-by-step tutorials combined with high level explanations of why each step is necessary. As such, it is an important contribution to the field and represents a manuscript that I am sure many learners will find extremely valuable—both newcomers and relatively experienced practitioners looking to learn a new method.

Professor of education and computer science
University of Pennsylvania

Ryan Baker

Foreword

My introduction to learning analytics was unexpected. A brief conversation with George Siemens in early June 2010 sparked a transformative journey for me, with lasting impact on many. George proposed co-organizing a conference on "learning analytics." It was the first time I'd heard of the phrase. The phrase resonated deeply, connecting with my previous collaborations with Jelena Jovanovic, Chris Brooks, Marek Hatala, Griff Richards, Gord McCalla, Colin Knight, and other colleagues in Canada's LORNET research network (mid 2000s).

Excited to collaborate, I was initially a bit skeptical about George's ambition to organize the conference by September 2010. However, I learned to appreciate his eagerness to implement ideas swiftly. We eventually reconnected and agreed on a late February 2011 date, giving us extra six months. Despite remaining nervous, I underestimated the potential impact. The experience proved me wrong in the most positive way, and I'm forever grateful to George, a dear friend, for his unwavering vision and belief.

The conference we organized, the 1st International Conference on Learning Analytics and Knowledge (LAK), was held in Banff, Canada, in late February and early March 2011. The frigid temperatures (–35 °C) were matched by the conference's energy. It attracted a large audience and fostered an open, vibrant, productive, and transformative community of researchers and practitioners. This conference is often considered the birth of learning analytics, and the definition we used in the chairs' message in the conference proceedings remains widely used in the field.

Learning analytics is now a well-established field. Since our early conversations, we've emphasized the importance of methods. My dear friend and esteemed colleague, Shane Dawson, a co-founder of the Society for Learning Analytics Research and field pioneer, first referred to learning analytics as a "bricolage field," one that borrows methods and theories from data science, artificial intelligence, network science, psychology, psychometrics, sociology, and natural language processing.

As founding editors-in-chief of the *Journal of Learning Analytics*, Shane and I recognized the need to introduce and promote diverse learning analytics methods. This led us to collaborate with Mykola Pechenizkiy, then-president of the Inter-

national Educational Data Mining Society, to edit a special section on learning analytics tutorials featuring five papers on different methods. These papers, based on tutorials and workshops from the first two Learning Analytics Summer Institutes, became some of the journal's most impactful publications. It demonstrated the need for learning analytics methods tutorials. However, 8 years have passed, a significant timeframe considering the rapid advancements in artificial intelligence and the methods they've driven.

Therefore, I'm delighted to see *Learning Analytics Methods and Tutorials* edited by Mohammed Saqr and Sonsoles López-Pernas. This timely book addresses a critical need in learning analytics.

The book offers a comprehensive guide to data analysis methods for researchers and practitioners of all experience levels. It starts with the basics, equipping beginners with R programming and data analysis skills through chapters on data cleaning and exploration. These skills are fundamental for understanding student data and preparing it for further analysis. Even for experts, the book offers advanced methods while emphasizing the broader applicability of these techniques beyond education.

The core of the book explores various analytical approaches. Machine learning methods receive well-deserved attention, including introductions to commonly used methods—specifically, predictive modeling and clustering. Predictive modeling is frequently used in learning analytics to identify at-risk students or classify online discussions by analyzing past data patterns. This allows for early intervention to support students at different progress levels. Cluster analysis, another machine learning technique, groups students based on similar characteristics, behaviors, or learning outcomes. It's commonly used to analyze learning strategies in different learning environments and can provide educators with valuable insights to tailor teaching support to diverse student needs.

We've long recognized the dynamic nature of learning, with many learning processes unfolding over time. This highlights the need for temporal analytic methods in learning analytics. I'm pleased to see the book provides a rich guide to various temporal methods that analyze the order and timing of events in data about learning and learners. Techniques like sequence analysis and process mining leverage student activity traces to understand how learning unfolds over time. This is crucial for studying longitudinal processes like student engagement throughout a program or explaining connections between cognitive and metacognitive processes in solo and group learning activities.

Network analysis has been a prominent methodological approach since the early days of learning analytics. The book offers excellent coverage of network analytic approaches that explore the relational aspects of data about learners and learning environments. By examining interactions between learners, teachers, and topics, researchers and practitioners can understand collaboration patterns and identify student communities. These methods can be further enriched by combining them with temporal analysis to explore how these relationships evolve over time. The book also covers recent advancements in quantitative ethnography, introducing

epistemic network and ordered network analysis, techniques I've extensively used with collaborators in recent years.

I have recently been advocating for the need to establish stronger collaborative ties between learning analytics and psychometrics. Psychometrics is a well-established discipline that can offer many relevant methods to learning analytics. Specifically, psychometrics can help us address issues that have received insufficient attention in learning analytics—reliability and validity. Without addressing these issues, learning analytics can be jeopardized. Therefore, I am delighted that the final section of the book delves into psychometrics, a field that investigates relationships between psychological constructs (like metacognition) and observable data (like test scores). The book explores techniques like factor analysis and structural equation modeling, which help researchers test hypotheses and theories about these relationships. By mastering these methods, learning analytics researchers and practitioners can gain valuable insights from student data to improve learning experiences and outcomes.

The book editors and chapter authors must be commended for their valuable contributions to learning analytics. What is outstanding about this book is that it provides source code illustrating all the methods introduced. Readers can directly use the code to build hands-on skills on the many high-importance methods for learning analytics that are so thoughtfully introduced in this book. The book can be directly used in any graduate or postgraduate course on learning analytics. I wish this book had been available 3 years ago when we developed the graduate certificate in learning analytics at Monash University. Many of its chapters would have been directly used as part of the graduate certificate program. And, I can easily see that many similar programs around the world will greatly benefit from this outstanding book. The book is equally suitable for practitioners and researchers in education institutions, schools, or industry who either want to develop basic data analytic skills or advance their skills with methods they haven't used before.

The overall learning analytics community is significantly richer because of this book. For that, we all owe immense thanks to Mohammed Saqr and Sonsoles López-Pernas, the book editors, for their incredible leadership and vision!

Dragan Gašević

Distinguished Professor of Learning Analytics,
Director of Research in the Department of Human Centred Computing,
Director of the Centre for Learning Analytics,
Monash University
Clayton, VIC, Australia

Preface

Learning analytics is a fast-paced discipline at the cross-section of cutting-edge methodological advances and theoretical innovations. Most of the methods have been born long before the field of learning analytics and continue to be developed and advanced across diverse fields. Therefore, researchers and students have to navigate through fragmented literature from several fields and learn to apply such methods. Notwithstanding the difficulties of identifying which literature or guides one can use, several methodological questions remain unanswered in these books. This is of course because such literature was prepared for other disciplines and offers case studies that are oftentimes not relevant.

Being learning analytics researchers ourselves, we have been through these difficulties. Learning a new methodology took several trials and searching for answers that were rather hard to get. Collaboration with other researchers from learning analytics and other fields—e.g., statistics and methodology—has helped us navigate these difficulties. In many situations, we would contact statisticians or package developers to help with solving an issue or collaborate on a paper. With the publication of our papers, we got several emails from researchers asking about technical details. It has become clear to us that there is a need for a well-documented resource for learning analytics methods. In fact, this is how we—the editors of this book—met. Sonsoles was doing mobility in Sweden, where Mohammed was working, and she wanted to do a learning analytics study for a dataset she collected from a programming course. Lacking any resource that could help her kick-start her exploration of the emerging field, she sought collaboration. Ever since, we had an interesting journey of working together on so many projects.

The lack of resources and methodological guidance was a problem then and continues to be a problem today. We thought that the arduous journey in learning analytics should not be endured by everyone, and we decided to make that resource with the help of the community as well as our collaborators. More than a year of intensive work is within your hands now. We hope that we have contributed to LA by providing an informative resource that students, researchers, and practitioners can use.

In this book, we tried to include all the basics of R as a programming language as well as the basics of data cleaning, statistics, and data manipulation. In doing so, we wanted the newcomers to find an easy entry to the field. We also tried to be as comprehensive as we could and included almost all major methodologies. For every method, we started with the basics, explaining the main concepts, the essential techniques, and basic functions. In subsequent chapters, we went deeper into advanced methods that are at the forefront of novel methodological innovations. We also used real-life learning analytics data and made it readily available to researchers. To do so, we collaborated with world-renowned researchers, package developers, and methodological experts from other fields to offer an unprecedented resource on novel topics that are hard to find any resource for.

We hope the readers find this book useful as a guide through learning analytics methods, highlighting the ways in which data-driven insights can benefit educators, learners, and researchers.

Joensuu, Finland

Mohammed Saqr
Sonsoles López-Pernas

Competing Interests

The authors have no conflicts of interest to declare that are relevant to the content of this book.

Acknowledgments

We are thankful for the great work everyone has done to make this book happen: authors have spent valuable time compiling these chapters and bringing their expertise to the wider audience of learning analytics. Authors from other fields did a great job and took the effort to navigate concepts they had little contact with to present their methods to a completely different audience than what they are used to.

We are also grateful to the reviewers who offered valuable feedback to refine and improve the chapters.

A special thanks goes to the associate editors who worked with us all over the last year to process this large number of chapters and handle the peer review process.

We would like to acknowledge the generous funding by the Academy of Finland (Research Council of Finland) for the project TOPEILA, Decision Number 350560, which was received by Mohammed Saqr.

Lastly, we would like to thank the generous funding from the University of Eastern Finland and, in particular, Markku Tukianen and Kari Lehtinen, who made this book open access for everyone.

Contents

List of Contributors

Editors

Mohammed Saqr University of Eastern Finland, Joensuu, Finland

Sonsoles López-Pernas University of Eastern Finland, Joensuu, Finland

Associate Editors

Miguel Ángel Conde-González University of León, León, Spain

Rogers Kaliisa University of Oslo, Oslo, Norway

Kamila Misiejuk University of Bergen, Bergen, Norway

Authors

Emorie Beck UC Davis, Davis, CA, USA

Javier Conde Universidad Politécnica de Madrid, Madrid, Spain

Miguel Ángel Conde-González University of León, León, Spain

Carlos Cuenca-Enrique Universidad Politécnica de Madrid, Madrid, Spain

Laura Del-Río-Carazo Universidad Politécnica de Madrid, Madrid, Spain

Marion Durand University of Eastern Finland, Joensuu, Finland

Merja Heinäniemi University of Eastern Finland, Kuopio, Finland

Jouni Helske University of Jyväskylä, Jyväskylän yliopisto, Finland

Satu Helske University of Turku, Turku, Finland

Ángel Hernández-García Universidad Politécnica de Madrid, Madrid, Spain

Rogers Kaliisa University of Oslo, Oslo, Norway

Jelena Jovanovic University of Belgrade, Belgrade, Serbia

Joran Jongerling Tilburg University, Tilburg, Netherlands

Juho Kopra University of Eastern Finland, Joensuu, Finland

Sonsoles López-Pernas University of Eastern Finland, Joensuu, Finland

Kamila Misiejuk University of Bergen, Bergen, Norway

Keefe Murphy Maynooth University, Maynooth, Ireland

Gilbert Ritschard University of Geneva, Geneva, Switzerland

A. R. Ruis University of Wisconsin, Madison, WI, USA

Mohammed Saqr University of Eastern Finland, Joensuu, Finland

Marieke J. Schreuder KU Leuven, Leuven, Belgium

Luca Scrucca Università degli Studi di Perugia, Perugia, Italy

Matthias Studer University of Geneva, Geneva, Switzerland

David Shaffer University of Wisconsin, Madison, WI, USA

Zachari Swiecki Monash University, Clayton, Australia

Yuanru Tan University of Wisconsin, Madison, WI, USA

Santtu Tikka University of Jyväskylä, Jyväskylä, Finland

Adrienne Traxler University of Copenhagen, København, Denmark

Leonie V. D. E. Vogelsmeier Tilburg University, Tilburg, Netherlands

Reviewers

Gökhan Akçapınar Hacettepe University, Ankara, Turkey

Daniel Amo-Filva Universitat Ramon Llull, Barcelona, Spain

Enrique Barra Universidad Politécnica de Madrid, Madrid, Spain

Umar Bin Qushem University of Turku, Turku, Finland

Manuel Castejón-Limas Universidad de León, León, Spain

Javier Conde Universidad Politécnica de Madrid, Madrid, Spain

Tudor Cristea TU Eindhoven, Eindhoven, Netherlands

Laura Del-Río-Carazo Universidad Politécnica de Madrid, Madrid, Spain

Ramy Elmoazen University of Eastern Finland, Joensuu, Finland

Dilrukshi Gamage Tokyo Tech, Tokyo, Japan

Brian P. Godor Erasmus University Rotterdam, Rotterdam, Netherlands

Francisco José García Peñalvo Universidad de Salamanca, Salamanca, Spain

Sami Heikkinen LAB University of Applied Sciences, Lahti, Finland

Rogers Kaliisa University of Oslo, Oslo, Norway

Qinyi Liu University of International Business and Economics, Beijing, China

Sonsoles López-Pernas University of Eastern Finland, Joensuu, Finland

Kamila Misiejuk University of Bergen, Bergen, Norway

Andrés Munoz-Arcentales Universidad Politécnica de Madrid, Madrid, Spain

Nicolae Nistor Ludwig-Maximilians-Universität München, Munich, Germany

Merlijn Olthof Radboud University, Nijmegen, Netherlands

Dijana Oreški Sveučilište u Zagrebu, Zagreb, Croatia

Sambit Praharaj Ruhr University Bochum, Bochum, Germany

Miroslava Raspopović Milić Belgrade Metropolitan University, Belgrade, Serbia

Mohammed Saqr University of Eastern Finland, Joensuu, Finland

Yuanru Tan University of Wisconsin-Madison, Madison, WI, USA

Santtu Tikka University of Jyväskylä, Jyväskylä, Finland

Tiina Törmänen University of Oulu, Oulu, Finland

Adrienne Traxler University of Copenhagen, Copenhagen, Denmark

Daphne van de Bongardt Erasmus University Rotterdam, Rotterdam, Netherlands

Thijs Vroegh Netherlands eScience Center, Amsterdam, Netherlands

Jin Zhou Central China Normal University, Wuhan, Hubei, China

List of Abbreviations

AGNES	Agglomerative Nesting
AHC	Agglomerative Hierarchical Clustering
AIC	Akaike Information Criterion
ANOVA	Analysis of Variance
ASW	Average Silhouette Width
AUC	Area Under the Curve
BIC	Bayesian Information Criterion
DFG	Directly Follows Graph
DIANA	Divisive Analysis
DNA	Deoxyribonucleic Acid
ECTS	European Credit Transfer System
EEG	Electroencephalogram
EFA	Exploratory Factor Analysis
EM	Expectation-Maximization
ENA	Epistemic Network Analysis
FMM	Finite Mixture Model
FN	False Negative
FP	False Positive
GBTM	Growth-Based Trajectory Model
GMM	Gaussian Mixture Modeling
GPA	Grade Point Average
HMM	Hidden Markov Model
ICL	Integrated Complete Likelihood
IDE	Integrated Development Environment
IP	Internet Protocol
IQR	Interquartile Range
JPEG	Joint Photographic Experts Group
KMO	Kaiser-Meyer-Olkin
KMS	Knowledge Management System
LA	Learning Analytics
LATUX	Learning Awareness Tools–User eXperience

LCA	Latent Class Analysis
LCP	Longest Common Prefix
LCS	Longest Common Subsequence
LMS	Learning Management System
LPA	Latent Profile Analysis
MAE	Mean Absolute Error
MAP	Maximum A Posteriori
MAR	Missing at Random
MBC	Model-Based Clustering
MDS	Multidimensional Scaling
MHMM	Mixture Hidden Markov Model
MI	Multiple Imputation
ML	Maximum Likelihood
MLE	Maximum Likelihood Estimation
MLR	Robust Maximum Likelihood
MM	Markov Model
MMM	Mixture Markov Model
MOOC	Massive Open Online Course
MPQ	Multicultural Personality Questionnaire
MULAS	Model of User-Centered Learning Analytics Systems
OM	Optimal Matching
ONA	Ordered Network Analysis
PAM	Partitioning Around Medoids
RF	Random Forest
RMSE	Root Mean Square Error
RMSEA	Root Mean Square Error of Approximation
ROC	Receiver Operating Characteristic
RSS	Really Simple Syndication
SD	Standard Deviation
SEM	Structural Equation Modeling
SNA	Social Network Analysis
SRL	Self-Regulated Learning
SRMR	Standardized Root Mean Square Residual
TAM	Technology Acceptance Model
TWCSS	Total Within-Cluster Sum-of-Squares
TN	True Negative
TP	True Positive
TSS	Total Sum of Squares
VIF	Variance Inflation Factor
WCTD	Within-Cluster Total Distance

Capturing the Wealth and Diversity of Learning Processes with Learning Analytics Methods

Sonsoles López-Pernas, Kamila Misiejuk, Rogers Kaliisa, Miguel Ángel Conde-González, and Mohammed Saqr

1 Introduction

The official birth of the field of learning analytics is often ascribed to the first Learning Analytics & Knowledge Conference in 2011, where the widely used definition was coined as "the measurement, collection, analysis, and reporting of data about learners and their contexts, for purposes of understanding and optimizing learning and the environments in which it occurs" [1]. Over the years, learning analytics has grown in scope, scale, and diversity and has since attracted researchers from diverse backgrounds bringing several disciplines together under one umbrella. Such increasing interest has resulted in a large number of publications, research groups, specialized units, and funding of large projects. Although other data-intensive fields started before learning analytics (e.g., academic analytics in 2007 and educational data mining in 2008), interest in such methods—and in closely related fields at large—has surged only after the rapid adoption of learning analytics. The unique position of learning analytics at the intersection of education and computer science while reaching out to several other disciplines such as statistics, psychometrics, econometrics, mathematics, and linguistics has accelerated the

S. López-Pernas (✉) · M. Saqr
School of Computing, University of Eastern Finland, Joensuu, Finland
e-mail: sonsoles.lopez@uef.fi

K. Misiejuk
Centre for the Science of Learning & Technology (SLATE), University of Bergen, Bergen, Norway

R. Kaliisa
Department of Education, University of Oslo, Oslo, Norway

M. Á. Conde-González
Robotics Research Group, Engineering School, University of León, León, Spain

M. Saqr, S. López-Pernas (eds.), *Learning Analytics Methods and Tutorials*,
https://doi.org/10.1007/978-3-031-54464-4_1

growth and expansion of the field. Therefore, it is a crucial endeavor for learning analytics researchers to stay abreast of the latest methodological and computational advances to drive their research forward. The diversity and complexity of the existing methods can make this task overwhelming both for newcomers to the learning analytics field and for experienced researchers. When conducting learning analytics research, researchers need to decide which data can and must be collected from learners to give answers to their research questions. They need to understand and decide which analytical methods may apply to such data and their potential limitations. They also need to interpret the findings and contextualize them in light of the existing literature and learning theories. With the motivation to accompany researchers in this challenging journey, this book aims to provide a methodological guide for researchers to study, consult, and take the first steps toward innovation.

Every method described in this book was developed before learning analytics, for instance, predicting students' performance can be traced back to almost a century ago [2], and using social network analysis to understand students' networks dates back to over five decades ago [3]. Similarly, process mining, sequence analysis, and Markov models can all be traced to times before the birth of the field of learning analytics (e.g., [4–6]). Not only were these methods born outside learning analytics, but they also continue to be developed, refined and advanced in their relevant fields. Nevertheless, learning analytics has succeeded in taking advantage of these methodological developments as well as the increasingly available digital data, computational resources, and data science, and in popularizing data-intensive research in education to bring this field together [7].

The diversity of methods and fields that have given rise to the field of analytics is evident in the list of authors of the chapters of this book, which include authors of R packages, and experts in methods and applications to drive a state-of-the-art blend. In the first category, we have the world renowned sequence analysis experts Gilbert Ritschard and Matthias Studer (University of Geneva), who are the creators of `TraMineR`, which is the most central R package for sequence analysis. Closely related, we have Satu and Jouni Helske (University of Turku), developers of `seqHMM`, an innovative R package for Mixture Hidden Markov Models of sequential data with tons of visualizations and statistical analysis potentials. Also, our list of authors includes Luca Scrucca (University of Perugia), author of `mclust`, one of the most widely used clustering packages for classification, and density estimation using Gaussian finite mixture models, and Keefe Murphy (Maynooth University), creator of several R packages such as `MoEClust` for Gaussian parsimonious clustering models with covariates. Our list of authors also includes David Shaffer (University of Wisconsin-Madison), the creator of `rENA` (an R package for Epistemic Network Analysis) along with other members of the `rENA` and the Ordered Network Analysis groups: Yuanru Tan and Andrew Ruis (University of Wisconsin-Madison), and Zachari Swiecki (Monash University). We also have Leonie V.D.E. Vogelsmeier (Tilburg University), author of `lmfa`, an innovative R package for latent Markov factor analysis which offers transition, and mixture factor analysis. We have Santtu Tikka (University of Jyväskylä),

author of the `dynamite` R package for dynamic multivariate panel models. In addition, among our authors, we have prominent methodology experts in several domains: Joran Jongerling (Tilburg University); Emorie Beck (UC Davis); Merja Heinäniemi and Juho Kopra (University of Eastern Finland), and Marieke Schreuder (KU Leuven). Lastly, we count on several senior and emerging learning analytics researchers: Jelena Jovanović (University of Belgrade); Ángel Hernández-García, Javier Conde, Laura Del-Río-Carazo, and Carlos Cuenca-Enrique (Universidad Politécnica de Madrid); Adrienne Traxler (University of Copenhagen); Kamila Misiejuk (University of Bergen); Miguel Ángel Conde (University of Leon), and Marion Durand (University of Eastern Finland). Besides, the editors (Mohammed Saqr and Sonsoles López-Pernas) from the University of Eastern Finland have a long history of learning analytics research working with diverse methods and interdisciplinary research that spans most learning analytics methods.

Thanks to the breadth and diversity of authors' backgrounds and expertise, the book offers a comprehensive array of methods that are described thoroughly. A step-by-step tutorial using the R programming language with real-life datasets and case studies is presented for each method. The book starts with an introductory section for readers to get up-to-speed with the R programming language, in which we cover the basics of the language, data preprocessing, basic statistics, and data visualization. Then, we move to classic machine learning methods, such as prediction and clustering applied to educational data. These methods enable readers to predict achievement, dropout, and students at risk, as well as to group students into different groups or profiles according to certain characteristics such as motivation or engagement. This is followed by an extensive section devoted to temporal methods such as sequence analysis, Markovian modeling, and process mining, which allow taking advantage of the endless possibilities of trace log data. The book then moves on to discuss network analysis in its many forms including social network analysis, epistemic network analysis, ordered network analysis, and temporal networks. Such methods are crucial for understanding collaboration and relationships between individuals and concepts, which are key aspects of learning. The book concludes with a section on psychometrics such as psychological networks, factor analysis, and structural equation modeling, which are fundamental tools for the analysis of self-reported data among others. We hope the readers find this book useful as a guide through learning analytics methods, highlighting the ways in which data-driven insights can benefit educators, learners, and researchers alike.

The book targets learning analytics researchers (and education researchers at large) at all stages. It is suitable to teach newcomers to the field, even with no experience in R, since the introductory chapters are aimed at getting readers acquainted with the basics of R programming and data analysis. The book also covers advanced methods that may be of interest to experts in the field of learning analytics or data science in general. Moreover, the skills taught are transferable to other fields, i.e., can be applied in other contexts outside education.

We hope the readers find this book useful as a guide through learning analytics methods, highlighting the ways in which data-driven insights can benefit educators, learners, and researchers.

2 How the Book Is Structured

2.1 Introductory Chapters

The first section of the book provides the basis for getting up to speed with the R programming language and the data that will be analyzed throughout the book. This section covers the fundamental steps of the data analysis process, such as data preprocessing and exploratory analysis. During data preprocessing, educational data is cleaned and prepared for further analysis. Many crucial decisions about building and conceptualizing learning indicators from raw data are made in this essential step of the learning analytics process. Exploratory analysis enables an early detection of interesting phenomena that can be discovered in the data using visualizations or simple statistics. Using these techniques helps to guide the direction of the in-depth analysis and the selection of more advanced analytical methods.

Chapter 2: A Broad Collection of Datasets for Educational Research Training and Application [8]

Sonsoles López-Pernas, Mohammed Saqr, Javier Conde, Laura Del-Río-Carazo

Since the goal of this book is to provide a guide and tutorial on how to implement learning analytics methods, the use of relevant data is a key aspect of the contextualization of these methods within learning analytics research. Chapter 2 kicks off the book with an introduction to the most relevant types of data in learning analytics and provides a diverse collection of curated datasets that we will use throughout the book to illustrate the different methods. Understanding the data under examination is a crucial step for the interpretability of the analyses that we will learn to perform in this book, and therefore, readers should familiarize themselves with the datasets described in this chapter to facilitate following the tutorials presented in subsequent ones.

Chapter 3: Getting Started with R for Education Research [9]

Santtu Tikka, Juho Kopra, Merja Heinäniemi, Sonsoles López-Pernas, Mohammed Saqr

The first tutorial-like chapter of the book provides an introduction to the basics of R programming, with a focus on the Rstudio integrated development environment and the `tidyverse` programming paradigm. The R programming language has become a popular tool for conducting data analysis in the field of learning analytics. The chapter covers topics such as data types and structures, control structures, pipes, functions, loops, and input/output operations. By the end of the chapter, readers should have a solid understanding of the basics of R programming and have the necessary tools to learn more in-depth topics such as data wrangling and basic statistics using R.

Chapter 4: An R Approach to Data Cleaning and Wrangling for Education [10]

Juho Kopra, Santtu Tikka, Merja Heinäniemi, Sonsoles López-Pernas, Mohammed Saqr

After learning the basics of the R programming language, Chap. 4 goes one step further by introducing the reader to data wrangling, also known as data cleaning and preprocessing. Data wrangling is a critical step in the data analysis process, particularly in the context of learning analytics. This chapter provides an introduction to data wrangling using R and covers topics such as data importing, cleaning, manipulation, and reshaping with a focus on tidy data. Specifically, readers will learn how to read data from different file formats (e.g., CSV, Excel), how to manipulate data using the `dplyr` package, and how to reshape data using the `tidyr` package. Additionally, the chapter covers techniques for combining multiple data sources.

Chapter 5: Introductory Statistics with R for Educational Researchers [11]

Santtu Tikka, Juho Kopra, Merja Heinäniemi, Sonsoles López-Pernas, Mohammed Saqr

Statistics play a fundamental role in learning analytics, providing a means to analyze and make sense of the vast amounts of data generated by learning environments, visualize relationships, test hypotheses, and make comparisons. Chapter 5 in this book provides an introduction to basic statistical concepts using R and covers topics such as measures of central tendency, variability, correlation, and regression analysis. Specifically, readers will learn how to compute descriptive statistics, test hypotheses, and perform simple linear regression analysis. The chapter also includes practical examples using realistic data sets from the field of learning analytics. By the end of the chapter, readers should have a solid understanding of the basic statistical concepts and methods commonly used in learning analytics, as well as a practical understanding of how to use R to conduct statistical analysis of learning data.

Chapter 6: Visualizing and Reporting Educational Data with R [12]

Sonsoles López-Pernas, Kamila Misiejuk, Santtu Tikka, Mohammed Saqr, Juho Kopra, Merja Heinäniemi

Visualizing data is central in learning analytics research, underpins learning dashboards, and is a prime method for reporting results and insights to stakeholders. Chapter 6 guides the reader through the process of generating meaningful and aesthetically pleasing visualizations of different types of datasets using well-known R packages. The main visualization types will be demonstrated with an explanation of their usage and use cases. Furthermore, learning-related examples will be discussed in detail. For instance, readers will learn how to visualize learners' logs extracted from learning management systems to show how trace data can be used to track students' learning activities. Readers will also be able to generate professional-looking tables with summary statistics.

2.2 Machine Learning Methods

The next section follows with some of the classic machine learning methods of learning analytics, which date to the early beginnings of the field: predictive

modelling and cluster analysis. Predictive modelling is a supervised learning method used widely in learning analytics research, where past data patterns are analysed to predict students' future outcomes. Clustering is an unsupervised learning method that detects similar patterns in the data and is typically used to group students based on their personal characteristics, observed behavior, or learning outcomes. Both methods are applied to address various challenges in education, such as preventing student drop-out, comparing strategies to improve academic performance, or identifying disengaged students. In addition, the results are often used to trigger specific interventions to help students succeed or to raise awareness about students' performance based on specific indicators.

Chapter 7: Predictive Modelling in Learning Analytics Using R [13]
Jelena Jovanovic, Sonsoles López-Pernas, Mohamed Saqr

Prediction of learners' course performance has been a central theme in learning analytics since the inception of the field. The main motivation behind it has been to identify learners who are at risk of low achievement so that they could be offered timely support based on intervention strategies derived from analysis of learners' data. To predict student success, numerous indicators, from varying data sources, have been examined and reported in the literature, as well as various predictive algorithms. Chapter 7 introduces the reader to predictive modeling, through a review of the main objectives, indicators, and algorithms that have been operationalized in previous works as well as a step-by-step tutorial on how to perform predictive modeling in learning analytics using R. The tutorial demonstrates how to predict student success using learning traces originating from a learning management system, guiding the reader through all the required steps from the data preparation to the evaluation of the built models.

Chapter 8: Dissimilarity-Based Clustering Educational Data Using R [14]
Keefe Murphy, Sonsoles López-Pernas, Mohammed Saqr

Chapter 8 presents another central method in learning analytics research: clustering. Clustering is a collective term that refers to techniques aimed at uncovering patterns and subgroups within the data. Finding patterns or differences among students enables teachers and researchers to improve their understanding of the diversity of students—and their learning processes—and tailor their support to different needs. This chapter introduces the theory underpinning the dissimilarity-based clustering methods. Then, the focus is placed on some of the most widely-used heuristic dissimilarity-based clustering algorithms; namely, K-means, K-medoids, and agglomerative hierarchical clustering. Methods for choosing the optimal number of clusters are provided, particularly the criteria that can guide the choice of clustering solution among multiple competing methodologies, and not only the choice of the number of clusters K for a given method. All of these are demonstrated in detail with a tutorial in R using a real-life educational dataset.

Chapter 9: An Introduction and R Tutorial to Model-Based Clustering in Education via Latent Profile Analysis [15]
Luca Scrucca, Mohammed Saqr, Sonsoles López-Pernas, Keefe Murphy

Chapter 9 presents an alternative approach for capturing different patterns or subgroups within students' behavior or functioning. Assuming that there is an

average pattern that represents the entirety of student populations requires the measured construct to have the same causal mechanism, the same development pattern, and affect students in exactly the same way. Using a person-centered method (Finite Gaussian mixture model or latent profile analysis), this chapter offers an introduction to model-based clustering that includes the principles of the methods, a guide to the choice of number of clusters, an evaluation of clustering results and a detailed guide with code and a real-life dataset. The tutorial part shows how to uncover the heterogeneity within engagement data by identifying latent or unobserved clusters. The discussion elaborates on the interpretation of the results, the advantages of model-based clustering as well as how this method compares with others.

2.3 Temporal Methods

We continue our journey with an introduction to temporal methods in learning analytics. Unlike the methods based on mere counts of events or activities, temporal methods acknowledge the order and temporality of events, as well as the transitions thereof, which are key aspects of learning. Temporal methods have garnered increasing attention since they allow researchers to take advantage of the trace log data that students leave behind when using educational technology and also to study longitudinal processes (e.g., a whole study program). Such methods originate in social sciences and have been imported and adapted into the learning analytics field. We provide three chapters focused on sequence analysis, and two on transition analysis through Markovian modeling and process mining.

Chapter 10: Sequence Analysis in Education: Principles, Technique, and Tutorial with R [16]

Mohammed Saqr, Sonsoles López-Pernas, Satu Helske, Marion Durand, Keefe Murphy, Matthias Studer, Gilbert Ritschard

Sequence analysis is a data mining technique that is increasingly gaining ground in learning analytics. Sequence analysis enables researchers to extract meaningful insights from sequential data, i.e., to summarize the sequential patterns of learning data and classify those patterns into homogeneous groups. Chapter 10 introduces readers to sequence analysis techniques and tools through real-life step-by-step examples of sequential trace log data of students' online activities. Readers are guided on how to visualize the common sequence plots and interpret such visualizations. An essential part of sequence analysis is the discovery of patterns within sequences through clustering techniques. Therefore, this chapter demonstrates the various sequence clustering methods, calculation of cluster indices, and evaluation of clustering results.

Chapter 11: Modeling the Dynamics of Longitudinal Processes in Education. A Tutorial with R for The VaSSTra Method [17]

Sonsoles López-Pernas, Mohammed Saqr

Building upon the knowledge acquired in the previous chapter, Chap. 11 covers VaSSTra, a method for analyzing multiple variables across multiple time points. The idea behind this method is to summarize multiple variables at each time point into a single state using person-based methods. Then, sequence analysis can be used to analyze the sequences of such states for each person, and clustering techniques can be implemented to detect similar trajectories of the evolution of such states. The method is illustrated in a case study about engagement. Several engagement-related variables are derived from students' online activities (frequency of each activity, regularity, etc.). These variables are used for clustering students into three states (active, moderate, and disengaged) at each course. Then, sequence analysis is used to map the sequence of engagement states across a whole program. Lastly, clustering mechanisms are used to detect distinct trajectories of engagement.

Chapter 12: A Modern Approach to Transition Analysis and Process Mining with Markov Models in Education [18]

Jouni Helske, Satu Helske, Mohammed Saqr, Sonsoles López-Pernas, Keefe Murphy

Chapter 12 presents Markov models, a widely used technique to model temporal processes. Contrary to the deterministic approach seen in the previous sequence analysis chapters, Markovian models are probabilistic models, focusing on the transitions between states instead of studying sequences as a whole. The chapter provides an introduction to Markov models and differentiates between its most common variations: first-order Markov models, hidden Markov models, mixture Markov models, and hidden mixture Markov models. All implementations are illustrated with a step-by-step tutorial using the R package seqHMM using students' longitudinal data. The chapter also provides a complete guide to performing stochastic process mining with Markovian models as well as plotting, comparing, and clustering different process models

Chapter 13: Multichannel Sequence Analysis in Educational Research Using R [19]

Sonsoles López-Pernas, Satu Helske, Mohammed Saqr, Keefe Murphy

When dealing with learners' data, sometimes one single source of information is not enough to capture all of the dimensions of the learning process. Fortunately, sequence analysis as a method supports the examination of multiple sequences (termed channels) at the same time as long as they follow the same time scheme. Chapter 13 covers multi-channel sequence analysis, allowing the reader to study and visualize synchronized sequences together, and cluster them into distinct trajectories based on the values of the various channels. We present two methods for clustering: one distance-based (see Chap. 10) and one based on Markovian models (see Chap. 12). We illustrate the method by studying the longitudinal association between student engagement and achievement across a study program.

Chapter 14: The Why, the How, and the When of Educational Process Mining in R [20]

Sonsoles López-Pernas, Mohammed Saqr

Process mining is a recent analytical method that enables the extraction of meaningful insights from time-ordered event logs. The goal of process mining is to

discover processes from the data, evaluate process efficiency, and help or enhance processes. Since its introduction in education, process mining has been used to map students' learning processes, visualize learners' strategies, as well as demonstrate differences in approach to learning across different learning groups. Chapter 14 illustrates how to prepare learners' data for process mining and how to visualize the process data using the `bupaverse` framework. Moreover, readers will learn how to examine the transitions between phases or activities within a learning process.

2.4 Network Analysis

The next section of the book deals with the relational aspects of analyzing educational data such as relationships between students, teachers, and topics. Network analysis is the underlying method used to study such relational aspects. Social network analysis allows researchers to study collaboration and discussion between peers and understand the role each student occupies in the network. Moreover, community finding allows the detection of distinct groups of peers in the network that interact with each other more than with the rest. We can even combine network analysis with the temporal methods presented in the last section through temporal networks or use epistemic or ordered network analysis to explore topic or construct co-occurrence.

Chapter 15: Social Network Analysis: A Primer, a Guide and a Tutorial in R [21]

Mohammed Saqr, Sonsoles López-Pernas, Miguel Ángel Conde, Ángel Hernández-García

For five decades, learning networks have been used to map collaboration networks among students, study the influence of peers, and capture the relational dimension of collaborative learning. Additionally, networks have been used to study the semantics of discourse, relations between behaviors, and patterns of relations among teachers. Networks offer a powerful framework with vast potential for data analysis. Chapter 15 introduces the concept and methods of social network analysis and a detailed guide on how researchers can use network analysis using real-world data. The chapter demonstrates network analysis and visualisation with an emphasis on learners' roles and relevance to the educational context. The chapter further provides a mathematical analysis and interpretation of the different social network metrics such as centrality and betweenness measures with several examples of how they can be used in practice.

Chapter 16: Community Detection in Learning Networks Using R [22]

Ángel Hernández-García, Carlos Cuenca-Enrique, Adrienne Traxler, Sonsoles López-Pernas, Miguel Ángel Conde, Mohammed Saqr

In learning situations, communities can be groups of students within a whole cohort who collaborate with each to a larger extent than with other students in a learning situation. Finding these communities is integral to understanding the interaction process, the structure and behaviour of the formed groups and how they

contribute to the overall learning process. Chapter 16 builds on the principles of social networks from Chap. 15 and introduces the topic of community detection. The main aim of community detection is to identify different groups or clusters of nodes within the network that share some similar characteristics. One way of understanding communities in social networks is as subnetworks where the number of internal connections is larger than the number of external connections, and therefore members of a community have a higher probability of being connected to each other than to members of other communities. The chapter focuses on detecting communities (groups of highly connected nodes) within a wider network and shows how to visualize them using R.

Chapter 17: Temporal Network Analysis: Introduction, Methods, and Analysis with R [23]

Mohammed Saqr

Learning can be viewed as relational, interdependent, and temporal and therefore, methods that account for such multifaceted dynamic processes that unfold overtime are required. Chapter 17 combines the temporal and relational aspects in a single analytics framework: temporal networks. Temporal networks allow modeling of the temporal learning processes i.e., the emergence and flow of activities, communities, and social processes through fine-grained dynamic analysis. This can provide insights into phenomena like knowledge co-construction, information flow, and relationship building. This chapter introduces the basic concepts of temporal networks, their types (i.e, contact and interval), and techniques. The chapter further provides a detailed guide to temporal network analysis, which involves network building, visualization, and statistical analysis at the graph and node level.

Chapter 18: Epistemic Network Analysis and Ordered Network Analysis in Learning Analytics [24]

Yuanru Tan, Zachari Swiecki, Andrew Ruis, David Shaffer

The increasing use of technology in many areas of society and life has led to an increasing amount of Big Data about human behavior and interaction. However, this volume of data is usually too large and strains the capabilities of human interpretation and the traditional social science research approaches. Chapter 18 presents two quantitative ethnographic approaches that link the power of statistics and in-depth ethnographic approaches to understand learning behaviour through large-scale qualitative data. Epistemic Network Analysis (ENA) and Ordered Network Analysis (ONA), are two methods for quantifying, visualizing, and interpreting network data. Taking coded data as input, ENA and ONA represent associations between codes (e.g., topics or categories) in undirected or directed weighted network models, respectively. Both techniques measure the strength of association among codes and illustrate the structure of connections in network graphs, quantify changes in the composition and strength of those connections over time, and enable comparison of networks. The chapter presents a thorough description of the methods and a step-by-step guide on how to implement them with R.

2.5 Psychometrics

We finalize the book with a section on psychometrics. In the field of educational psychology, psychometrics aims to study how psychological constructs (e.g., intelligence or aptitude) are related to observable variables (e.g., test scores). Traditionally, psychometric methods in educational psychology have relied on self-reported data from validated questionnaire-like instruments, although nowadays researchers have begun to make use of digital data. We present several techniques to investigate the relationship between measured variables and to test hypotheses and theories: psychological networks, factor analysis, and structural equation modeling (SEM).

Chapter 19: Psychological Networks: A Modern Approach to Analysis of Learning and Complex Learning Processes [25]

Mohammed Saqr, Emorie Beck, Sonsoles López-Pernas

When analyzing psychological phenomena that take place in educational settings, a multitude of variables are at play that may interact with, trigger, and influence each other. To understand such dependency between variables, it is not enough to analyze the linear relationships between each pair of variables, but rather such complexity calls for using more sophisticated methods that capture the full breadth of the interplay between variables: psychological networks. As opposed to social networks where nodes often represent people and edges represent the interactions or relations between them, the nodes in psychological networks represent observed psychological variables and edges represent a statistical relationship between them. Chapter 19 opens the section on psychometric methods by presenting the concept of psychological networks as well as a tutorial for their estimation, visualization, and interpretation with R.

Chapter 20: Factor Analysis in Education Research Using R [26]

Leonie V. D. E. Vogelsmeier, Mohammed Saqr, Sonsoles López-Pernas, Joran Jongerling

Chapter 20 presents factor analysis, a method employed to reduce a large number of variables into fewer numbers of factors. The method is commonly used to identify which observable indicators are representative of latent, not directly-observed constructs. This is a key step in developing valid instruments to assess latent constructs such as student engagement in educational research. The chapter describes the two main approaches for conducting factor analysis in detail and provides a tutorial on how to implement both techniques. The first is confirmatory factor analysis (CFA), a more theory-driven approach, in which a researcher actively specifies the number of underlying constructs as well as the pattern of relations between these dimensions and observed variables. The second is exploratory factor analysis (EFA), a more data-driven approach, in which the number of underlying constructs is inferred from the data, and all underlying constructs are assumed to influence all observed variables (at least to some degree).

Chapter 21: Structural Equation Modeling with R for Education Scientists [27]

Joran Jongerling, Sonsoles López-Pernas, Mohammed Saqr, Leonie V. D .E. Vogelsmeier

Chapter 21 presents the last method in our book: Structural Equation Modeling (SEM). SEM is a suitable and useful method for modeling the multitude of relationships between latent variables and the observable indicators, as well as the relationship between the latent variables themselves to test theories. In its most common form, SEM combines CFA (covered in Chap. 20) with another method named path analysis. Just like CFA, SEM relates observed variables to latent variables that are measured by those observed variables and, as path analysis does, SEM allows for a wide range of regression-type relations between sets of variables (both latent and observed). This chapter presents an introduction to SEM, an integrated strategy for conducting SEM analysis that is well-suited for educational sciences, and a tutorial on how to carry out an SEM analysis in R.

3 The Companion Code and Data

To enhance your learning experience and practical understanding of the concepts discussed in this book, we have developed a companion code repository that accompanies each chapter. The repository contains all the code included in the step-by-step tutorials that illustrate how to implement the different learning analytics methods covered in the book chapters. The code also contains custom functions that automate complex operations, making it easier for anyone to apply the techniques to their own datasets. Moreover, the code will guide the reader on how to generate visualizations, graphs, and plots aimed at helping to interpret and communicate findings effectively. The companion code repository can be accessed at:

https://github.com/lamethods/code

Along with the code, we provide a collection of datasets carefully curated to represent educational scenarios, allowing readers to experiment with the techniques discussed in the book and beyond. Each dataset is described in detail in Chap. 2.

https://github.com/lamethods/data

The combination of theoretical knowledge and practical application through the companion code and datasets will prepare the reader to begin their journey into the multidisciplinary field of learning analytics.

References

1. Conole G, Gašević D, Long P, Siemens G (2011) Message from the LAK 2011 general & program chairs. In: Conole G, Gašević D (eds) Proceedings of the 1st international conference on learning analytics and knowledge. Association for Computing Machinery (ACM), Banff
2. Cornog J, Stoddard GD (1925) Predicting performance in chemistry. J Chem Educ 2:701. https://doi.org/10.1021/ed002p701

3. Platts CV, Wyant G (1969) Network analysis and the possibility of its use in education. Educ Rev 21:109–119. https://doi.org/10.1080/0013191690210203
4. Pechenizkiy M, Trcka N, Vasilyeva E, Van der Aalst W, De Bra P (2009) Process mining online assessment data. ERIC Clearinghouse
5. Soller AL, Lesgold A (2003) A computational approach to analyzing online knowledge sharing interaction. In: Proceedings of artificial intelligence in education
6. Wang F-H (2002) On analysis and modeling of student browsing behavior in web-based asynchronous learning environments. In: Advances in web-based learning. Springer, Berlin, pp 69–80
7. Adam K, Bakar NAA, Fakhreldin MAI, Majid MA (2018) Big data and learning analytics: a big potential to improve e-learning. Adv Sci Lett 24:7838–7843. https://doi.org/10.1166/asl.2018.13028
8. López-Pernas S, Saqr M, Conde J, Del-Río-Carazo L (2024) A broad collection of datasets for educational research training and application. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
9. Tikka S, Kopra J, Heinäniemi M, López-Pernas S, Saqr M (2024) Getting started with r for education research. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
10. Kopra J, Tikka S, Heinäniemi M, López-Pernas S, Saqr M (2024) An r approach to data cleaning and wrangling for education research. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
11. Tikka S, Kopra J, Heinäniemi M, López-Pernas S, Saqr M (2024) Introductory statistics with r for educational researchers. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
12. López-Pernas S, Misiejuk K, Tikka S, Saqr M, Kopra J, Heinäniemi M (2024) Visualizing and reporting educational data with r. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
13. Jovanovic J, López-Pernas S, Saqr M (2024) Predictive modelling in learning analytics using R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
14. Murphy K, López-Pernas S, Saqr M (2024) Dissimilarity-based cluster analysis of educational data: a comparative tutorial using R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
15. Scrucca L, Saqr M, López-Pernas S, Murphy K (2024) An introduction and r tutorial to model-based clustering in education via latent profile analysis. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
16. Saqr M, López-Pernas S, Helske S, Durand M, Murphy K, Studer M, Ritschard G (2024) Sequence analysis in education: principles, technique, and tutorial with r. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
17. López-Pernas S, Saqr M (2024) Modeling the dynamics of longitudinal processes in education. A tutorial with r for the VaSSTra method. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
18. Helske J, Helske S, Saqr M, López-Pernas S, Murphy K (2024) A modern approach to transition analysis and process mining with Markov models in education. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
19. López-Pernas S, Saqr M, Helske S, Murphy K (2024) Multichannel sequence analysis in educational research using R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
20. López-Pernas S, Saqr M (2024) The why, the how, and the when of educational process mining in R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024

21. Saqr M, López-Pernas S, Conde MÁ, Hernández-García Á (2024) Social network analysis: a primer, a guide and a tutorial in r. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
22. Hernández-García Á, Cuenca-Enrique C, Traxler A, López-Pernas S, Conde MÁ, Saqr M (2024) Community detection in learning networks using R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
23. Saqr M (2024) Temporal network analysis: introduction, methods, and analysis with r. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
24. Tan Y, Swiecki Z, Ruis A, Shaffer D (2024) Epistemic network analysis and ordered network analysis in learning analytics. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
25. Saqr M, Beck E, López-Pernas S (2024) Psychological networks: a modern approach to analysis of learning and complex learning processes. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
26. Vogelsmeier LVDE, Saqr M, López-Pernas S, Jongerling J (2024) Factor analysis in education research using R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024
27. Jongerling J, López-Pernas S, Saqr M, Vogelsmeier LV (2024) Structural equation modeling with r for education scientists. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, this volume, 2024

Part I
Getting Started

A Broad Collection of Datasets for Educational Research Training and Application

Sonsoles López-Pernas, Mohammed Saqr, Javier Conde, and Laura Del-Río-Carazo

1 Introduction

Learning analytics involves the combination of different types of data such as behavioral data, contextual data, performance data, and self-reported data to gain a comprehensive understanding of the learning process [1, 2]. Each type of data provides a unique perspective of the learning process and, when analyzed together, can provide a more complete picture of the learner and the learning environment. Throughout the book, we will work with different types of learning analytics data to illustrate the analysis methods covered. This chapter explores the most common types of data that are used in learning analytics and that we will work with in the subsequent book chapters. Such data include demographic and other contextual data about students, performance data, online activity, interactions with other students and teachers, and self-reported data.

This chapter also describes a set of datasets that will be used throughout the book, as well as additional datasets that may be useful for readers to put the newly learned methods into practice. We will discuss the characteristics, structure, and contents of each dataset, as well as the context in which they have been used within the book. The goal of this chapter is to give readers a solid foundation for working with the datasets used in the book, as well as to provide a starting point for those interested in exploring additional data sources.

S. López-Pernas (✉) · M. Saqr
School of Computing, University of Eastern Finland, Joensuu, Finland
e-mail: sonsoles.lopez@uef.fi

J. Conde · L. Del-Río-Carazo
Universidad Politécnica de Madrid, Madrid, Spain

M. Saqr, S. López-Pernas (eds.), *Learning Analytics Methods and Tutorials*,
https://doi.org/10.1007/978-3-031-54464-4_2

2 Types of Data

2.1 Contextual Data

Contextual data refer to data that provide information about the environment in which learning takes place, such as demographic information, socioeconomic status, and prior academic achievement. This type of data can be used to understand how external factors may impact learning, to identify students with similar profiles, and to develop targeted interventions. Demographic data can be used to understand the characteristics of the learners, such as age, gender, race, and ethnicity [3]. Socioeconomic data can be used to examine the impact of the socio-economic status of learners, such as income, employment status, and education level [4]. Prior academic achievement data can be used to understand how the academic background of the learners, such as their previous grades and test scores, may influence their learning at present [5]. The data about the learning context is also relevant to better understand and support students; for example, the level and difficulty of the educational program and courses, format (online vs. face-to-face), or pedagogical approach (e.g., flipped classroom, laboratory course, etc.) [6].

Descriptive statistics can be used to summarize and describe the main characteristics of the contextual data, such as the mean, median, and standard deviation. In Chapters 3 [7] and 4 [8], we will learn how to clean and manipulate data and how to summarize it using descriptive statistics. In Chapter 6 [9], we will learn how to create different types of plots that will allow us to better understand the data. Cluster analysis can also be used to group similar students together. This can be used to identify patterns in the data and, for example, to understand which different groups of students exist in a course or degree and whether such groups differ in terms of, e.g., performance [10]. We cover clustering in Chapters 8 and 9 [11, 12].

It is important to bear in mind that contextual data are essential to understand learners' and the learning process, but they should be used in combination with other types of data to obtain a comprehensive understanding [13]. It is also crucial to comply with data protection laws and regulations and to consider the ethical implications of collecting and operationalizing this type of data, especially when it comes to the existence of bias when making decisions based on demographic data [14].

2.2 Self-reported Data

Self-reported data refers to data provided by students themselves (or other relevant stakeholders), such as data collected through surveys or questionnaires. This type of data can provide valuable insight into learners' and teachers' attitudes, motivation, and perspectives on their learning experiences, and can be used to inform the design of educational programs [15]. It is important to keep in mind that the data should

be cleaned and pre-processed before applying any analytical techniques, especially when dealing with qualitative data (e.g., free text, video, or recordings), and the results should be interpreted with caution, keeping in mind the limitations of self-reported data [16].

Regarding the techniques employed to analyze self-reported data, descriptive statistics and data visualization are commonly used to understand the distribution of responses and to identify patterns in the data (see Chapters 4 [8] and 6 [9]). Moreover, inferential statistics can be used to make inferences about a population based on a sample of data. This can include techniques such as t-tests and analysis of variance to identify significant differences between groups or chi-squared tests to identify associations in the data. Chapter 5 will help us better understand the most common statistical tests and how to implement them with R [17]. Depending on the research question, the type of data, and the level of detail required, a more sophisticated choice of analytical techniques might be needed. For instance, Factor Analysis is a statistical technique that can be used to identify underlying factors or dimensions that explain the relationships between multiple variables [18]. We will learn about it in Chapter 20 [19]. Similarly, Structural Equation Modeling (SEM) can be used to test complex models that involve multiple observed and latent variables that depend on one another. We cover this method in Chapter 21 [20]. Moreover, self-reported data can be analyzed using psychological networks, a relatively new approach in the field of psychology that focuses on understanding psychological phenomena as interconnected networks of individual components. We cover this method in Chapter 19 [21]. Lastly, text mining can be used to analyze unstructured data, such as open-ended responses to surveys or interviews. It can be used to identify key concepts and themes, perform sentiment analysis, and summarize text [22]. This type of analysis is beyond the scope of this book.

2.3 Activity Data

Activity data in learning analytics refers to the data that is collected about a student's interactions with educational technology. Activity data can include information such as the learning resources a student visits, the time spent on a resource, the buttons clicked, and the messages posted [23]. Data can be collected automatically by the learning management system (LMS) or other educational technology (e.g., a game, an intelligent tutoring system, eBooks, or coding environments). Log activity data can be used to track student progress, identify areas where students may be struggling, and personalize instruction [24]. For example, if a student is spending an excessive amount of time on a particular concept, it may indicate that they are having difficulty understanding that concept. In this case, the teacher can provide additional support or re-teach the concept to help the student improve. Log activity data can also be used to measure student engagement with the course content and to identify students who are not engaging with the material [25]. Log activity data

have been used to detect students' online tactics and strategies [26] paying attention not only to the frequency but to the order and timing of students' events.

Besides basic analysis using descriptive and inferential statistics, activity logs have been operationalized in many ways in learning analytics, especially using temporal methods that allow to take advantage of the availability of large amounts of timestamped data. For example, process mining—which we cover in Chapter 14 [27]—has been used to investigate how students navigate between different online activities [28]. Sequence analysis has been used to detect and interpret students' online tactics and strategies based on the order of learning activities within learning sessions [29]. We dedicate several chapters to this technique [30–33]. Such analyses have been complemented with cluster analysis, which allows to detect distinct patterns of students with different online behavior [34] (see Chapters 8 and 9 [11, 12]).

2.4 Social Interaction Data

Social interaction data in learning analytics refers to the data collected about students' interactions with each other (and sometimes teachers too) in a learning environment, social media, or messaging platforms. This can include data such as the frequency and nature of interactions, the content of discussions, and the participation of students in group work or collaborative activities. Social interaction data can be used to understand how students are engaging with each other and to identify patterns or roles that students assume [35]. For example, if a student is not participating in group discussions, it may indicate that they are feeling disengaged or are having difficulty understanding the material. Furthermore, social interaction data can be used to study how students' depth of contributions to the discussion influences performance [36]. For example, an analysis of social interaction data may reveal that students who receive more replies from other students perform better in the course than students whose contributions do not spark a lot of interest.

Social Network Analysis (SNA) is the most common method to study social interaction data. SNA comprises a wealth of quantitative metrics that summarize relationships in a network. In most cases in learning analytics, this network is formed based on students' interactions. These metrics, named centrality measures, pave the path to apply other analytical methods such as cluster analysis to detect collaboration roles [37], or predictive analytics to determine whether performance can be predicted from students' centrality measures [38]. We cover the basics of SNA in Chapter 15 of the book [39], community detection in Chapter 16 [40], and temporal network analysis in Chapter 17 [41]. Moreover, the nature and content of students' interactions can be analyzed with Epistemic Network Analysis (ENA), a method for detecting and quantifying connections between elements in coded data and representing them in dynamic network models [42]. We cover this method in Chapter 18 [43].

2.5 Performance Data

Performance data refers to data that measures how well learners are able to apply what they have learned. This type of data can be used to evaluate the effectiveness of a particular educational activity, to identify areas where additional support may be needed, or to detect students at risk. Performance includes assessment data from tests, quizzes, projects, essays, exams, and other forms of evaluation used to track students' progress. Assessment can be performed by different entities, such as teachers, peers or automated assessment tools. Moreover, assessment data can have different levels of granularity: it can be the grade for a specific task, a midterm or final exam, or a project; it can be the final grade for a course, or even a whole program GPA [44]. Performance data used for learning analytics may not necessarily be assessment data. For instance, pre-test and post-test data are used to evaluate the effectiveness of a particular educational intervention [45]. Another example is the data captured by audience response systems (ARSs) [46], which are often used to evaluate learners' knowledge retention during lectures.

Performance data are rarely analyzed on its own, but rather used in combination with other sources of data. For example, a common use case in learning analytics is to correlate or predict grades with indicators from several sources [13, 47], such as demographic data, activity data or interaction data. In the book, we cover predictive modelling in Chapter 7 [48]. Moreover, grades are often compared among groups or clusters of students, for example, to evaluate the performance of students that use different online learning strategies [29] or to establish whether students' assuming different levels of collaboration also show differences in performance [49]. Clustering is covered in Chapters 8 and 9 [11, 12].

2.6 Other Types of Data

In recent years, the landscape of data used for learning analytics has undergone a remarkable expansion beyond demographics, grades, surveys and digital logs [50]. This evolution has led to the incorporation of novel methodologies designed to capture a more holistic understanding of students' learning experiences, including their physiological responses [51]. This progression encompasses a diverse range of data acquisition techniques, such as eye-tracking data that traces the gaze patterns of students, electrodermal activity which measures skin conductance and emotional arousal, EEG (electroencephalogram) recordings that capture brain activity patterns, heartbeat analysis reflecting physiological responses to learning stimuli, and motion detection capturing physical movements during learning activities [52]. These physiological datasets are often combined with other forms of information, such as video recordings (e.g., [50]). Through the combination of various data modalities, researchers and educators can better understand how students engage with educational content and respond to different teaching methodologies [51]. This

analysis goes beyond the capabilities of conventional digital learning tools, offering insights into the emotional, cognitive, and physical aspects of learning that might otherwise remain concealed [53, 54]. This synergistic analysis of multiple data sources is often referred to as "multimodal learning analytics". In Chapter 13, we will cover multi-channel sequence analysis, a method suitable for analyzing several modalities of data at the same time [33].

3 Dataset Selection

The present section describes a set of curated datasets that will be used throughout the book. In the introductory chapters, the reader will learn how to import datasets in different formats [7], clean and transform data [8], conduct basic statistics [17], and create visualizations [9]. Each of the remaining chapters covers a specific method, which is illustrated in a tutorial-like way using one or more of the datasets described below. All the datasets are available on Github.[1]

3.1 LMS Data from a Blended Course on Learning Analytics

Link to the dataset

The first dataset in our book is a synthetic dataset generated from a real blended course on Learning Analytics offered at the University of Eastern Finland. The course has been previously described in a published article [55] which used the original (non-synthetic) dataset. The lectures in the course provided the bases for understanding the field of learning analytics: the recent advances in the literature, the types of data collected, the methods used, etc. Moreover, the course covered learning theories as well as ethical and privacy concerns related to collecting and using learners' data. The course had multiple practical sessions which allowed students to become skilled in learning analytics methods such as process mining and social network analysis using real-life datasets and point-and-click software.

Students in the course were required to submit multiple assignments; most of them were practical, in which they had to apply the methods learned in the course, but others were focused on discussing learning theories, ethics, and even conducting a small review of the literature. The course had a final project that accounted for 30% of the course final grade in which students had to analyze several datasets in multiple ways and comment and discuss their findings. Moreover, there was a group project in which students had to present an implementation of learning analytics application in an institutional setting, discussing the sources of data collection, the analyses that could be conducted, and how to present and make use of the data and analyses. The

[1] Github repository: https://github.com/lamethods/data.

course was implemented in a blended format: instruction was face-to-face while the learning materials and assignments were available online, in the Moodle LMS. Discussions among students in the group project also took place online in the LMS forum.

The dataset contains four files: a file containing students' online activities in Moodle, a file containing their grades, a file containing their demographic data, and a file that aggregates all the information. It is shared with a CC BY 4.0 license, which means that anyone is free to share, adapt, and distribute the data as long as appropriate credit is given. The dataset has been used in the introductory chapters of the book to learn the basics of R [7], data cleaning [8], basic statistics [17] and data visualization [9]. Moreover, it has been used in two additional chapters to illustrate to well-known learning analytics methods: sequence analysis [30] and process mining [27]. Below, we provide further details on each of the files of the dataset.

3.1.1 Events

The `Events.xlsx` file contains 95,580 timestamped Moodle logs for 130 distinct students. The activities include viewing the lectures, discussing on forums, and working on individual assignments, as well as discussion in small groups, among other events. The logs were re-coded to balance granularity with meaningfulness, i.e., grouping together logs that essentially represent the same action. For example, the activities related to the group project were all coded as `Group_work`, log activities related to feedback were coded as `Feedback`, logs of students' access to practical resources or assignments were coded as `Practicals`, social interactions that are unrelated to learning were coded as `Social`, etc. Below we describe the columns of the dataset. A preview of the dataset can be seen in Table 1. In Fig. 1, we show the distribution of events per student.

- **Event.context**: Resource of the LMS where the event takes place, for example "Assignment: Literature review".
- **user**: User name in the LMS.
- **timecreated**: Timestamp in which each event took place, ranging from September 9th 2019 to October 27th 2019.
- **Component**: Type of resource involved in the event. There are 13 distinct entries, such as Forum (39.11%); System (34.33%); Assignment (15.50%) and 10 others.
- **Event.name**: Name of the event in Moodle. There are 27 distinct entries, such as Course module viewed (35.89%); Course viewed (26.28%); Discussion viewed (13.77%) and 24 others.
- **Action**: Column coded based on the combination of the event name and context. There are 12 distinct entries, such as Group_work (34.25%); Course_view (26.45%); Practicals (10.48%) and 9 others.

Table 1 Preview of the LA course Moodle Events dataset

	Event.context	User	Timecreated	Component	Event.name	Action
1	Assignment: Final Project	9d744e5bf	1572082632	Assignment	Course module viewed	Assignment
2	Assignment: Final Project	91489f7a9	1572080974	Assignment	The status of the submission has been viewed.	Assignment
3	Assignment: Final Project	278a75edf	1571400328	Assignment	Course module viewed	Assignment
4	Assignment: Final Project	53d6ab60c	1571491717	Assignment	The status of the submission has been viewed.	Assignment
5	Assignment: Final Project	aab7ad69f	1571182693	Assignment	Course module viewed	Assignment
6..95625						
95626	Zoom meeting: Group presentation 8 AM	215b886a6	1571119806	Zoom meeting	Clicked join meeting button	Group_work

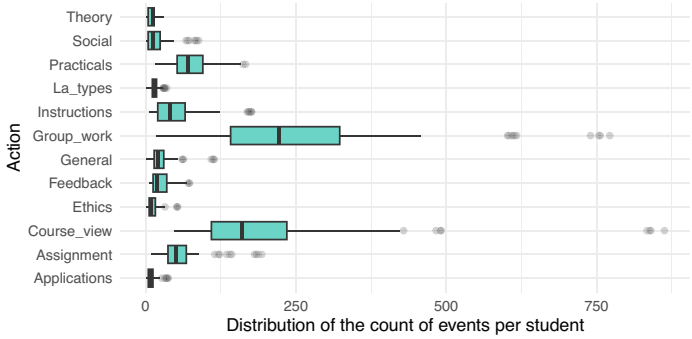


Fig. 1 Number of actions per student per type

Table 2 Preview of the LA course Demographic dataset

	User	Name	Surname	Origin	Gender	Birthdate	Location	Employment
1	6eba3ff82	Amanda	Mora	Costa Rica	F	28.2.1998	On campus	None
2	05b604102	Lian	Abdullah	Yemen	F	19.11.1996	On campus	None
3	111422ee7	Bekim	Krasniqi	Kosovo	M	30.1.1999	On campus	None
4	b4658c3a9	Yusuf	Kaya	Turkey	M	16.6.1998	On campus	None
5	e6ec47f29	Zoran	Babić	Serbia	M	29.10.1998	On campus	Part-time
6..129								
130	278a75edf	Marwa	Singh	Qatar	F	29.3.1996	Remote	None

3.1.2 Demographics

The `Demographics.xlsx` file contains simulated demographic data on the students, including name, date of birth, gender, location (on-campus vs. remote student), and employment status. Table 2 shows a preview of the data.

- **user**: User identifier in the learning management system, with 130 distinct entries.
- **Name**: User's first name.
- **Surname**: User's last name.
- **Origin**: Country of origin.
- **Gender**: User's gender: F (Female, 50%) or M (Male, 50%).
- **Birthdate**: Date of birth.
- **Location**: Whether the student is on campus or studying remotely, with 2 distinct entries, such as On campus (81.54%); Remote (18.46%).
- **Employment**: Whether the student is working part-time, full-time or not at all, with 3 distinct entries, such as None (70.77%); Part-time (25.38%); Full-time (3.85%).

3.1.3 Results

Performance data is provided in the `Results.xlsx` file, including grades per assignment and the overall course grade. Table 3 shows a preview of the data (the column names have been abbreviated in the preview). Figure 2 shows the distribution of grades per graded item.

- **user**: User name in the learning management system (it matches the previous files).
- **Grade.SNA_1**: Grade of the first SNA assignment (0–10).
- **Grade.SNA_2**: Grade of the second SNA assignment (0–10).
- **Grade.Review**: Grade of the studies' review assignment (0–10).
- **Grade.Group_self**: Individual grade of the group project (0–10).
- **Grade.Group_All**: Group grade of the group project (0–10).
- **Grade.Excercises**: Grade of the practical exercises (0–10).
- **Grade.Project**: Final project grade (0–10).
- **Grade.Literature**: Grade of the literature review assignment (0–10).
- **Grade.Data**: Grade of the data analysis assignment (0–5).
- **Grade.Introduction**: Grade of the introductory assignment (0–10).
- **Grade.Theory**: Grade of the theory assignment (0–10).
- **Grade.Ethics**: Grade of the ethics assignment (0–10).
- **Grade.Critique**: Grade of the critique assignment (0–10).
- **Final_grade**: Final course grade (0–10).

3.1.4 AllCombined

This file `AllCombined.xlsx` contains the students' demographics, grades, and frequency of each action in the LMS. The columns from students' demographic data and grades remain the same, while the event data has been grouped per student for each type of event (`Action` column in the `Events` dataset), i.e., there is a new column for each type of event that contains the number of events of that type that each student had. Moreover, a new column `AchievingGroup` separates the high achievers from the low achievers. Table 4 shows a preview of the new columns of the data (the column names have been abbreviated) that were not shown in the previous previews (Fig. 3).

- **Frequency.Applications**: Number of events related to the "Applications" resource.
- **Frequency.Assignment**: Number of events related to the assignments' submission.
- **Frequency.Course_view**: Number of visits to the course main page.
- **Frequency.Feedback**: Number of views of the assignment feedback.
- **Frequency.General**: Number of events related to general learning resources.
- **Frequency.Group_work**: Number of events related to the group work.

Table 3 Preview of the LA course results (grades) dataset

	User	SNA_1	SNA_2	Review	Group_self	Group_All	Excercises	Project	Literature	Data	Introduction	Theor	Ethics	Critique	Final_grade
1	6eba3ff82	0	0	6.67	5	4.00	10	0.00	6.67	4	6	2	2	4	2.626970
2	05b604102	8	10	6.67	1	3.00	10	7.00	6.67	3	6	2	8	4	4.670169
3	111422ee7	10	10	10.00	10	9.11	10	9.33	10.00	5	10	10	10	10	9.244600
4	b4658c3a9	5	5	0.00	1	4.00	10	6.00	4.33	3	4	2	2	6	0.000000
5	e6ec47f29	10	10	10.00	10	9.18	10	5.33	10.00	5	10	10	10	10	8.238179
6..129															
130	278a75edf	9	10	7.33	10	8.56	10	5.33	7.33	4	6	2	6	6	7.026270

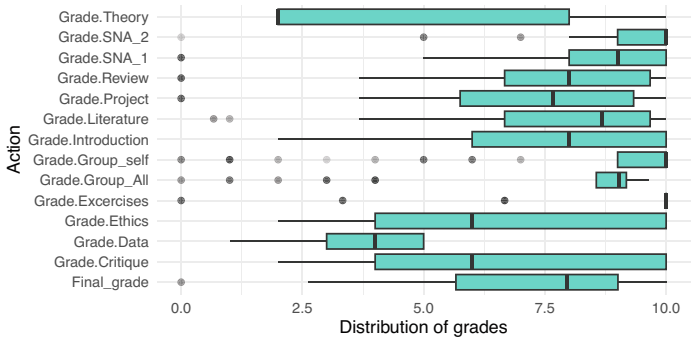


Fig. 2 Grade per student and graded item

- **Frequency.Instructions**: Number of events related to assignment instructions.
- **Frequency.La_types**: Number of events related to the "LA types" resource.
- **Frequency.Practicals**: Number of events related to the practicals.
- **Frequency.Social**: Number of events related to the forum discussion.
- **Frequency.Ethics**: Number of events related to the "Ethics" resource.
- **Frequency.Theory**: Number of events related to the "Theory" resource.
- **Frequency.Total**: Number of events overall.
- **AchievingGroup**: Categorization as high achievers (top 50% grades) and lows achievers (bottom 50% grades).

3.2 LMS Data from a Higher Education Institution in Oman

Link to the dataset

The next dataset includes the log data of students enrolled in a computing specialization in a higher education institution in Oman. The dataset contains data from students enrolled in the sixth semester or beyond. The data was recorded across five modules (Spring 2017–2021) and includes the records of 326 students with 40 features in total, including the students' academic information (24 features), logs of students' activities performed on the Moodle LMS (10 features), and students' video interactions on the eDify mobile application (6 features). The academic data includes demographic data, academic data, study plan, and academic violations. The Moodle activity data includes students' timestamped activities in Moodle. The eDify data contains video interactions in the eDify mobile application.

The dataset has been described in an article by Hasan et al. [56] and was originally made available in the Zenodo repository [57] with a CC BY 4.0 license, which means anyone can share and adapt the dataset but must give credit to the author and cannot apply any further restrictions to the dataset. Besides the raw data,

Table 4 Preview of the LA course combined dataset

	F.Applications	F.Assignment	F.Course_view	F.Feedback	F.General	F.Group_work	F.Instructions	F.La_types	F.Practicals	F.Social	F.Ethics	F.Theory	F.Total	AchievingGroup
1	2	121	103	7	10	55	17	1	89	12	0	0	417	Low achiever
2	0	62	294	22	19	323	68	12	69	32	14	3	918	Low achiever
3	0	41	53	0	11	19	6	7	47	0	0	15	199	Low achiever
4	0	44	49	0	10	19	7	8	48	0	0	14	199	Low achiever
5	0	9	119	9	9	218	7	1	61	0	0	3	436	Low achiever
6..129														
130	7	84	491	56	49	457	176	25	145	67	32	28	1617	High achiever

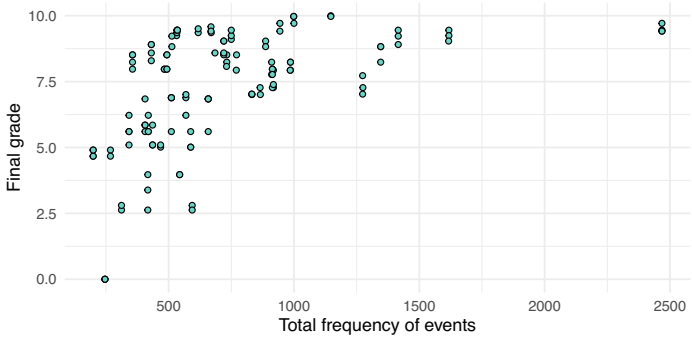


Fig. 3 Relationship between total frequency of events and final grade

the dataset includes a processed file that contains a series of indicators that can be used for different purposes such as predictive modeling or clustering of students' profiles. It has been used in several publications with the purpose of predicting student performance using different algorithms and approaches [58, 59]. The main files of the dataset are described below.

3.2.1 Student Academic Information

Students' academic information downloaded from the institutional information system. The data are spread in 15 files (starting with `KMS`), but have been combined in a single file (`KMSmodule.RDS`) for convenience. Table 5 shows a preview of the data. The resulting dataset has the following structure:

- **file**: Original file name that contains the module offering identifier (e.g., "KMS Module 1 F19.csv"). There are 15 distinct entries (one for each file).
- **ModuleCode**: Module identifier (Module 1–5).
- **ModuleTitle**: Name of the module (Course 1–5).
- **SessionName**: Class section in which the student has been enrolled (Session-A or Session-B).
- **RollNumber**: Student identifier (e.g., "Student 83"). There are 306 distinct students.
- **CGPA**: Students' cumulative GPA (1–4).
- **AttemptCount**: Number of attempts per student and module.
- **Advisor**: Students' advisor identifier (e.g., 'Advisor 16'). There are 50 distinct advisors.
- **RemoteStudent**: Whether the student is studying remotely. There are 2 possible values: Yes (0.31%) or No (99.69%).
- **Probation**: Whether the student has previous incomplete modules. There are 2 possible values: Yes (3.36%) or No (96.64%).

Table 5 Preview of the Higher Education Institution Academic data

	File	Module-Code	Module-Title	Session-Name	RollNumber	CGPA	Attempt-Count	Advisor	Remote-Student	Probation	High-Risk	Term-Exceeded	AtRisk	AtRisk-SSC	Other-Modules	Prerequisite-Modules	Plagiarism-History	MalPractice-History
1	KMS Module 1 F19.csv	Module 1	Course 1	Session-A	Student 221	2.50	1	Advisor 16	No	No	No	No	No	No	2	No	Module 3	No History
2	KMS Module 1 F19.csv	Module 1	Course 1	Session-A	Student 208	3.25	1	Advisor 25	No	No	No	No	No	No	3	No	No History	No History
3	KMS Module 1 F19.csv	Module 1	Course 1	Session-A	Student 209	2.50	1	Advisor 3	No	No	No	No	No	No	2	No	No History	Module 2
4	KMS Module 1 F19.csv	Module 1	Course 1	Session-A	Student 210	2.50	1	Advisor 8	No	No	No	No	No	No	3	No	No History	No History
5	KMS Module 1 F19.csv	Module 1	Course 1	Session-A	Student 211	2.50	1	Advisor 14	No	No	No	No	No	No	3	No	No History	No History
6..326																		
327	KMS Module 5 SP18.csv	Module 5	Course 5	Session-A	Student 89	2.00	1	Advisor 1	No	No	No	Yes	No	No	2	Yes	Module 20	Module 18

- **HighRisk**: Whether the module has a high risk of failure. There are 2 possible values: Yes (6.73%) or No (93.27%).
- **TermExceeded**: Whether the student is progressing in their degree plan. There are 2 possible values: Yes (1.83%) or No (98.17%).
- **AtRisk**: Whether the student has failed two or more modules in the past. There are 2 possible values: Yes (23.55%) or No (76.45%).
- **AtRiskSSC**: Whether the student has been registered for having any educational deficiencies. There are 2 possible values: Yes (4.59%) or No (95.41%).
- **OtherModules**: Number of other modules the student is enrolled in on the same semester.
- **PrerequisiteModules**: Whether the student has enrolled in the prerequisite module.
- **PlagiarismHistory**: Modules for which the student has a history of plagiarism.
- **MalPracticeHistory**: Modules for which the student has a history of academic malpractice.

3.2.2 Moodle

The Moodle data is spread around 15 files (starting with `Moodle`), which contain all the clicks that students performed in the Moodle LMS in each module. The 15 files have been combined in a single file (`moodleEdify.RDS`) for convenience. Table 6 shows a preview of the data. The resulting dataset has the following structure:

- **csv**: Module offering identifier. There are 15 distinct entries (one for each file).
- **Time**: Timestamp in which the event occurred, ranging between January 1st 2018 to December 9th 2020.
- **Event context**: Resource of the LMS where the event takes place, for example "File: Lecture 4".
- **Component**: Type of resource involved in the event. There are 16 distinct entries, such as System (51.71%); File (22.30%); Turnitin Assignment 2 (15.04%) and 13 others.
- **Event name**: Name of the event in Moodle. There are 70 distinct entries, such as Course viewed (36.59%); Course module viewed (25.98%); List Submissions (11.08%) and 67 others.

3.2.3 Activity

Aggregated information per student based on the Moodle data, including the activity on campus and at home. The data are also spread in 15 files (starting with `Activity`), but has been combined in a single file (`ActivityModule.RDS`) for convenience. Table 7 shows a preview of the data. The resulting dataset has the following structure:

Table 6 Preview of the Higher Education Institution Moodle data

	csv	Time	Event context	Component	Event name
1	Moodle Module 1 F19	17/10/21, 12:39	Course: Object Oriented Programming(Fall 2019 - F19 COMP 0328.1)	Logs	Log report viewed
2	Moodle Module 1 F19	17/10/21, 12:38	Course: Object Oriented Programming(Fall 2019 - F19 COMP 0328.1)	System	Course viewed
3	Moodle Module 1 F19	19/08/20, 12:40	Course: Object Oriented Programming(Fall 2019 - F19 COMP 0328.1)	System	Course viewed
4	Moodle Module 1 F19	6/08/20, 21:00	Course: Object Oriented Programming(Fall 2019 - F19 COMP 0328.1)	System	Course viewed
5	Moodle Module 1 F19	4/08/20, 12:51	File: MIG	File	Course module viewed
6..98696					
98697	Moodle Module 5 SP18	29/02/18, 18:59	Course: Business Technology Management(Spring 2018 - S18 COMP 30009)	System	Course viewed

Table 7 Preview of the Higher Education Institution Activity data

	File	RollNumber	Online C	Online O
1	Activity Module 1 F19.csv	Student 208	35	108
2	Activity Module 1 F19.csv	Student 209	332	256
3	Activity Module 1 F19.csv	Student 210	158	20
4	Activity Module 1 F19.csv	Student 211	345	90
5	Activity Module 1 F19.csv	Student 212	352	459
6..325				
326	Activity Module 5 SP18.csv	Student 89	216	43

Table 8 Preview of the Higher Education Institution grade data

	File	RollNumber	SessionName	CW1	CW2	ESE
1	Result Module 1 F19.csv	Student 208	Session-A	29	34	63
2	Result Module 1 F19.csv	Student 209	Session-A	18	36	54
3	Result Module 1 F19.csv	Student 210	Session-A	16	18	34
4	Result Module 1 F19.csv	Student 211	Session-A	15	28	43
5	Result Module 1 F19.csv	Student 212	Session-A	20	34	54
6..325						
326	Result Module 5 SP18.csv	Student 89	Session-A	79	80	30

- **file**: Original file name that contains the module offering identifier (e.g., "Activity Module 1 F19.csv"). There are 15 distinct entries (one for each file).
- **RollNumber**: Student identifier (e.g., "Student 208").
- **Online C**: Duration of the activity (in minutes) within campus.
- **Online O**: Duration of the activity (in minutes) off campus.

3.2.4 Results

Student performance data in each module. The data are also spread in 10 files (starting with `Result`), but has been combined in a single file (`ResultModule.RDS`) for convenience. Table 8 shows a preview of the data. The resulting dataset has the following structure:

- **file**: Original file name that contains the module offering identifier (e.g., "Result Module 1 F19.csv"). There are 10 distinct entries (one for each file).
- **RollNumber**: Student identifier (e.g., "Student 208"). There are 326 distinct students.
- **SessionName**: Class section, Session-A (84.66%), or Session-B (9.82%).
- **CW1**: Grade of students' first assignment (0–100).
- **CW2**: Grade of students' second assignment (0–100).
- **ESE**: Grade of students' end-of-semester examination (0–100).

Table 9 Preview of the Higher Education Institution eDify data

	File	RollNumber	Played	Paused	Likes	Segment
1	VL Module 1 F19.csv	Student 208	5	0	1	3
2	VL Module 1 F19.csv	Student 209	1	4	3	3
3	VL Module 1 F19.csv	Student 210	5	0	2	5
4	VL Module 1 F19.csv	Student 211	5	0	1	3
5	VL Module 1 F19.csv	Student 212	4	0	2	3
6..325						
326	VL Module 5 SP18.csv	Student 89	5	7	1	4

3.2.5 eDify

Student aggregated activity data in the eDify mobile application. The data are also spread in 15 files (starting with `VL`), but has been combined in a single file (`VLModule.RDS`) for convenience. Table 9 shows a preview of the data. The resulting dataset has the following structure:

- **file**: Original file name that contains the module offering identifier (e.g., "Result Module 1 F19.csv"). There are 15 distinct entries (one for each file).
- **RollNumber**: Student identifier (e.g., "Student 208"). There are 326 distinct students.
- **Played**: Number of times the student has played a video.
- **Paused**: Number of times the student has paused a video.
- **Likes**: Number of times the student has liked a video.
- **Segment**: Number of times a student has scrolled to a specific part of the video.

3.3 School Engagement, Academic Achievement, and Self-regulated Learning

Link to the dataset

This dataset includes measures of school engagement, self-regulation and academic performance of a group of primary school students in northern Spain [60]. The data contains responses to the questionnaire from 717 primary education students. The subjects were recruited using convenience sampling from 15 schools (5 privately funded and 10 publicly funded) in northern Spain. The objective of collecting these data was to characterize school engagement and to explore to which extent different engagement profiles are associated with academic performance and self-regulation. In the questionnaire, engagement was assessed with the school engagement measure [61], which allows to differentiate between behavioral, cognitive and emotional engagement. Self-regulation was assessed using the self-regulation strategy inventory [62], which allows measuring students' approaches

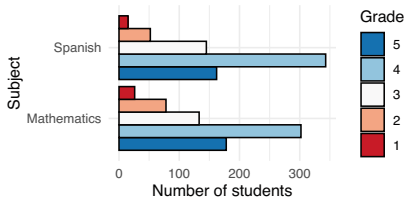


Fig. 4 Self-reported grades of students in Mathematics and Spanish

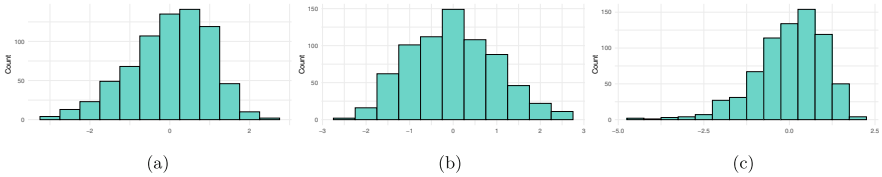


Fig. 5 School engagement results of the survey. (**a**) Emotion engagement z-score. (**b**) Cognitive engagement z-score. (**c**) Behavior engagement z-score

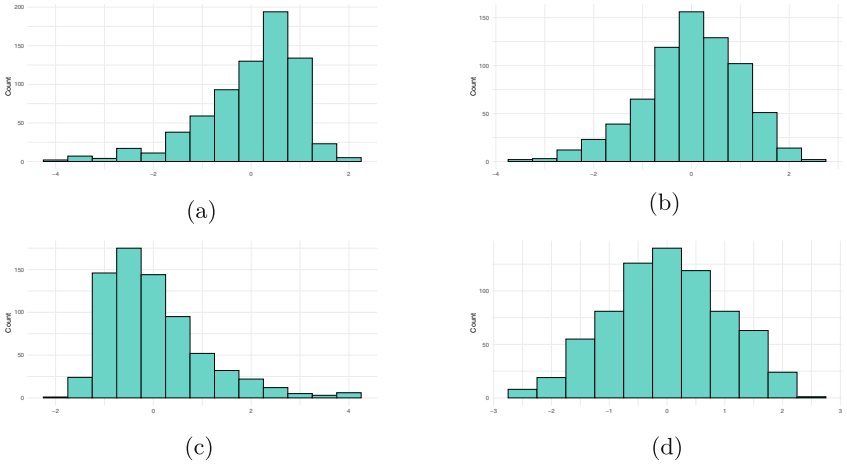


Fig. 6 Self-regulation results of the survey. (**a**) Environment management z-score. (**b**) Information help management z-score. (**c**) Maladaptative behavior z-score. (**d**) Time management z-score

to seeking and learning information, maladaptive regulatory behavior, environment management, and time management. The measure used for achievement was students' self-reported information about their grades in Spanish and mathematics on a scale of 1 (fail) to 5 (outstanding).

Figure 4 summarizes the responses of the students in both subjects, Fig. 5 presents a set of histograms with the engagement measures, and Fig. 6 includes a set of histograms with the self-regulation measures. The dataset was analyzed in a previous article [63], where the authors carried out cluster analysis using LPA (Latent Profile Analysis) to identify different groups of students according to their

engagement and self-regulation and to compare the performance between these groups. The dataset is used in Chapter 9 [11] of this book, which covers model-based clustering. The dataset is published with a CC BY 4.0 license, which means that you can share, copy and modify this dataset so long as appropriate credit is given. Table 10 shows a preview of the dataset. Below, we describe the dataset's main variables.

- **alumno**: Student identifier in the school.
- **sexo**: Student's gender (1 = Male, 48.40%; 2 = Female, 51.46%).
- **coleg**: School identifier (1–15).
- **curso**: Grade that students were in 5th grade (62.48%) or 6th grade (37.52%).
- **grup**: Class section (1–3).
- **ren.mat**: Mathematics self-reported academic achievement (1–5).
- **ren.leng**: Spanish self-reported academic achievement (1–5).
- **Emotion_Engage**: Emotional engagement (z-score).
- **Cognitive_Engage**: Cognitive engagement (z-score).
- **Behavior_Engage**: Behavioural engagement (z-score).
- **Enviroment_Manage**: Environment management (z-score).
- **Information_help_Manage**: Information and help management (z-score).
- **Maladapative_Behavior**: Maladaptative self-regulation (z-score).
- **Time_Manage**: Time management self-regulation (z-score).

3.4 Teacher Burnout Survey Data

Link to the dataset

The next dataset presents the responses collected from a survey about teacher burnout in Indonesia [64]. The survey questionnaire contains 18 items in five different categories. The first category contains five items to assess the teacher self-concept (TSC), from the TSC Evaluation Scale [65]. The second category is teacher efficacy (TE), with 5 items adapted from [66]. The remaining categories are Emotional Exhaustion (EE, 5 items), Depersonalization (DP, 3 items), and Reduced Personal Accomplishment (RPA, 5 items), adapted from the Maslach burnout inventory [67]. The survey items were measured using a 5-point Likert scale, where 1 represents "never", and 5 represents "always".

Table 11 shows a preview of the dataset with the questions and answers, and Fig. 7 represents it in a Likert scale chart. The dataset has been analyzed using several statistical methods [68]: Content Validity Index (CVI), Exploratory Factor Analysis (EFA), Confirmatory Factor Analysis (CFA), Covariance-Based SEM (CB-SEM). The main aim was to determine the factors that may be predictors of teacher burnout. In this book, we also make use of this dataset to illustrate Factor Analysis [19] and SEM [20]. The files associated with this dataset are licensed under a CC BY 4.0 license, which means you can share, copy and modify this dataset so

Table 10 Preview of the School Engagement, Academic Achievement, and SRL data

	alumno	sexo	coleg	curso	grup	ren.mat	ren.leng	Emotion_ Engage	Cognitive_ Engage	Behavior_ Engage	Enviroment_ Manage	Information_ help_Manage	Maladapative_ Behavior
1	1	2	1	6	1	4	4	0.8989251	0.25057461	−0.66817340	−0.1483424	0.2489923	0.08695216
2	2	1	1	6	1	4	3	−2.4712139	1.50596375	0.55572643	0.6627681	0.3436559	−0.79488941
3	3	2	1	6	1	4	4	0.2614537	1.43763063	−0.49791451	0.3090213	0.1281156	−0.13740548
4	4	2	1	6	1	3	3	−0.5421392	−0.19963989	−0.48553522	−0.0706820	−0.7501752	0.31906640
5	5	1	1	6	1	5	3	0.5337485	−0.05256182	−0.04035667	0.1241936	0.2688936	−0.77730497
6..716													
717	25	2	15	6	2	5	5	0.1609796	−0.88981754	1.28726373	0.3358852	1.4474958	0.05243749

Table 11 Preview of the burnout survey data answers

	TSC1	TSC2	TSC3	TSC4	TSC5	TE1	TE2	TE3	TE4	TE5	EE1	EE2	EE3	EE4	EE5	DE1	DE2	DE3	RPA1	RPA2	RPA3	RPA4	RPA5
1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
2	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
3	4	4	5	4	5	5	4	4	5	4	4	5	5	4	5	5	5	5	5	4	5	4	5
4	4	4	5	4	5	5	4	4	5	4	4	5	5	4	5	5	5	5	5	4	5	4	5
5	4	5	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
6..875																							
876	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	4	5	5	5	5	5	5

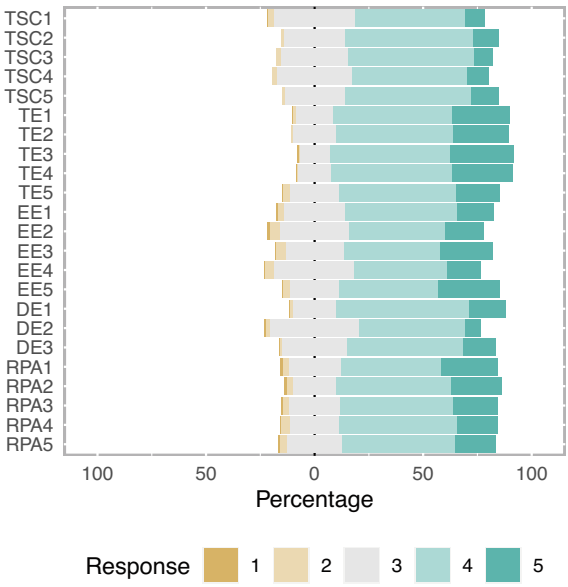


Fig. 7 Results of the burnout survey

long as appropriate credit is given to the authors. The variables of the dataset are described below.

- **Teacher self-concept**
  - **TSC1**: Response to the item "I think I have good teaching skills and ability".
  - **TSC2**: Response to the item "I have a reputation for being an efficient teacher".
  - **TSC3**: Response to the item "My colleagues regard me as a competent teacher".
  - **TSC4**: Response to the item "I feel I am a valuable person".
  - **TSC5**: Response to the item "Generally speaking, I am a good teacher".
- **Teacher efficacy**
  - **TE1**: Response to the item "I help my students value learning".
  - **TE2**: Response to the item "I motivate students who show low interest in schoolwork".
  - **TE3**: Response to the item "I improve understanding of students who are failing".
  - **TE4**: Response to the item "I provide appropriate challenges for very capable students".
  - **TE5**: Response to the item "I get students the students to follow classroom rules".
- **Emotional exhaustion**
  - **EE1**: Response to the item "I feel emotionally drained from my work".

  - **EE2**: Response to the item "I feel used up at the end of the workday".
  - **EE3**: Response to the item "I feel fatigued when I get up in the morning and have to face another day on the job".
  - **EE4**: Response to the item "I feel burnt out from my work".
  - **EE5**: Response to the item "Working with people all day is really a strain on me".
- **Depersonalization**
  - **DE1**: Response to the item "I've become more callous toward people since I took this job".
  - **DE2**: Response to the item "I worry that this job is hardening me emotionally".
  - **DE3**: Response to the item "I feel frustrated by my job".
- **Reduced Personal Accomplishment**
  - **RPA1**: Response to the item "I cannot easily create a relaxed atmosphere with my students".
  - **RPA2**: Response to the item "I do not feel exhilarated after working closely with my students".
  - **RPA3**: Response to the item "I have not accomplished many worthwhile things in this job".
  - **RPA4**: Response to the item "I do not feel like I'm at the end of my rope".
  - **RPA5**: Response to question "In my work, I do not deal with emotional problems very calmly".

3.5 Interdisciplinary Academic Writing Self-efficacy

Link to the dataset

This dataset contains the result of nursing students' responses to the Situated Academic Writing Self-Efficacy Scale (SAWSES) [69] questionnaire for interdisciplinary students (543 undergraduate and 264 graduate). Participating students were recruited from three higher education institutions and from the general public using social media. The survey contains 16 items based on Bandura's self-efficacy theory [70] and the model proposed by [71].

The questionnaire items are related to three dimensions. The first dimension is Writing Essentials, with three items related to synthesis, emotional control, and language. The second dimension is Relational Reflective Writing, with seven items related to relationship building with writing facilitators and the self through reflection. The third dimension is Creative Identity, with six items related to gaps in student achievement of transformative writing. Demographic data for age, gender, years in post-secondary, English language status, English writing status, and writing attitude are also included.

Table 12 Preview of the SAWSES demographic data

	Age	Gender	Writing-Attitude	TypeStudent	Ugyears	Gryears	WriteEnglish	SpeakEnglish
1	2	2	3	1	5	NA	1	1
2	2	2	1	1	5	NA	1	1
3	4	2	1	1	1	NA	3	4
4	2	2	3	1	3	NA	1	1
5	2	2	3	1	4	NA	1	1
6..806								
807	3	3	1	2	NA	3	1	1

The survey has been validated in a published article [72]. We make use of this dataset in Chapter 8 [12], devoted to clustering algorithms. The dataset is published under the CC0 1.0, which means that anyone can copy, modify, distribute it, even for commercial purposes, without asking permission from the authors. The dataset variables are described below.

First, we describe the **demographic** variables, which can be previewed in Table 12.

- **Age**: Age.
- **Gender**: Student's gender, 1 = male (24.91%), 2 = female (72.12%), 3 = non-binary (2.97%).
- **WritingAttitude**: Writing attitude, 1 = dislikes writing (44.73%), 2 = somewhere in between (14.37%), or 3 = likes writing (44.73%).
- **TypeStudent**: Academic level, 1 = undergraduate (67.29%), 2 = graduate (32.71%).
- **Ugyears**: Undergraduate years in the school for undergraduate students.
- **Gryears**: Graduate years in the school for graduate students.
- **WriteEnglish**: English writing status (1–5).
- **SpeakEnglish**: English language status (1–5).

Next, we present the questionnaire responses for each of the dimensions. The responses are on a scale of 0–100. Table 13 shows a preview of the data, and Fig. 8 presents a box plot with some the response distribution of the questionnaire.

- **Writing Essentials**
  - **overcome**: Response to the item "Even when the writing is hard, I can find ways to overcome my writing difficulties".
  - **words**: Response to the item "I can successfully use scholarly academic words and phrases when writing in my courses".
  - **synthesize**: Response to the item "I can combine or synthesize multiple sources I've read to create an original product or text".

Table 13 Preview of the SAWSES questionnaire responses

	Overcome	Words	Synthesize	Creativity	Meaning	Improve	Reflect	Spark	Ideas	Overall	Voice	Original	Discipline	Wander	Adapt	Feedback
1	100	100	100	100	100	100	100	100	80	100	80	100	97	82	100	86
2	100	100	100	61	70	100	100	64	92	100	100	100	100	93	100	100
3	100	100	100	100	100	100	100	65	81	100	100	100	100	95	100	71
4	51	82	49	20	42	76	93	84	91	83	46	18	75	25	21	53
5	16	100	37	75	100	100	80	67	100	59	59	47	100	27	90	100
6..806																
807	71	93	95	94	93	100	100	100	100	74	76	100	91	94	100	100

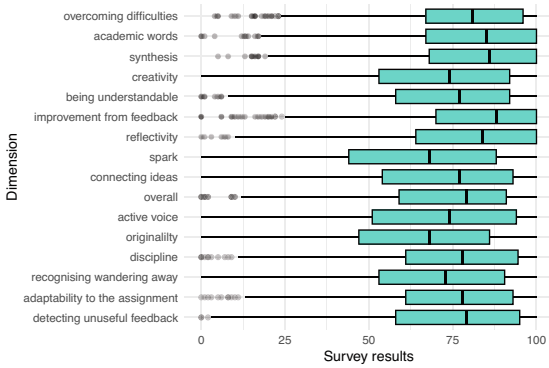


Fig. 8 Results of the SAWSES survey

- **Relational Reflective Writing**
  - **meaning**: Response to the item "When I write, I can think about my audience and write so they clearly understand my meaning".
  - **improve**: Response to the item "When I receive feedback on my writing, no matter how it makes me feel, I can use that feedback to improve my writing in the future".
  - **reflect**: Response to the item "When I reflect on what I am writing I can make my writing better".
  - **ideas**: Response to the item "When I read articles about my topic, the connections I feel with the ideas of other authors can inspire me to express my own ideas in writing".
  - **overall**: Response to the item "When I look at the overall picture I've presented in my writing, I can assess how all the pieces tell the complete story of my topic or argument".
  - **wander**: Response to the item "I can recognize when I've wandered away from writing what my audience needs to know and have begun writing about interesting, but unrelated, ideas".
  - **adapt**: Response to the item "With each new writing assignment, I can adapt my writing to meet the needs of that assignment".
  - **feedback**: Response to the item "When I seek feedback on my writing, I can decide when that feedback should be ignored or incorporated into a revision in my writing".
- **Creative Identity**
  - **creativity**: Response to the item "I can use creativity when writing an academic paper".
  - **spark**: Response to the item "I feel I can give my writing a creative spark and still sound professional".
  - **voice**: Response to the item "I feel I can develop my own writing voice (ways of speaking in my writing that are uniquely me)".

– **original**: Response to the item "Even with very specific assignment guidelines, I can find ways of writing my assignment to make it original or unique".
– **discipline**: Response to the item "I can comfortably express the concepts, language, and values of my discipline or major in my writing assignments".

3.6 Educators' Discussions in a MOOC (SNA)

Link to the dataset

This dataset belongs to two offerings of the MOOC "The Digital Learning Transition in K-12 Schools" [73]. The course was aimed at helping school and district leaders implement digital learning initiatives in K-12 education. The objectives of the course were for participants to understand the advantages of digital learning in schools, to assess the specific goals for their own school, and to devise a plan to achieve such goals. The course consisted of five learning units dealing with the schools of the future, teaching and learning culture, successful digital transition, leading the transition, and crowd sourcing. The MOOCs were offered to American as well as international teachers. There were two offerings of the MOOC, with minor variations regarding duration and groups. The dataset contains the interactions of the MOOC discussion forums and concerns teachers' communications throughout the courses. The dataset also contains the characteristics of the teachers, e.g., their professional roles, and experience.

The dataset is extensively described in a dedicated publication where the authors give details about the context and the courses, the files, and the fields contained in each file [74]. In this book, we use this dataset to illustrate dissimilarity-based clustering [39], Social Network Analysis [39] and Temporal Network Analysis [41]. The dataset is available with a CC0 1.0 license. Therefore, permission to copy, modify, and distribute, even for commercial purposes, is granted. As a standard Social Network Analysis dataset, it comes in two files for each of the courses (four in total), which we describe in detail below.

- **Edges file**: This file defines who interacted (the source or the sender of the communication) with whom (the target or the receiver of the communication). The edges file comes with other metadata, such as time, discussion topic and group. Table 14 presents a preview of one of the edge files, and Fig. 9 shows the network of collaboration between forum contributors.

  – **Sender**: Source of the communication identifier (1–445).
  – **Receiver**: Target of the communication identifier (1–445).
  – **Timestamp**: Timestamp of the intervention in "m/d/Y H:M" format, ranging from April 10th 2013 to June 8th 2013.
  – **Discussion Title**: Title of the discussion.
  – **Discussion Category**: Category of the discussion.

Table 14 Preview of the MOOC for educators discussion data (edges)

	Sender	Receiver	Timestamp	Discussion title	Discussion category
1	360	444	4/4/13 16:32	Most important change for your school or district?	Group N
2	356	444	4/4/13 18:45	Most important change for your school or district?	Group D-L
3	356	444	4/4/13 18:47	DLT Resources—Comments and Suggestions	Group D-L
4	344	444	4/4/13 18:55	Most important change for your school or district?	Group O-T
5	392	444	4/4/13 19:13	Most important change for your school or district?	Group U-Z
6..2528					
2529	11	11	6/16/13 17:12	How to make changes in teacher's understanding of importance of innovative teaching?	Curriculum & Instruction

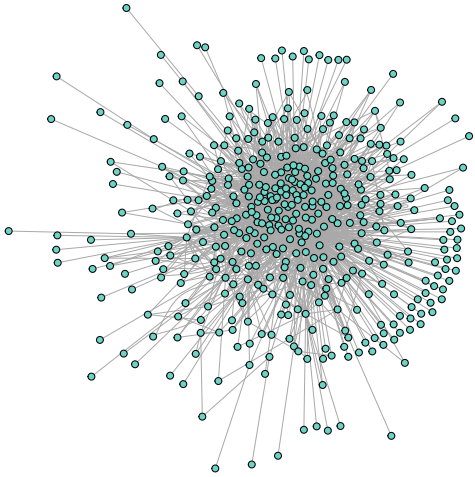


Fig. 9 MOOC participants' network

- **Nodes file**: The file defines the characteristics of the interacting teachers, their IDs, their professional status and expertise level. Table 15 shows a preview of one of the nodes file data.
  - **UID**: Teacher identifier (1–445).
  - **role1**: Role of the teacher.
  - **experience**: Level of experience (1–3).
  - **experience2**: Years of experience.
  - **country**: Country of origin.
  - **gender**: Teachers' gender, female (68.09%); male (31.69%).
  - **expert**: Level of expertise (0–1).

- **Centralities file**: The file contains the centrality measures of the participants which indicate their number of contributions (`OutDegree`), replies (`InDegree`), position in the network (`Closeness_total`), worth of their connections (`Eigen`), spread of their ideas (`Diffusion_degree`), and more. For more information on how to calculate centralities from interaction data, refer to Chapter 15 [39]. Table 16 shows a preview.
  - **name**: Teacher identifier (1–445).
  - **InDegree**: In-degree centrality. Number of responses received.
  - **OutDegree**: Out-degree centrality. Number of messages sent.
  - **Closeness_total**: Closeness centrality. Position in the network.
  - **Betweenness**: Betweenness centrality. Influential position.
  - **Eigen**: Eigen centrality. Worth of connections.
  - **Diffusion.degree**: Diffusion degree centrality [75]. Spread of ideas.
  - **Coreness**: Coreness centrality. Spreading capability.
  - **Cross_clique_connectivity**: Cross clique connectivity. Facilitation of information propagation.

Table 15 Preview of the MOOC for educators discussion data (nodes)

	UID	Facilitator	role1	Experience	experience2	Grades	Location	Region	Country	Group	Gender	Expert	Connect
1	1	0	libmedia	1	6 to 10	secondary	VA	South	US	UZ	female	0	1
2	2	0	classteaching	1	6 to 10	secondary	FL	South	US	DL	female	0	0
3	3	0	districtadmin	2	11 to 20	generalist	PA	Northeast	US	OT	female	0	1
4	4	0	classteaching	2	11 to 20	middle	NC	South	US	N	female	0	0
5	5	0	otheredprof	3	20+	generalist	AL	South	US	AC	female	0	0
6..444													
445	445	1	otheredprof	3	20+	generalist	NC	South	US	N	female	0	0

Table 16 Preview of the MOOC for educators discussion data (nodes)

	Name	InDegree	OutDegree	Closeness_total	Betweenness	Eigen	Diffusion.degree	Coreness	Cross_clique_connectivity
1	1	20	33	0.0010952903	1258.14319	0.205523262	1865	18	305
2	2	2	5	0.0008084074	26.52429	0.010717789	218	6	13
3	3	2	4	0.0007987220	30.60112	0.008623924	191	6	11
4	4	2	14	0.0010193680	72.52345	0.080264834	965	13	37
5	5	16	17	0.0010604454	309.03274	0.161503654	1508	18	154
6..444									
445	445	276	56	0.0013175231	16690.42611	0.699247801	3574	31	2218

3.7 High School Learners' Interactions (SNA)

Link to the dataset

The next dataset [76] concerns a course of interactions among 30 students in a high school in Kenitra, Morocco. The course under examination had a duration of two months and covered topics related to computer science: the computer information system, algorithms and programming. The course was implemented in the Moodle LMS, using the forum as a discussion space. Students' interactions were aimed at communicating, discussing and exchanging knowledge among them.

The dataset has been analyzed using social network analysis, is briefly described in an article [77], and is shared under Creative Commons license CC BY 4.0, which means that anyone can share, copy and modify this dataset so long as appropriate credit is given. The dataset includes two files described below.

- **Edges file**: The file contains the interactions source, target and weights. Table 17 shows a preview of the edge files. Figure 10 presents the graph of all the interactions on the network.
  - **source**: Source node identifier (1–21).
  - **Target**: Target node identifier (1–21).
  - **W**: Weight of the link (the value is always 1).

Table 17 Preview of the High school learners' interactions data (edges)

	Source	Target	W
1	17	6	1
2	17	5	1
3	17	11	1
4	17	17	1
5	17	6	1
6..221			
222	21	8	1

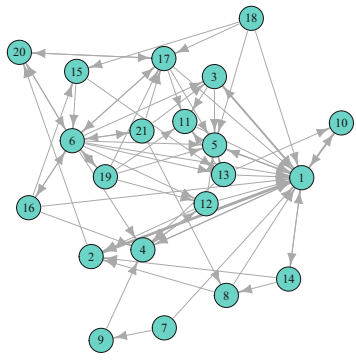


Fig. 10 High school learners' network

Table 18 Preview of the High school learners' interactions data (nodes)

	ID	Username	Name	Genre	Date de naissance
1	1	L1	Aya	F	22/11/2002
2	2	L2	Nouhaila Mahsana	F	25/10/2002
3	3	L3	Said Rhioui	M	26/02/2001
4	4	L4	Mehdi ouchne	M	04/06/2000
5	5	L5	ilyass liagoubi	M	03/04/2001
6..29					
30	30	L30	Kaoula Elyamani	F	06/05/2002

- **Nodes file**: contains the characteristics of the interacting students, e.g., gender and age. Table 18 presents a preview of the dataset.
  - **ID**: Student identifier (1–30).
  - **Username**: Username of the student.
  - **name**: Name of the student.
  - **genre**: Gender of the student F (n = 23); M (n = 7).
  - **Date de naissance**: Birthdate of the student in format "D/M/Y".

3.8 Interactions in an LMS Forum from a Programming Course (SNA)

Link to the dataset

This dataset includes message board data collected from a programming undergraduate course in a higher education institution in Spain using the Moodle LMS. The most particular characteristic of the course is that it follows the CTMTC (Comprehensive Training Model of the Teamwork Competence) methodology [78], which allows individualized training and assessment of teamwork across all stages of teamwork-based learning: storming, norming, performing, delivery and documentation [79].

This is a mandatory course with a workload of 6 ECTS. The data dates back to the first semester of the 2014–2015 academic year. The course offers foundational knowledge on programming and numeric calculus, has a strong focus on the development of algorithms, and a hands-on approach to teaching. The course starts with a two-hour long introductory session; after this session, students work in class and out of class during the whole semester on a team project, following the CTMTC methodology. Individual evidence of teamwork is collected from forum activity, where the work phases are presented, and group evidence of teamwork is collected from Dropbox and wikis.

The dataset refers to individual evidence of teamwork; in other words, it contains information about forum interactions during the course. The original dataset, obtained from Moodle logs, was processed with the help of GraphFES [80] to

provide condensed information. The output of GraphFES consists of three different datasets: views, or the number of times user `a` read a message posted by user `b`; replies, which informs about the number of replies from user `a` to user `b`; and messages, which provides a network with the hierarchical structure of messages in the forum. This dataset has been used previously in [81] and is now being publicly released under a CC 4.0 BY-NC-SA license, which means that anyone is free to share, adapt, and distribute the data as long as appropriate credit is given, it is not used for commercial purposes, and the original license is kept. The dataset is used in Chapter 16 [40] of this book, about community detection. This dataset presents the replies network, a directed graph, and consists of an edges file and a nodes file.

- **Edges file**: this file includes information about who (attribute `source`) interacted with (replied to) whom (attribute `target`); in other words, the sender and the receiver of the informational exchange, and how many times that exchange happened during the course (attribute `weight`), considering all messages exchanged. The dataset includes a total of 662 weighed edges. Table 19 presents a preview of the edges file data, and Fig. 11 represents the complete network of interactions among the Moodle users.
  - **source**: Source node identifier (108 distinct values).
  - **target**: Target node identifier (108 distinct values).
  - **weight**: Weight of the link.

- **Nodes file**: this file contains information about all users with access to the course in the Moodle space, including students, instructors and administrators. The file includes a total of 124 nodes (users); of these, 110 users are students, distributed in 19 groups of between 5 and 7 members each. The file includes an identifier for each user (attribute `id`), the username (attribute `user`; after anonymization, all usernames have the format `user_id`), the number of initial posts, which refers to the number of first posts in a thread (attribute `initPosts`), the number of replies, or posts that were a reply to another post (attribute `replyPosts`) and the total number of posts by that user in the forum (attribute `totalPosts`), which is the sum of `initPosts` and `replyPosts`. It is worth noting that `user_55`, a central node of the network, corresponds to the main instructor of the course. Table 20 shows a preview of the nodes file.
  - **User**: User identifier. There are 124 distinct users.

Table 19 Preview of the programming course interaction data (edges)

	Source	Target	Weight
1	192	164	1
2	164	55	2
3	139	142	1
4	142	55	2
5	175	55	5
6..661			
662	194	153	1

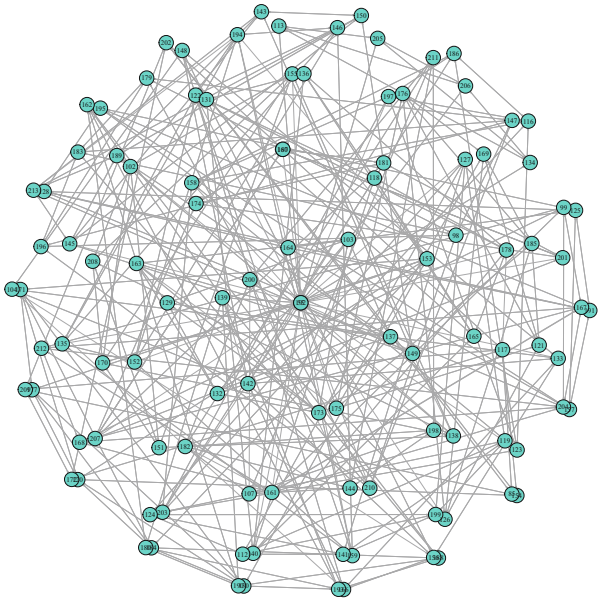


Fig. 11 Interactions of the users in the Moodle discussion forum

Table 20 Preview of the programming course interaction data (nodes)

	Id	User	initPosts	replyPosts	totalPosts
1	54	user_54	0	0	0
2	55	user_55	23	2	25
3	56	user_56	0	0	0
4	57	user_57	0	0	0
5	58	user_58	0	0	0
6..123					
124	214	user_214	1	0	1

- **initPosts**: Number of first posts in a thread.
- **replyPosts**: Number of replies to posts in a thread.
- **totalPosts**: Total number of posts by a user in the forum.

3.9 Engagement and Achievement Throughout a Study Program

Link to the dataset

This dataset contains simulated data of students' online engagement and academic achievement throughout a study program. The dataset has been simulated based on the results of a published article [82]. The article used students' logs

extracted from a university's Moodle LMS for all the subjects and for all the students who attended the years: 2015, 2016, 2017 and 2018. The logs were used to derive indicators of engagement, such as frequency of performing learning activities (course browsing, forum consumption, forum contribution, and lecture viewing), session count, total session time, active days and regularity of each activity. Regularity of viewing the course main page, for example, was calculated by dividing main page browse actions for the given student over the total main page browse actions over the course duration; the resulting probabilities are used within a Shannon entropy formula to calculate the entropy.

Then, Latent Class Analysis was used to cluster students into engagement states for each course: Active, Average or Disengaged. Achievement was measured through course final grades, which were divided into tertiles: Achiever, Intermediate and Low. Hence, for each course, students had an engagement state, and an achievement state. The motivation and process of deriving these states from students' engagement indicators is explained in Chapter 11 [31].

The simulated dataset contains the data for 142 students for 8 sequential courses, including their engagement and achievement states, as well as covariate data. It is shared with a CC BY 4.0 license, which means that anyone is free to share, adapt, and distribute the data, as long as appropriate credit is given. The dataset is used in Chapter 13 [33] of this book, to illustrate multi-channel sequence analysis. A preview of the dataset can be seen on Table 22. A visual representation of the evolution of engagement and achievement throughout the program is depicted in Fig. 13. The dataset contains two files:

3.9.1 Longitudinal Engagement Indicators and Grades

The file `LongitudinalEngagement.csv` contains all of the engagement indicators per student and course (frequency, duration and regularity of learning activities) as well as the final grade. Each column is described below and a preview can be seen in Table 21 and in Fig. 12.

- **UserID**: User identifier. There are 142 distinct users.
- **CourseID**: Course identifier. There are 38 distinct courses.
- **Sequence**: Course sequence for the student (1–8).
- **Freq_Course_View**: Number of views of the course main page.
- **Freq_Forum_Consume**: Number of views of the forum posts.
- **Freq_Forum_Contribute**: Number of forum posts created.
- **Freq_Lecture_View**: Number of lectures viewed.
- **Regularity_Course_View**: Regularity of visiting the course main page.
- **Regularity_Lecture_View**: Regularity of visiting the lectures.
- **Regularity_Forum_Consume**: Regularity of reading forum posts.
- **Regularity_Forum_Contribute**: Regularity of writing forum posts.
- **Session_Count**: Number of online learning sessions.
- **Total_Duration**: Total activity time online.

Table 21 Preview of engagement indicators and final grades throughout the eight courses

	UserID	CourseID	Sequence	FCV	FFCs	FFCt	FLV	RCV	RLV	RFCs	RFCt	SC	TD	AD	FG
1	D2C5F64E	C6107FC4	1	150	251	79	132	0.42	0.38	0.18	0.83	121	53330	12	50.04
2	D2C5F64E	4C3F37F0	2	98	84	17	108	0.26	0.49	0.11	0.03	53	14465	5	38.86
3	D2C5F64E	E54A52A3	3	254	354	34	284	0.29	0.56	0.14	0.07	159	64324	12	44.73
4	D2C5F64E	AB7EC624	4	332	825	185	256	0.63	0.84	0.53	0.53	287	122821	19	48.99
5	D2C5F64E	B0E95213	5	386	960	233	356	0.74	0.80	0.75	0.56	321	148792	23	47.84
6..1135															
1136	B235D544	4B912491	8	152	546	203	56	0.32	0.23	0.28	0.48	125	88432	10	69.12

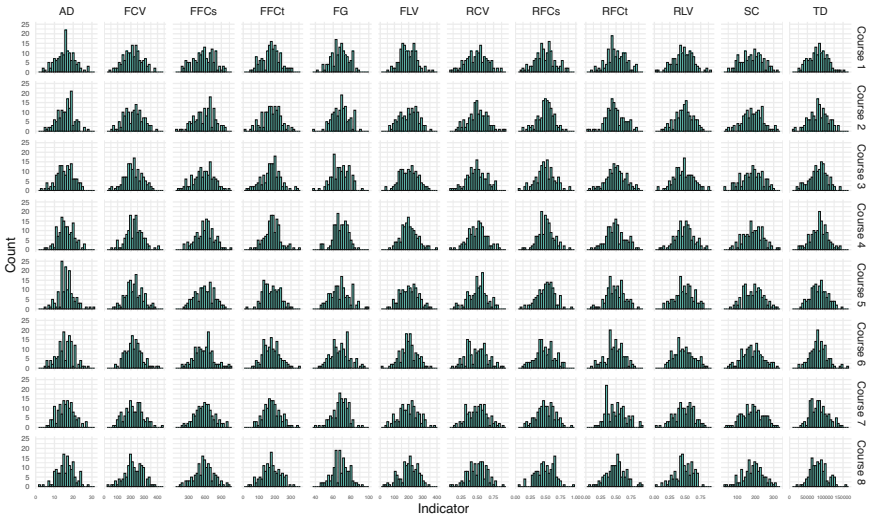


Fig. 12 Histogram of engagement indicators and final grade throughout the eight courses

- **Active_Days**: Number of active days (with online activity).
- **Final_Grade**: Final grade (0–100).

3.9.2 Longitudinal Engagement and Achievement States

The file `SequenceEngagementAchievement.xlsx` contains students' engagement and achievement states for each course as well as covariates (their previous grade, their attitude towards learning, and their gender). Engagement states were obtained by applying model-based clustering techniques to the engagement indicators in the previous file. Achievement states were obtained in a similar way for the final grade. The dataset columns are described below and a preview of the data can be seen at Table 22 and a graphical representation is shown in Fig. 13.

- **UserID**: User identifier. There are 142 distinct users.
- **CourseID**: Course identifier. There are 38 distinct courses.
- **Sequence**: Course sequence for the student (1–8).
- **Engagement**: Engagement state. There are 3 distinct states: Active (29.93%), Average (47.98%), amd Disengaged (22.10%).
- **Final_Grade**: Final grade of each student for each course (1–100).
- **Achievement**: Achievement state calculated using model-based clustering. There are 3 distinct states: Achiever (39.26%), Intermediate (23.33%), and Low (37.41%).

Table 22 Preview of the dataset about engagement and achievement throughout the eight courses

	UserID	CourseID	Sequence	Engagement	Final_Grade	Achievement	AchievementNtile	Prev_grade	Attitude	Gender
1	00050F0E	4C3F37F0	1	Average	72.27	Achiever	Intermediate	6.826613	12.92833	Male
2	00050F0E	E54A52A3	2	Disengaged	72.56	Achiever	Achiever	6.826613	12.92833	Male
3	00050F0E	AB7EC624	3	Average	78.78	Achiever	Achiever	6.826613	12.92833	Male
4	00050F0E	B0E95213	4	Average	74.19	Achiever	Achiever	6.826613	12.92833	Male
5	00050F0E	0B301F55	5	Average	87.35	Achiever	Achiever	6.826613	12.92833	Male
6..1135										
1136	FE2E4C85	79ED6873	8	Active	87.69	Achiever	Achiever	4.189420	20.00000	Male

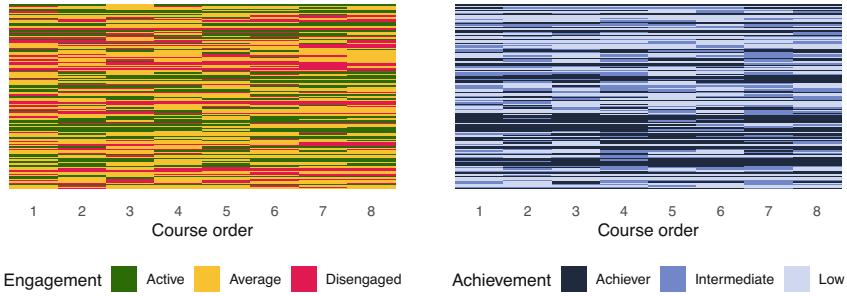


Fig. 13 Sequence and engagement for each student across eight courses

- **AchievementNtile**: Achievement state calculated using tertiles. There are 3 distinct states: Achiever (33.27%), Intermediate (33.36%), and Low (33.36%).
- **Prev_grade**: GPA with which the student applied to the program (1–10).
- **Attitude**: Attitude towards learning (0–20).
- **Gender**: Gender (male/female). There are 44% females and 56% males.

3.10 University Students' Basic Need Satisfaction, Self-regulated Learning and Well-Being During COVID-19

Link to the dataset

This dataset contains the results of a survey investigating students' psychological characteristics related to their well-being during the COVID-19 pandemic. The variables under study are related to the satisfaction of basic psychological needs (relatedness, autonomy, and experienced competence), self-regulated learning, positive emotion and intrinsic learning motivation. Moreover, the dataset contains demographic variables, such as country, gender, and age. The data were collected from 6071 students from Austria and Finland. There are, however, 564 records with missing responses to at least one item.

This dataset has been used in a published study to examine the relationships between the different variables using SEM [83]. The dataset has been used in Chapter 19 [21] of this book, to illustrate the implementation of psychological networks. The dataset has been published under a CC BY 4.0 license, which means that you are free to use and adapt the data but you must give appropriate credit, provide a link to the license, and indicate if changes were made. A preview of the dataset can be found on Table 23. A summary of the responses is in Fig. 14. Below, we describe each of the dataset columns.

- **Demographic data**
  - **country**: Country of the student: 0 = Austria (78.60%), 1 = Finland (21.40%).

Table 23 Preview of the COVID-19 well-being survey data

	Country	Gender	Age	sr1	sr2	sr3	comp1	comp2	comp3	auto1	auto2	auto3	pa1	pa2	pa3	lm1	lm2	lm3	gp1	gp2	gp3
1	0	1	36	5	4	2	2	3	3	3	4	NA	4	3	2	5	5	2	NA	NA	NA
2	0	2	36	4	4	1	4	4	3	4	4	3	4	4	4	5	5	4	3	3	4
3	0	2	36	1	1	1	3	4	2	5	4	4	4	4	3	4	4	4	2	1	2
4	0	2	36	1	2	1	2	2	2	4	3	3	2	2	2	3	3	3	2	1	2
5	0	1	36	1	1	3	3	2	3	4	4	1	3	3	2	4	3	3	2	5	3
6..7723																					
7724	1	1	69	2	2	1	2	2	2	2	2	2	3	3	3	1	1	1	2	2	2

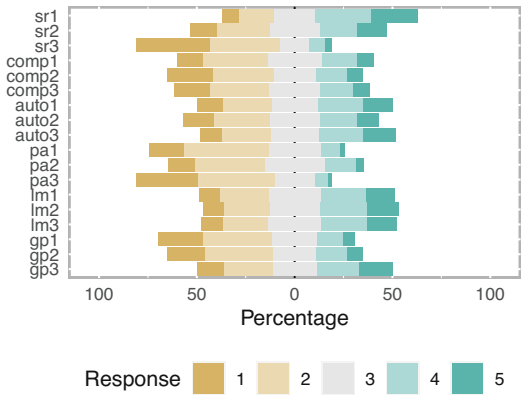


Fig. 14 Results of COVID-19 well-being survey

- **gender**: Gender of the student: 1 = Female (70.74%), 2 = Male (28.25%), 3 = Other (0.70%).
- **age**: Age of the student.

Below we describe the items of the questionnaire for each construct. The possible responses are: 1 = strongly agree, 2 = agree, 3 = somewhat agree, 4 = disagree, 5 = strongly disagree.

- **Basic Psychological Needs: Relatedness**
  - **sr1**: Response to the item "Currently, I feel connected with my fellow students".
  - **sr2**: Response to the item "Currently, I feel supported by my fellow students".
  - **sr3**: Response to the item "Currently, I feel connected with the people who are important to me (family, friends)".
- **Basic Psychological Needs: Competence**
  - **comp1**: Response to the item "Currently, I am dealing well with the demands of my studies".
  - **comp2**: Response to the item "Currently, I have no doubts about whether I am capable of doing well in my studies".
  - **comp3**: Response to the item "Currently, I am managing to make progress in studying for university".
- **Basic Psychological Needs: Autonomy**
  - **auto1**: Response to the item "Currently, I can define my own areas of focus in my studies".
  - **auto2**: Response to the item "Currently, I can perform tasks in the way that best suits me".
  - **auto3**: Response to the item "In the current home-learning situation, I seek out feedback when I need it".

- **Positive Emotion**
  - **pa1**: Response to the item "I feel good".
  - **pa2**: Response to the item "I feel confident".
  - **pa3**: Response to the item "Even if things are difficult right now, I believe that everything will turn out all right".
- **Intrinsic learning motivation**
  - **lm1**: Response to the item "Currently, doing work for university is really fun".
  - **lm2**: Response to the item "Currently, I am really enjoying studying and doing work for university".
  - **lm3**: Response to the item "Currently, I find studying for university really exciting".
- **Self-regulated learning**
  - **gp1**: Response to the item "In the current home-learning situation, I plan my course of action".
  - **gp2**: Response to the item "In the current home-learning situation, I think about how I want to study before I start".
  - **gp3**: Response to the item "In the current home-learning situation, I formulate learning goals that I use to orient my studying".

4 Discussion

In this chapter, we have provided an overview of the types of data operationalized in learning analytics research. We covered a wide spectrum of data types, ranging from foundational demographic information to the footprints left by the interactions of students with online learning technologies, including clicks, activities, social interactions, and assessment data. We have pointed to some of the most commonly employed analytical techniques for each type of data and we referred the reader to the chapters of the book that have covered each type of analysis. Thereafter, we presented a curation of illustrative datasets. We have described each dataset in detail, describing and representing the relevant variables. We also acknowledged the ways each dataset have been analyzed throughout the remaining chapters of the book.

We must disclose that collecting learners' data is not an easy endeavor. First and foremost, it is crucial to consider the ethical implications of collecting and using different types of data, and to comply with data protection laws and regulations [14]. Moreover, it is important to ensure the data quality to draw relevant conclusions from the data, especially in scenarios where data come from heterogeneous sources and are provided in large quantities [84], such as in the educational field [85]. These requirements make finding good-quality open datasets online extremely challenging. In this regard, we hope that the selection offered in this chapter is useful for the reader beyond the scope of the book. A few articles have offered other dataset collections suitable for learning analytics [86] or educational data

mining [87]. Moreover, the reader is encouraged to consult open data repositories where datasets are continuously published in multiple fields: Zenodo (https://zenodo.org), US Department of Education Open Data Platform (https://data.ed.gov), Harvard Dataverse (https://dataverse.harvard.edu), European Data Portal (https://data.europa.eu), Mendeley Data (https://data.mendeley.com), openICPSR (https://www.openicpsr.org), Google Dataset Search (https://datasetsearch.research.google.com), figshare (https://figshare.com), Open Science Framework (https://osf.io), or data.world (https://data.world).

References

1. Sclater N (2017) Data. In: Learning analytics explained. Routledge, New York, pp 78–87
2. Nistor N, Hernández-García Á (2018) What types of data are used in learning analytics? An overview of six cases. Comput Human Behav 89:335–338. https://doi.org/10.1016/j.chb.2018.07.038
3. Li W, Sun K, Schaub F, Brooks C (2021) Disparities in Students' propensity to consent to learning analytics. Int J Artif Intell Educ 32:564–608. https://doi.org/10.1007/s40593-021-00254-2
4. Rodríguez-Hernández CF, Cascallar E, Kyndt E (2020) Socio-economic status and academic performance in higher education: a systematic review. Educ Res Rev 29:100305. https://doi.org/10.1016/j.edurev.2019.100305
5. Mengash HA (2020) Using data mining techniques to predict student performance to support decision making in university admission systems. IEEE Access 8:55462–55470. https://doi.org/10.1109/access.2020.2981905
6. Mullen CA (2019) Does modality matter? A comparison of aspiring leaders' learning online and face-to-face. J Further Higher Educ 44:670–688. https://doi.org/10.1080/0309877x.2019.1576859
7. Tikka S, Kopra J, Heinäniemi M, López-Pernas S, Saqr M (2024, this volume) Basics of R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
8. Kopra J, Tikka S, Heinäniemi M, López-Pernas S, Saqr M (2024, this volume) Data cleaning and wrangling. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
9. López-Pernas S, Misiejuk K, Tikka S, Saqr M, Kopra J, Heinäniemi M (2024, this volume) Visualizing and reporting educational data with r. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
10. Meaney M, Fikes T (2022) Adding a demographic lens to cluster analysis of participants in entry-level massive open online courses (MOOCs). In: Proceedings of the Ninth ACM conference on learning @ scale. https://doi.org/10.1145/3491140.3528306
11. Scrucca L, Saqr M, López-Pernas S, Murphy K (2024, this volume) An introduction and r tutorial to model-based clustering in education via latent profile analysis. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
12. Murphy K, López-Pernas S, Saqr M (2024, this volume) Dissimilarity-based cluster analysis of educational data: a comparative tutorial using r. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
13. Du X, Yang J, Shelton BE, Hung J-L, Zhang M (2019) A systematic meta-review and analysis of learning analytics research. Behav Inf Technol 40:49–62. https://doi.org/10.1080/0144929x.2019.1669712
14. Slade S, Prinsloo P (2013) Learning analytics. Am Behav Sci 57:1510–1529. https://doi.org/10.1177/0002764213479366

15. Tempelaar D, Rienties B, Nguyen Q (2021) The contribution of dispositional learning analytics to precision education. Educ Technol Soc 24:109–122. https://www.jstor.org/stable/26977861
16. Brenner PS, DeLamater J (2016) Lies, damned lies, and survey self-reports? Identity as a cause of measurement bias. Soc Psychol Quart 79:333–354. https://doi.org/10.1177/0190272516628298
17. Tikka S, Kopra J, Heinäniemi M, López-Pernas S, Saqr M (2024, this volume) Basic statistics with R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
18. Oster M, Lonn S, Pistilli MD, Brown MG (2016) The learning analytics readiness instrument. In: Proceedings of the sixth international conference on learning analytics & knowledge - LAK '16. https://doi.org/10.1145/2883851.2883925
19. Vogelsmeier LVDE, Saqr M, López-Pernas S, Jongerling J (2024, this volume) Factor analysis in education research using R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
20. Jongerling J, López-Pernas S, Saqr M, Vogelsmeier L (2024, this volume) Structural equation modeling with R for education scientists. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
21. Saqr M, Beck E, López-Pernas S (2024, this volume) Psychological networks. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
22. Ullmann T, Rienties B (2021) Using text analytics to understand open-ended student comments at scale: insights from four case studies. Springer International Publishing, Berlin, pp 211–233
23. Henrie CR, Bodily R, Larsen R, Graham CR (2017) Exploring the potential of LMS log data as a proxy measure of student engagement. J Comput Higher Educ 30:344–362. https://doi.org/10.1007/s12528-017-9161-1
24. Alvarez P, Fabra J, Hernandez S, Ezpeleta J (2016) Alignment of teacher's plan and students' use of LMS resources. Analysis of moodle logs. In: 2016 15th international conference on information technology based higher education and training (ITHET). https://doi.org/10.1109/ithet.2016.7760720
25. Saqr M, López-Pernas S (2021) The longitudinal trajectories of online engagement over a full program. Comput Educ 175:104325. https://doi.org/10.1016/j.compedu.2021.104325
26. Jovanović J, Gašević D, Dawson S, Pardo A, Mirriahi N (2017) Learning analytics to unveil learning strategies in a flipped classroom. Internet Higher Educ 33:74–85. https://doi.org/10.1016/j.iheduc.2017.02.001
27. López-Pernas S, Saqr M (2024, this volume) Process mining. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
28. Ahmad Uzir N, Gašević D, Matcha W, Jovanović J, Pardo A, Lim L-A, Gentili S (2019) Discovering time management strategies in learning processes using process mining techniques. Springer International Publishing, Berlin, pp 555–569
29. Saqr M, López-Pernas S, Jovanović J, Gašević D (2023) Intense, turbulent, or wallowing in the mire: a longitudinal study of cross-course online tactics, strategies, and trajectories. Internet Higher Educ 57:100902. https://doi.org/10.1016/j.iheduc.2022.100902
30. Saqr M, López-Pernas S, Helske S, Durand M, Murphy K, Studer M, Ritschard G (2024) Sequence analysis in education: principles, technique, and tutorial with r. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, Berlin
31. López-Pernas S, Saqr M (2024, this volume) Modelling the dynamics of longitudinal processes in education. A tutorial with R for the VaSSTra method. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
32. Helske J, Helske S, Saqr M, López-Pernas S, Murphy K (2024, this volume) A modern approach to transition analysis and process mining with Markov models: a tutorial with R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
33. López-Pernas S, Saqr M, Helske S, Murphy K (2024, this volume) Multichannel sequence analysis in educational research using r. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer

34. Matcha W, Gašević D, Ahmad Uzir N, Jovanović J, Pardo A, Maldonado-Mahauad J, Pérez-Sanagustín M (2019) Detection of learning strategies: a comparison of process, sequence and network analytic approaches. Springer International Publishing, Berlin, pp 525–540
35. Saqr M, López-Pernas S (2022) How CSCL roles emerge, persist, transition, and evolve over time: a four-year longitudinal study. Comput Educ 189:104581. https://doi.org/10.1016/j.compedu.2022.104581
36. Saqr M, López-Pernas S (2021) Modelling diffusion in computer-supported collaborative learning: a large scale learning analytics study. Int J Comput-Support Collab Learn 16:441–483. https://doi.org/10.1007/s11412-021-09356-4
37. Dowell NMM, Nixon TM, Graesser AC (2018) Group communication analysis: a computational linguistics approach for detecting sociocognitive roles in multiparty interactions. Behav Res Methods 51:1007–1041. https://doi.org/10.3758/s13428-018-1102-z
38. Saqr M, Elmoazen R, Tedre M, López-Pernas S, Hirsto L (2022) How well centrality measures capture student achievement in computer-supported collaborative learning? – A systematic review and meta-analysis. Educ Res Rev 35:100437. https://doi.org/10.1016/j.edurev.2022.100437
39. Saqr M, López-Pernas S, Conde MÁ, Hernández-García Á (2024, this volume) Social network analysis: a primer, a guide and a tutorial in R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
40. Hernández-García Á, Cuenca-Enrique C, Traxler A, López-Pernas S, Conde MÁ, Saqr M (2024, this volume) Community detection in learning networks using R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
41. Saqr M (2024, this volume) Temporal network analysis: introduction and methods and analysis with R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
42. Shaffer DW, Collier W, Ruis AR (2016) A tutorial on epistemic network analysis: analyzing the structure of connections in cognitive, social, and interaction data. J Learn Anal 3:9–45. https://doi.org/10.18608/jla.2016.33.3
43. Tan Y, Swiecki Z, Ruis A, Shaffer D (2024, this volume) Epistemic network analysis and ordered network analysis in learning analytics. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
44. Teasley SD (2019) Learning analytics: where information science and the learning sciences meet. Inf Learn Sci 120:59–73. https://doi.org/10.1108/ils-06-2018-0045
45. Gordillo A, Lopez-Fernandez D, López-Pernas S, Quemada J (2020) Evaluating an educational escape room conducted remotely for teaching software engineering. IEEE Access 8:225032–225051. https://doi.org/10.1109/access.2020.3044380
46. Li KC, Wong BTM (2020) The use of student response systems with learning analytics: a review of case studies (2008–2017). Int J Mob Learn Organ 14:63. https://doi.org/10.1504/ijmlo.2020.103901
47. Namoun A, Alshanqiti A (2020) Predicting student performance using data mining and learning analytics techniques: a systematic literature review. Appl Sci 11:237. https://doi.org/10.3390/app11010237
48. Jovanovic J, López-Pernas S, Saqr M (2024, this volume) Predictive modelling in learning analytics using R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer
49. Saqr M, López-Pernas S (2022) The curious case of centrality measures: a large-scale empirical investigation. J Learn Anal 9:13–31. https://doi.org/10.18608/jla.2022.7415
50. Blikstein P (2013) Multimodal learning analytics. In: Proceedings of the third international conference on learning analytics and knowledge. Association for Computing Machinery, New York, pp 102–106
51. Mu S, Cui M, Huang X (2020) Multimodal data fusion in learning analytics: a systematic review. Sensors 20. https://doi.org/10.3390/s20236856
52. Sharma K, Giannakos M (2020) Multimodal data capabilities for learning: What can multimodal data tell us about learning? Br J Educ Technol 51:1450–1484. https://doi.org/10.1111/bjet.12993

53. Kubsch M, Caballero D, Uribe P (2022) Once more with feeling: emotions in multimodal learning analytics. In: Giannakos M, Spikol D, Di Mitri D, Sharma K, Ochoa X, Hammad R (eds) The multimodal learning analytics handbook. Springer International Publishing, Cham, pp 261–285
54. Bleck M, Le N-T (2022) A physiology-aware learning analytics framework. In: Giannakos M, Spikol D, Di Mitri D, Sharma K, Ochoa X, Hammad R (eds) The multimodal learning analytics handbook. Springer International Publishing, Cham, pp 231–257
55. Saqr M, López-Pernas S (2024, this volume) Why learning and teaching learning analytics is hard: an experience from a real-life LA course using LA methods. In: Proceedings of the eleventh international conference on technological ecosystems for enhancing multiculturality (TEEM'23). Springer, Bragança
56. Hasan R, Palaniappan S, Mahmood S, Abbas A, Sarker KU (2021) Dataset of students' performance using student information system, moodle and the mobile application 'eDify'. Data 6: https://doi.org/10.3390/data6110110
57. Hasan R (2021) Dataset of Student's Performance using Student Information System, Moodle and Mobile Application 'eDify'
58. Hasan R, Palaniappan S, Raziff ARA, Mahmood S, Sarker KU (2018) Student academic performance prediction by using decision tree algorithm. In: 2018 4th international conference on computer and information sciences (ICCOINS). IEEE, pp 1–5
59. Hasan R, Palaniappan S, Mahmood S, Sarker KU, Abbas A (2020) Modelling and predicting student's academic performance using classification data mining techniques. Int J Bus Inf Syst 34:403–422
60. Rodríguez S, Valle A, Piñeiro I, Vieites T, González-Suárez R, Rodríguez-Llorente C (2020) School engagement, SRL and academic achievement
61. Fredricks JA, Blumenfeld P, Friedel J, Paris A (2005) School engagement. What do children need to flourish? Conceptualizing and measuring indicators of positive development, pp 305–321
62. Cleary TJ (2006) The development and validation of the self-regulation strategy inventory—self-report. J School Psychol 44:307–322
63. Estévez I, Rodríguez-Llorente C, Piñeiro I, González-Suárez R, Valle A (2021) School engagement, academic achievement, and self-regulated learning. Sustainability 13: https://doi.org/10.3390/su13063011
64. Prasojo LD, Habibi A, Yaakob MFM, Pratama R, Yusof MR, Mukminin A, Suyanto, Hanum F (2020) Teachers' burnout: a SEM analysis in an Asian context. Heliyon 6:e03144. https://doi.org/10.1016/j.heliyon.2019.e03144
65. Villa A, Calvete E (2001) Development of the teacher self-concept evaluation scale and its relation to burnout. Stud Educ Eval 27:239–255. https://doi.org/10.1016/s0191-491x(01)00028-1
66. Yu G, Xin T, Shen J (1995) Teacher's sense of teaching efficacy: its structure and influencing factors. Acta Psychol Sin 27:159
67. Champion DF, Westbrook BW (1984) Maslach burnout inventory. Meas Eval Couns Dev 17:100–102. https://doi.org/10.1080/07481756.1984.12022754
68. Prasojo LD, Habibi A, Yaakob MFM, Pratama R, Yusof MR, Mukminin A, Suyanto, Hanum F (2020) Dataset relating to the relationship between teacher self-concept and teacher efficacy as the predictors of burnout: a survey in Indonesian education. Data Brief 30:105448. https://doi.org/10.1016/j.dib.2020.105448
69. Mitchell K (2020) Interdisciplinary undergraduate and graduate student data. https://doi.org/10.7910/DVN/M07HQ7. Harvard Dataverse
70. Bandura A, Freeman WH, Lightsey R (1999) Self-efficacy: the exercise of control. J Cogn Psychother 13:158–166. https://doi.org/10.1891/0889-8391.13.2.158
71. Pajares F, Valiante G (2006) Self-efficacy beliefs and motivation in writing development. In: Handbook of writing research. The Guilford Press, New York, pp 158–170
72. Mitchell KM, McMillan DE, Lobchuk MM, Nickel NC, Rabbani R, Li J (2021) Development and validation of the situated academic writing self-efficacy scale (SAWSES). Assess Writ 48:100524. https://doi.org/10.1016/j.asw.2021.100524

73. Kellogg S, Edelmann A (2015) Massively Open Online Course for Educators (MOOC-Ed) network dataset. https://doi.org/10.7910/DVN/ZZH3UB. Harvard Dataverse
74. Kellogg S, Edelmann A (2015) Massively Open Online Course for Educators (MOOC-Ed) network dataset. Br J Educ Technol 46:977–983. https://doi.org/10.1111/bjet.12312
75. Saqr M, López-Pernas S (2021) Modelling diffusion in computer-supported collaborative learning: a large scale learning analytics study. Int J Comput-Support Collab Learn 16:441–483. https://doi.org/10.1007/s11412-021-09356-4
76. Adraoui M, Akachar E, Retbi A, Idrissi MK, Bennani S (2022) Dataset of learners' interactions in forum discussions [dataset]. Mendeley. https://doi.org/10.17632/CKNF9FVYBR.1
77. Adraoui M, Retbi A, Idrissi MK, Bennani S (2017) Social learning analytics to describe the learners' interaction in online discussion forum in moodle. In: 2017 16th international conference on information technology based higher education and training (ITHET). IEEE
78. Lerís D, Fidalgo Á, Sein Echaluce ML (2014) A comprehensive training model of the teamwork competence. Int J Learn Intellect Cap 11:1. https://doi.org/10.1504/ijlic.2014.059216
79. Fidalgo-Blanco Á, Lerís D, Sein-Echaluce ML, García-Peñalvo FJ, et al. (2015) Monitoring indicators for CTMTC: comprehensive training model of the teamwork competence in engineering domain. Int J Eng Educ 31(Extra 3):829–838
80. Chaparro-Peláez J, Acquila-Natale E, Iglesias-Pradas S, Suárez-Navas I (2015) A web services-based application for LMS data extraction and processing for social network analysis. In: New information and communication technologies for knowledge management in organizations. Springer International Publishing, Berlin, pp 110–121
81. Hernández-García Á, Suárez-Navas I (2016) GraphFES: a web service and application for moodle message board social graph extraction. In: Big data and learning analytics in higher education. Springer International Publishing, Berlin, pp 167–194
82. Saqr M, López-Pernas S (2021) The longitudinal trajectories of online engagement over a full program. Comput Educ 175:104325. https://doi.org/10.1016/j.compedu.2021.104325
83. Holzer J, Lüftenegger M, Korlat S, Pelikan E, Salmela-Aro K, Spiel C, Schober B (2021) Higher education in times of COVID-19: University students' basic need satisfaction, self-regulated learning, and well-being. AERA Open 7:233285842110031. https://doi.org/10.1177/23328584211003164
84. Becker D, King TD, McMullen B (2015) Big data, big data quality problem. In: 2015 IEEE international conference on big data (big data), pp 2644–2653
85. Klašnja-Milićević A, Ivanović M, Budimac Z (2017) Data science in education: big data and learning analytics. Comput Appl Eng Educ 25:1066–1078. https://doi.org/10.1002/cae.21844
86. Dietze S, Siemens G, Taibi D, Drachsler H (2016) Editorial: datasets for learning analytics. J Learn Anal 3:307–311. https://doi.org/10.18608/jla.2016.32.15
87. Mihaescu MC, Popescu PS (2021) Review on publicly available datasets for educational data mining. WIREs Data Min Knowl Discov 11. https://doi.org/10.1002/widm.1403

Getting Started with R for Education Research

Santtu Tikka, Juho Kopra, Merja Heinäniemi, Sonsoles López-Pernas, and Mohammed Saqr

1 Introduction

R is a free, versatile, and open source programming language and software environment specifically designed for statistical computing and data analysis. R has a vast library of packages that enable data manipulation, visualization, modeling and machine learning [1]. Such a wealth of packages enable a wide range of functionalities that saves the users time, effort or the need to write complex code. The fact that R is freely available makes all these functionalities accessible to all. R has a large community of developers, users and researchers who support the development of the platforms as well as provide support and shared knowledge on popular sites such as StackExchange. Thereupon, R is becoming an increasingly popular choice for students, researchers and data scientists [2].

Being open source and accessible to researchers, several packages are added continuously to expand the possibilities and functions that R offers. Some of the R packages included in this book have been added by researchers during the last few years to address contemporary scientific problems and state-of-the-art innovations [3]. For example, R software packages for the analysis of psychological networks were developed in the past five years and ever since have grown tremendously due to contributions from a large base of researchers [4].

S. Tikka (✉)
Department of Mathematics and Statistics, University of Jyväskylä, Jyväskylä, Finland
e-mail: santtu.tikka@jyu.fi

J. Kopra · S. López-Pernas · M. Saqr
School of Computing, University of Eastern Finland, Joensuu, Finland

M. Heinäniemi
Institute of Biomedicine, University of Eastern Finland, Kuopio, Finland

M. Saqr, S. López-Pernas (eds.), *Learning Analytics Methods and Tutorials*,
https://doi.org/10.1007/978-3-031-54464-4_3

Although many of the methods described in this book can be implemented with other software tools, it is hard to find a comprehensive platform that can be used to perform practically all the existing learning analytics methods with such maturity, performance and range of possibilities. For instance, Social Network Analysis (SNA) can be performed with several programming languages (e.g., Python) and desktop software applications (e.g., Gephi) [5]. However, both options provide limited capabilities compared to what R provides for the analysis of SNA. This includes a wider range of SNA centrality measures, mathematical models, community finding algorithms and generative models. Sequence analysis is another example in which the possibilities offered by R are hard to match with other software solutions.

This book does not make any assumptions about the superiority of R over any other platform. Other languages and software platforms are indeed very helpful and have vast capabilities for researchers. For instance, Python has remarkable tools for machine learning and Gephi offers beautiful graphics for the visualization of networks. Oftentimes, readers may need to learn or use other tools to do specific tasks. Put another way, where R offers a rich toolset for researchers, there is a space for other tools that researchers can use to accomplish certain tasks. In summary, investing time in learning R is a worthwhile endeavor that helps interested researchers to perform and expand their research skills and toolset. Since R is a large platform, it represents a doorway to the vast capabilities of its ever expanding repertoire of functions and packages.

2 Learning R

The goal of programming is to write, i.e. code, a program that performs a desired task. A program consists of several commands, each of which does something very simple. In the statistics and data analysis context, R is typically used to write short programs called scripts. R is therefore not intended for developing games or other complicated programs. R is also not a language originally intended for web programming, although with the right packages you can also make web applications with R.

R is a high-level programming language. This means that there are many ready-made commands in R, which have much more code "underneath" that the R programmer does not have to touch. For example, a statistical t-test requires several mathematical intermediate steps, but an R programmer can perform the test with a single command (`t.test`) that provides all the necessary computations and information about the test.

The best way to learn how to use and program R code is by doing. This text has R code embedded between the text in gray boxes, as in the example below. Lines starting with two hashes, i.e. `##`, are not code, but output generated by running the code. Let's first take the classic "Hello, world!" command as an example:

```
print("Hello, world!")
```

```
[1] "Hello, world!"
```

The print function prints the given text to the console. It is convenient, for example, for testing the operation of a program and monitoring the progress of a longer program. R can also be used as a calculator. In the example below, we calculate the price of a product after a 35% discount that was originally priced at 80 euros.

```
80 * (1 - 0.35)
```

```
[1] 52
```

However, running individual commands is usually not useful unless the results can be saved somewhere. In programming languages, data is stored in variables, which you will become familiar with later.

3 RStudio

R has a large array of tools and integrated development environments (IDEs) that make writing and managing code easier and more accessible. The most widely used R IDE is RStudio, which is a free open source software that—similarly to R—runs on all major operating systems [6]. RStudio provides a comprehensive and user-friendly interface for writing, running, and debugging R code, which makes it easier for users to get started and become more productive. Together, R and RStudio allow for the creation of reproducible research. The code and results can be easily shared and replicated, making R and RStudio great tools for collaboration and transparency.

R and RStudio can be very useful in analyzing and visualizing different types of data. This can help researchers, educators and administrators make data-driven decisions and improve the learning experience for students. Whether you are analyzing student performance, demographic data, or tracking the effectiveness of instructional interventions, R and RStudio provide a flexible and efficient platform for achieving your goal.

First install R (step 1) and then RStudio Desktop for your operating system. R and RStudio are available for many operating systems. The interface of RStudio shown in Fig. 1 has the following default components:

(1) *Editor*: The editor is used to write files containing R code, i.e., R scripts. Scripts will be introduced later, but they are simply a collection of R commands that carry out a specific task when placed together, for example analyze the data of

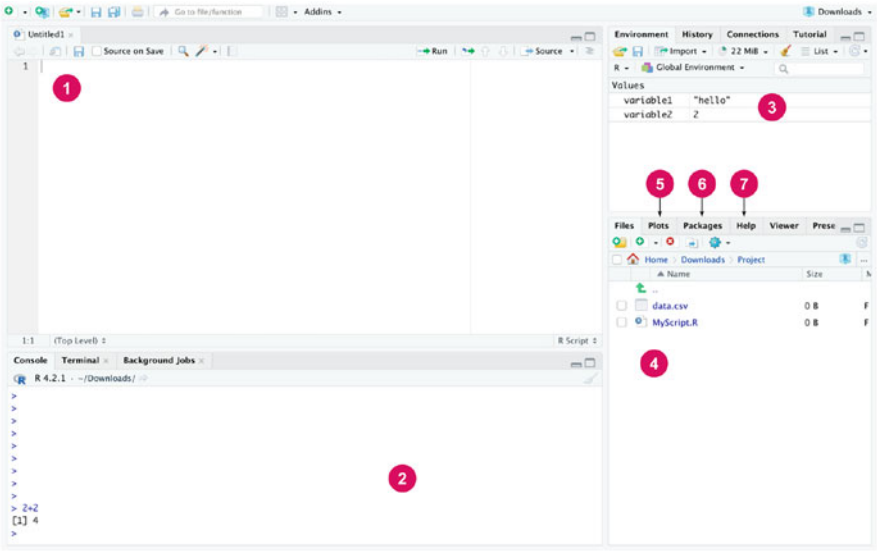


Fig. 1 RStudio user interface

a research project or draw figures of the finished results. A new script can be opened from the "File" menu by selecting "New File" and then "R script". The same can be accomplished via a keyboard shortcut by pressing Ctrl + Shift + N on Windows and Linux. On macOS, users can use Cmd + Shift + N. Also, see the "Keyboard Shortcuts Help" under the "Tools" menu for all the shortcuts available. The code written in the editor can be run line by line by pressing Ctrl + Enter at the line. Several lines can also be selected and run at once. The Source button at the top runs all the code in the current file. R scripts can be saved just like other files and their file extension is .R. All the code you use to analyze your data should be written in scripts. When you save your code in this way, the next time you can simply run the script to perform the desired task instead of rewriting the code from scratch.

(2) *Console*: R commands are executed in the console. If the code written in the editor is run, RStudio automatically executes the commands in the console. In the console, just pressing Enter is enough to execute a line of code. You can try to write a calculation in the console, such as 2 * 3 and press Enter, and the result will be printed in the console. You can also write code in the editor and press Ctrl + Enter to accomplish the same result. Possible messages, warnings and errors are also printed into the console. The main difference between the console and the editor is that the commands written in the console are not saved in any file. So if you want to keep your code, it should be written in the editor and saved in a script file. Commands made during the same session can be scrolled in the console with the up and down arrows. In addition, the command history can be viewed in the History tab in RStudio.

(3) *Workspace*: Displays the variables in the current working environment of the R session.
(4) *Files*: Shows the directory structure of the operating system, by default the working directory.
(5) *Plots*: Graphics drawn with R appear here.
(6) *Packages*: Here you can manage the installed packages (instructions for installing the packages are below).
(7) *Help*: Here you can browse the R manual with instructions for every R command. You can try running the `?print` command in the editor or console, which opens the help page for the `print` function.

4 Best Practices in Programming

As the word "script" already suggests, data analysis often requires a "script" of various steps, such as reading measurements from a file, organizing a table, or describing the results of an analysis, for example by calculating the averages of different sample groups and drawing figures. Writing the R commands that perform these steps in a script provides a documentation of how the analysis was done, and it's also easy to come back to and possibly add a few additional steps, such as performing a t-test. The script file is also easy to share with others who work with similar data analysis tasks, because with small modifications (file names, table structure) the same code also works for different datasets and workflows. Comments are often used inside the code. Comments are text written along the R commands, which are not written in a programming language, and which are ignored when running the code. The purpose of comments is to describe the behavior and purpose of the code. It is good to practice to comment your own code from the beginning, even if the code for the first tasks is very simple. In R, comments are marked with the # symbol.

```
# Assign arbitrary numbers to two variables
x <- 3
y <- 5
# Sum of two variables
z <- x + y
# Print the results
z
```

```
[1] 8
```

4.1 R Markdown

While scripts can only store code and comments, a more comprehensive format called R Markdown is also available in RStudio. R Markdown is an extension of the Markdown markup language that allows users to create dynamic reports and interactive notebooks that can integrate text, code, and visualizations. R Markdown documents are created in a plain text format and can be rendered into various output formats, such as HTML, PDF, Word, or even presentations. To create a new R Markdown document in RStudio, go to the "File" menu, select "New File" and finally "R Markdown". In the dialog that opens, choose the output format you want to use, and give your document a title. Figure 2 shows the RStudio editor panel for a default R Markdown document of R Studio.

The YAML metadata wrapped between the `---` delimiters at the top of the document can be used to customize the output and contents in various ways. For example, you can change the title, author, or add a table of contents. In this example we have defined the title, the output format (HTML) and the date, but many more options are available besides these. Use Markdown syntax to format your text, and enclose your R code in code chunks with the ```` ```{r} ```` and ```` ``` ```` tags which can also be provided other options. For example, our first code chunk has been given the label `setup` and the following option `include = FALSE` means that this chunk will be executed but its output will not be printed into the final document. Within the chunk, a global `knitr` option is set so that all code chunks will print (echo) their

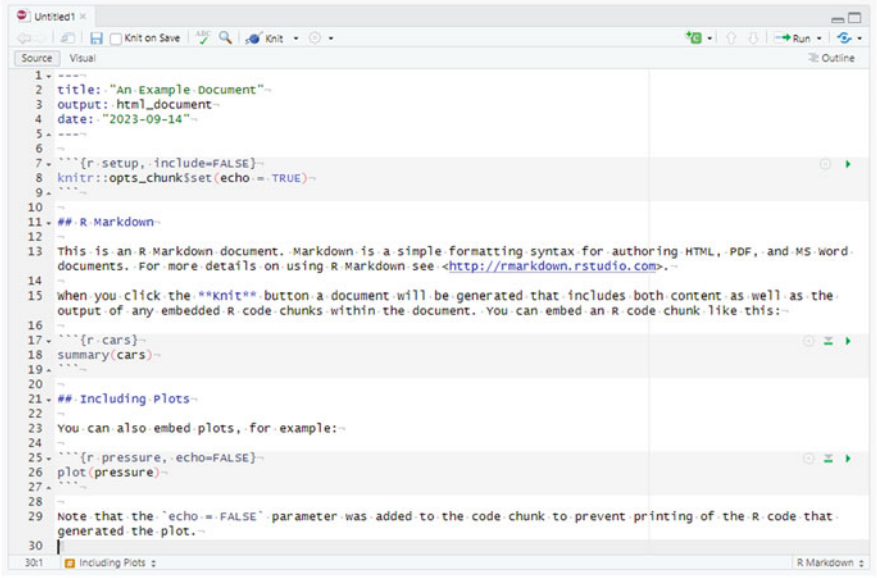


Fig. 2 An example R Markdown document in R Studio

output by default. You can also use inline chunks to quickly reference the results of computations or other R objects. Some examples of standard markdown syntax in the document are second level headers marked with ## (note that # is not used for comments in an R Markdown document) and bold text denoted by wrapping the text with **. Once you are satisfied with your document, you can render it into the chosen output format by clicking the "Knit" button in RStudio, or use the keyboard shortcut "Ctrl + Shift + K". Figure 3 shows the corresponding rendered HTML document.

This simple example shows only a fraction of the full features of R Markdown documents and the Markdown syntax. R Markdown is a powerful tool for creating reproducible research reports, teaching materials, or even websites. It allows users to integrate code and output seamlessly into their written work, making it easier to share and reproduce analyses. As an alternative to R Markdown documents, R Studio also supports the creation of R Notebooks, which are in essence interactive R Markdown documents. R Notebooks can be useful for example when the goal is not to produce one comprehensive analysis report but instead to keep track of the code and try out various approaches to a problem interactively. For an in-depth guide to R Markdown, see [7] which is also freely available online at https://bookdown.org/yihui/rmarkdown/.

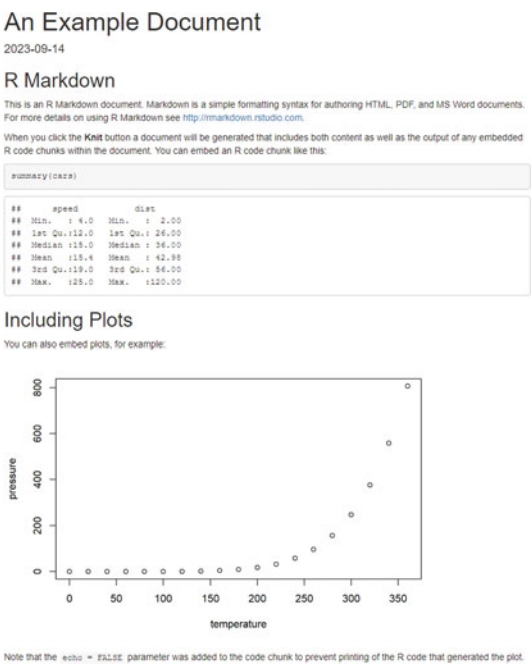


Fig. 3 The example R Markdown document rendered into HTML

4.2 How Is Code Developed?

Code development typically follows similar steps:

1. Design parts of the code.
2. Start by writing a small piece of code.
3. Test whether the code you wrote works. If it doesn't, find out why and fix it.
4. Go to the next piece of code and continue accordingly, always testing piece by piece whether your code works.

Along with this material, many packages include a "Cheat Sheet" as a summary of basic tasks and functions related to the package. Cheat sheets provide a quick and easy reference for checking how something is done in R if you don't remember it by heart. There are cheat sheets for various R packages and other entities on the Internet e.g., the base R cheat sheet, or the tidyr [8] cheat sheet.

In addition to the basic commands presented in this chapter of the book, practical R programming relies to a great extent on the use of various packages developed by the scientific community. Packages are collections of code that contain new functions, classes and data, i.e., they extend R. Most R packages are available from the Comprehensive R Archive Network (CRAN). They can be installed with the `install.packages()` function, or via RStudio's installation window which in practice calls the `install.packages()` function. You can also install several packages at once. The command below installs the `dplyr` [9] and `tidyr` packages:

```
install.packages(c("dplyr", "tidyr"))
```

In order to use the commands contained in an R package, the package must be installed and attached to the R workspace. This is done with the `library()` command:

```
library("tidyr")
```

Now that the `tidyr` package is loaded, we can use the commands it provides, for example to manage the learning analytics data in data frame format that we will address later. If you don't want to attach the entire package, you can use individual commands from packages with the format `name_of_the_package::name_of_the_command()`.

5 Basic Operations

Basic operations in R consist of arithmetic operations, logical operations, and assignment. In addition, there are several commands that are often helpful when

starting a new project or managing the working directory. For example, the current working directory can be obtained with the following command:

```
getwd()
```

```
[1] "/home/sonsoles/labook/chapters/ch03-intro-r"
```

5.1 Arithmetic Operators

R can be used to compute basic arithmetic operations such as addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (^). These operations follow standard precedence rules, and additional brackets can be added to control the evaluation order if needed.

```
1 + 1 # Addition
```

```
[1] 2
```

```
2 - 1 # Subtraction
```

```
[1] 1
```

```
2 * 4 # Multiplication
```

```
[1] 8
```

```
5 / 2 # Division
```

```
[1] 2.5
```

```
2 ^ 4 # Exponentiation
```

```
[1] 16
```

5.2 Relational Operators

Relational operators compare objects or values to other objects or values. These operators are often required for conditional data filtering, for example when selecting a subset of individuals that satisfy some criterion. In R, there are six such operators: smaller than, greater than, smaller or equal to, greater or equal to, equal to, and not equal to. There operators have the following syntax in R:

```
1 < 2  # Smaller than
```

```
[1] TRUE
```

```
3 > 2  # Greater than
```

```
[1] TRUE
```

```
2 <= 2 # Smaller or equal to
```

```
[1] TRUE
```

```
3 >= 3 # Greater or equal to
```

```
[1] TRUE
```

```
5 == 5 # Equal to
```

```
[1] TRUE
```

```
1 != 2 # Equal to
```

```
[1] TRUE
```

In the previous example we used these operators to compare integers, but we may also use them to compare other types of values, such as characters:

```
"a" == "b"
```

```
[1] FALSE
```

5.3 Logical Operators

Similar to relational operators, logical operators are used to evaluate the logical value of a conjunction of two logical values. In R, there are five logical operators: negation, AND, OR, elementwise AND, and elementwise OR. These operators have the following syntax:

```
!TRUE          # Negation
```

```
[1] FALSE
```

```
TRUE && TRUE  # Logical AND
```

```
[1] TRUE
```

```
TRUE || FALSE # Logical OR
```

```
[1] TRUE
```

```
TRUE & TRUE   # Elementwise AND
```

```
[1] TRUE
```

```
TRUE | FALSE  # Elementwise OR
```

```
[1] TRUE
```

The elementwise operators `&` and `|` can be used to compare multiple pairs of logical values simultaneously, for example

```
c(TRUE, FALSE, TRUE) | c(TRUE, FALSE, FALSE)
```

```
[1]  TRUE FALSE  TRUE
```

whereas the operators `&&` and `||` only accept single values. In the previous example, we also used one of the most important operations of R: the `c()` function (the letter 'c' is short for "combine") which we used to combine the logical values into a vector, i.e., an ordered sequence of values of the same type. Vectors and other important data types will be discussed in greater details in the next section.

5.4 Special Operators

Another core functionality in R is the assignment operator <-. Assignment can be used to store the results of computations into variables which can then be used again in other computations without having to redo the original computations. For instance

```
x <- 5 # Assign value 5 into variable named x
y <- 7 # Assign value 7 into variable named y
x      # Access value of x (value is printed into the
                            console in RStudio)
```

```
[1] 5
```

```
y      # Access value of y
```

```
[1] 7
```

```
x + y  # Compute the sum of x and y
```

```
[1] 12
```

```
x > y  # Is x greater than y?
```

```
[1] FALSE
```

Here we chose the names x and y for our variables, but the names are arbitrary with the caveat that one should avoid assigning values to objects with names that R already uses internally, such as names of common functions like c(), exp(), or lm() to name a few. Variables currently assigned in the working environment can be displayed with the following command:

```
ls()
```

```
[1] "x" "y" "z"
```

It is also possible to use the equal sign = as the assignment operator, but this is often not recommended, because the equal sign also has other purposes in the R language and may cause confusion if used for assignment. There are also some special instances, where the equals sign does not function identically to <-. Therefore, we recommend always using the standard assignment operator <-.

When constructing vectors, the function `c()` can be cumbersome in some scenarios. For instance, say we wanted to create a vector that contains all integers from 1 to 100. With `c()`, we would have to write each value individually. Fortunately, such sequences can be constructed effortlessly using the `:` (colon) operator:

```
1:100
```

```
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
 [20]  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
 [39]  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57
 [58]  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76
 [77]  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95
 [96]  96  97  98  99 100
```

Often we may wish to select only specific values from a vector. Values in a vector can be accessed by their index, starting from 1. This is accomplished by using the subset operator which uses the following square bracket syntax. For example, we could select the first value of a vector `x` by writing `x [1]`. For a more involved example, we could simultaneously select the first, 50th and 100th value of the vector `1:100` by writing:

```
x <- 1:100
x[c(1, 50, 100)]
```

```
[1]   1  50 100
```

Essentially, we write the positions of the values that we wish to select inside the square brackets as a vector. Alternatively, we can create a vector of logical values that has the same length as the vector we are selecting from, and the value `TRUE` for the values we wish to select, and `FALSE` otherwise. In the next section, we also show how to select values based on a condition.

6 Basic Data Types and Variables

In the examples of the previous section, we already used several of the most common data types that users are most likely to encounter in practical data analyses. Each object in R has a type, which can be determined with the function `typeof()`. Types are used to describe what kind of data our variables of interest contain and what kind of operations can be carried out on them. Perhaps the most common type is the `numeric` type, which describes values that can be interpreted as numbers. Two special instances of the `numeric` type are `integer` and `double`, which correspond to integer values and decimal values, respectively.

```
typeof(1.0)
```

```
[1] "double"
```

```
typeof(1L)
```

```
[1] "integer"
```

The capital `L` on the second line denotes that we mean the integer 1, and not the decimal number 1.0. Text data is often represented by the `character` type. Unlike in some other programming languages, in R the `character` type does not necessarily describe individual characters as the name would suggest, but character strings, for instance:

```
typeof("a")
```

```
[1] "character"
```

```
typeof("hello world!")
```

```
[1] "character"
```

As we can see, both have the same type.

The `logical` type is used to represent the boolean values `TRUE` and `FALSE`. Logical values are typically not as common in actual data where such values may often be represented by the integers 1 and 0 instead. However, their importance lies in forming conditions for data filtering and manipulations that we may wish to carry out based on a criterion that is true for only a subset of subjects in our data. For example, suppose we have the values 1, 2, 3, 4, 5, and we wish to programmatically select only those values that are greater than 2. We can accomplish this as follows:

```
some_numbers <- 1:5
some_numbers[some_numbers > 2]
```

```
[1] 3 4 5
```

Let's walk through what the code above does step by step. On the first line, we create a vector of numeric values 1 through 5 by using the `:` operator and assign the result to the variable named `some_numbers`. On the second line, we use the square brackets (the subset operator) to select only those values of `some_numbers`

for which the condition `some_numbers > 2` is true. Here we introduce the vectorization feature of R which applies to a vast majority of arithmetic and relational operators. By writing `some_numbers > 2`, we actually evaluate the condition for every value in the vector `some_numbers`:

```
some_numbers > 2
```

```
[1] FALSE FALSE  TRUE  TRUE  TRUE
```

All of the aforementioned types are called atomic types, meaning that vectors of such values can only contain values of one specific atomic type. For example, a vector cannot contain values of both character and integer types:

```
c(0L, "a")
```

```
[1] "0" "a"
```

We see that the result was automatically converted to an atomic character vector.

Alongside type, objects in R also have a class, which can be viewed with the `class()` function. For atomic types, their class is the same as their type, but for more complicated types of variables, a class essentially describes a special instance of a type. Objects within a specific class typically have their own set of functions or methods that can only be applied for that specific class. A common example of a class is `factor`. Factors are a special class of `integer` vectors designed to represent categorical and ordinal data. Variables that are of the `factor` class have levels, which differentiates `factor` variables from ordinary `integer` variables. For example, a factor could represent group membership in a randomized experiment indicating inclusion in the control group or the treatment group for each individual. The levels of this factor could be called `"control"` and `"treatment"` for example. Supposing that 0 indicates that an individual belongs in the control group and 1 indicates that an individual belongs in the treatment group, we could create such a factor by writing

```
group <- c(0, 0, 1, 0, 1, 0, 1, 1)
factor(group, labels = c("control", "treatment"))
```

```
[1] control   control   treatment control   treatment control   treatment treatment
Levels: control treatment
```

Factors are important when fitting statistical models or when performing statistical tests, because if we would simply use the corresponding integer values, they may be erroneously interpreted as continuous. The levels of the factor also often help to provide more informative output. In the next section we will discuss more complicated data types including data frames, which can contain data of various types simultaneously.

7 Basic R Objects

In this section we will discuss the concepts of data frame and tibble. Let's assume that your data can be stored in a two-dimensional array where the columns represent variables and each row represents a case of measurement. In R, a data frame is a concept for storing such a two-dimensional array of data. Let's first study how data frames work in R and we will then move on to see how another concept called `tibble` extends the capabilities of a data frame.

A data frame which has been loaded into R under the name `grades` and printed in the R console will look as follows.

```
grades
```

```
  group grade
1     2  4.67
2     2  4.90
3     3  2.63
4     4  3.39
5     4  6.89
```

In the above printout we can see that R prints the data frame just as we would expect the data to look like. One issue with a standard data frame is that if there are very many columns or rows, then the printout may be difficult to read. In RStudio, a neater (and more flexible) way of inspecting the data is by using a command called `View()`:

```
View(grades)
```

To go further with the data, we need tools for data manipulation. We begin with the very basic tools which are commonplace in any data analysis workflow. More comprehensive knowledge about data transforming, cleaning etc. can be found in Chap. 2.

A data frame can be constructed directly in the R by using the function called `data.frame()` and then listing variable names and their values. The `grades` data above has two columns, `group` and `grade`, and each row represents a student. The variable `group` is indicates the number of a group in which each person has studied. The variable `grade` stores the final grade that the student has received from a course.

8 Working with Dataframes

To extract a column from a data frame, one needs to start with the name of the data frame object and connect the column name with the name of the data frame object

by using the dollar symbol ($). Thus, extracting the column `group` from `grades` data can be accomplished by writing

```
grades$group
```

```
[1] 2 2 3 4 4
```

Next, let's use a couple of basic functions which are often needed when developing R code. First of all, to get a summary of every variable of a data frame, we can call

```
summary(grades)
```

```
     group       grade
 Min.   :2   Min.   :2.630
 1st Qu.:2   1st Qu.:3.390
 Median :3   Median :4.670
 Mean   :3   Mean   :4.496
 3rd Qu.:4   3rd Qu.:4.900
 Max.   :4   Max.   :6.890
```

The result show us the minimum, maximum, median, mean and quartiles of both variables. You may note that as `group` was intended to be a categorical variable, so computing its mean value in the data does not make sense. We can change that behavior by converting the `group` column into a factor.

```
grades$group <- as.factor(grades$group)
summary(grades)
```

```
 group     grade
 2:2   Min.   :2.630
 3:1   1st Qu.:3.390
 4:2   Median :4.670
       Mean   :4.496
       3rd Qu.:4.900
       Max.   :6.890
```

On the other hand, we might want to calculate the mean or the sample standard deviation of the variable `grade`. This can be done with functions called `mean()` and `sd()`, respectively.

```
mean(grades$grade)
```

```
[1] 4.496
```

```
sd(grades$grade)
```

```
[1] 1.630178
```

The functions above can only take numeric vectors as input. If we tried using another type of argument, we would encounter an error message.

```
sd(grades)
```

```
Error in is.data.frame(x): 'list' object cannot be coerced to type 'double'
```

The above message can actually teach us a couple of things. First of all, the error comes from the function `is.data.frame()`, which is called somewhere in the definition of `sd()`. Second, the actual error message tells us that the object which we gave to the function as its argument is a `list` object. List is an object type of R on top of which `data.frame` objects have been built. We will describe lists in greater detail later. Further, the message tells us that R has tried to coerce the argument object into the `double` type. This means that the object we supplied to the function is not of the right type and cannot easily be converted into the proper format.

8.1 tibble

A `tibble` is an expansion of `data.frame` objects which is used in the tidyverse programming paradigm [10]. To use tidyverse, we advise to load the `tidyverse` [11] (meta)package which loads all key tidyverse packages.

```
# load tidyverse to use as_tibble
library("tidyverse")
# convert a data frame as tibble
grades2 <- as_tibble(grades)
```

Next, let's see what a `tibble` looks like when printed in the console

```
grades2
```

```
# A tibble: 5 x 2
  group grade
  <fct> <dbl>
1 2      4.67
2 2      4.9
3 3      2.63
4 4      3.39
5 4      6.89
```

Tibbles behave similarly compared to data frames when printed, but they also describe the dimensions of the data and the types of the columns right under the column names. For instance, `<fct>` refers to a `factor` column and `<dbl>` refers to a `double` column. In order to discover the column types when using data frames, one would need to apply the `class()` or `typeof()` function to the columns, or write `str(grades)` to see the types of the columns and the structure of the data. Another useful property of `tibble` tables is that if a tibble has a large number of observations or variables, then only the rows or the columns which can fit on to the screen are printed.

Tidyverse and tibbles also support so called lazy evaluation, which is useful when your data is stored in a database, for instance. With lazy evaluation, the commands that you use on your data would be evaluated directly in the database (if possible). Without lazy evaluation, the entire data would be downloaded onto your computer only after which the commands would be evaluated. Lazy evaluation can perform many tasks faster and it can also alleviate memory usage of the computer.

9 Pipes

Piping is a fairly recent concept in R and was very rarely used in R code just a few years ago. The concept of a pipe originates from a package called `magrittr`, and pipes are commonly used under the tidyverse programming paradigm, but the pipe was also later added to base R. The notations for a tidyverse pipe and a native R pipe are `%>%` and `|>`, respectively. The idea of a pipe is that you can connect multiple function calls sequentially while keeping the code more readable. Pipes also serve to unnest standard R code which often involves using many nested parentheses, and can quickly become hard to read as one has to read the code based on the order of the operations instead of reading it linearly. For example, consider the following code where we apply a sequence of operations on a numeric vector `x`.

```
x <- 1:10
round(mean(diff(log(x))), digits = 2)
```

```
[1] 0.26
```

This code computes the rounded mean differences of the logarithms of the vector `x`, however this description does not match the order of operations, where the logarithm is computed first. To accomplish the same result using pipes we would write

```
x <- 1:10
x |> log() |> diff() |> mean() |> round(digits = 2)
```

```
[1] 0.26
```

Here, the order of operations can be easily read from left to right. Next, we will discuss the use of pipes in more detail.

9.1 magrittr pipe %>%

Let's have a look at an example, where we call the `summary()` function for the `grades2` data using the magrittr pipe `%>%`, and we also define that results should be printed with two digits.

```
grades2 %>%
  summary(digits = 2)
```

```
 group     grade
 2:2   Min.   :2.6
 3:1   1st Qu.:3.4
 4:2   Median :4.7
       Mean   :4.5
       3rd Qu.:4.9
       Max.   :6.9
```

In the code above, the object `grades2` is taken by the pipe operator `%>%` and forwarded to the first argument of the `summary()` function. The `summary()` function also has a second argument, which is defined by `digits = 2`. Thus, the pipe only takes the object mentioned before the pipe operator and forwards it to the function after the pipe as the first free argument. It is very common and

recommended to structure R code so that there is only one pipe per row and that a new line is started after each pipe.

Although the above example is easy to understand as we already know the `summary()` function, there is also a more general way to compute summarized information following the tidyverse style. The function `summarise()` can be used to compute arbitrary statistics from the data, for example the number of observations (via the function `n()`) and the mean and the sample standard deviation of the variable `grade`.

```
grades2 %>%
  summarise(
    n = n(),
    mean = mean(grade),
    sd = sd(grade)
  )
```

```
# A tibble: 1 x 3
      n  mean    sd
  <int> <dbl> <dbl>
1     5  4.50  1.63
```

The `summarise()` function also produces a `tibble` enabling further operations via piping, if desired.

9.2 *Native pipe |>*

In R version 4.1, the native pipe `|>` was introduced to the R language, which does not require any external packages to use. In most scenarios, it does not matter whether the native or the `magrittr` pipe is used. However, there are two technical differences between the `magrittr` pipe and the native pipe. First, the `magrittr` pipe is actually a function

```
class(`%>%`)
```

```
[1] "function"
```

while the native pipe is not, and simply converts the written code into a non-piped version, i.e., into a form that one would write without using the pipe:

```
x <- 1:5
quote(x |> sum())
```

```
sum(x)
```

We see that providing `x` to the `sum` function via the native pipe is exactly the same as writing `sum(x)` directly. What this means in practice is that the native pipe may have better performance for example when passing a large dataset through a large number of pipes. The reason for this is that the `magrittr` pipe incurs an additional computational function call overhead each time it is called. The second difference between the pipes is that parentheses have to be provided for function calls when using the native pipe, but they can be omitted when using the `magrittr` pipe

```
x %>% sum
```

```
[1] 15
```

```
x %>% sum()
```

```
[1] 15
```

```
x |> sum()
```

```
[1] 15
```

```
x |> sum # produces an error
```

```
Error: The pipe operator requires a function call
         as RHS (<text>:1:6)
```

10 Lists

Earlier, we already briefly mentioned lists in the context of data frames. Lists are one the most common types of data in R and they resemble basic vectors in many aspects. Like vectors, a list is an ordered sequence of elements, but unlike vectors, lists can contain elements of different types simultaneously and may even contain other lists. For example, we could construct a list that contains a `logical` value, a `numeric` value and a `character` value using the `list()` function as follows.

```
y <- list(TRUE, 7.2, "this is a list")
```

Subsetting a `list` object works slightly differently compared to vectors. When single brackets are used, a sublist is selected, i.e., a `list` object that contains the elements at the supplied indices, for example:

```
y[1:2]
```

```
[[1]]
[1] TRUE

[[2]]
[1] 7.2
```

```
typeof(y[1:2])
```

```
[1] "list"
```

To extract an actual element of a list, double brackets should be used:

```
y[[2]]
```

```
[1] 7.2
```

The elements of a list may also be named, which enables subsetting via the dollar sign operator similar to data frames, or by giving the element name in double brackets instead of the index:

```
z <- list(bool = TRUE, num = 7.2,
                        description = "this is another list")
z$bool
```

```
[1] TRUE
```

```
z[["description"]]
```

```
[1] "this is another list"
```

One benefit of using the dollar sign is that it is not necessary to provide the full element name, unlike when using the double brackets. It is sufficient to provide a prefix of the element name so that the full name can be uniquely determined from the prefix. Because all the names of our elements in the previous list `z` start with a different letter, the first letter of the name suffices as the prefix. The same functionality also applies when using the dollar sign to select columns of data frames.

```
z$b
```

```
[1] TRUE
```

```
z$n
```

```
[1] 7.2
```

```
z$d
```

```
[1] "this is another list"
```

11 Functions

A function is a set of statements that when organized together perform a specific task. Each function in R has a name, a set of arguments, a body and a return object. The name of the function usually describes the purpose of the function. For example, the base R function `mean()` computes the arithmetic mean of the argument vector. The result is returned as a numeric value.

```
x <- 1:5
mean(x)
```

```
[1] 3
```

Functions are often much more complicated, which is why it is often helpful to view the documentation of a function before using it in practice. To view the documentation pages of a function, one can simply write the name of the function prefixed by a question mark.

```
?mean
```

In RStudio, the documentation will open in the "Help" tab in the bottom right pane by default. Functions will only be executed when they are called, i.e., when arguments are supplied to them. Simply writing the function name without parentheses will instead print the body of the function to the console, meaning the code that the function consists of and which is executed if the function is called.

In the previous sections, we've already familiarized ourselves with some commonly used basic functions such as `c()`, `sd()`, and `summary()`. Base R has a wide range of function to accomplish common tasks needed in data analysis, which is further extended by `tidyverse` and other R packages. This means that one does not typically have to write their own functions when programming in R. We will explore several of the functions provided by the `tidyverse` in later chapters.

12 Conditional Statements

Sometimes we may only wish to execute a piece of code when a certain condition is met. Conditional statements in R can be defined via the `if` and `else` clauses. The `if` clause evaluates a condition, which is an R expression that evaluates to a single `logical` value, and if this condition evaluates to `TRUE`, the expression following the clause if executed. Further, if an `else` clause is also provided, the expression following `else` will be executed instead if the condition evaluates to `FALSE`. As R code, the syntax for these clauses is

```
if (cond) expr
if (cond) expr else alt_expr
```

where `cond` is the condition being evaluated, `expr` is the expression that will be evaluated if `cond == TRUE`, and `alt_expr` will be evaluated if `cond == FALSE`.

Note that `if` will only evaluate a single condition. If `cond` is a vector, an error will be produced:

```
cond <- c(TRUE, FALSE)
if (cond) {
  print("This will not be printed")
}
```

```
Error in if (cond) {: the condition has length > 1
```

As expected, the error message tells us that the condition contained more than one element when it was evaluated. However, there are often scenarios where we

may wish to conditionally select or define values based on a vector of conditions. For such instances, the function `ifelse()` can be used. This function has three arguments: `test`, `yes`, and `no`. When called, the function will pick those elements of the vector `yes` for which the logical vector `test` evaluates to `TRUE`, and those elements of the vector `no` for which `test` evaluates to `FALSE`:

```
cond <- c(TRUE, FALSE, FALSE, TRUE)
x <- 1:4
y <- -(1:4)
ifelse(cond, x, y)
```

```
[1]  1 -2 -3  4
```

13 Looping Constructs

In some cases, we may wish to execute the same piece of code multiple times under varying conditions. Instead of writing the same code multiple times for each condition, we can use a looping construct. There are two types of loops in R: the `for` loop and the `while` loop. The main difference between the two loops is that `for` always executes the code associated with the loop a fixed number of times whereas `while` will continue executing the code as long as a specific condition remains satisfied. The syntax of these loops is

```
for (var in seq) expr
while (cond) expr
```

In other words, `for` will execute the expression `expr` for every element `var` in some object `seq` that can be indexed. For-loops are very general, and can be used to loop over most ordered structures such as vectors and lists. Similarly, `while` will execute the expression `expr` as long as the condition `cond` evaluates to `TRUE`. Care must be taken when using while-loops to ensure that the condition will eventually evaluate to `FALSE`, otherwise the loop will simply run indefinitely and the program will be stuck. As an example, we will print the number 1 through 5 to the console using both `for` and `while` loops:

```
x <- 1:5
for (i in x) {
  print(x[i])
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

```
i <- 0
while (i < length(x)) {
  i <- i + 1
  print(x[i])
}
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

In contrast to explicit `for` and `while` loops, the so-called `apply` family of functions can often be a simpler alternative (see `?apply`). As the name suggests, these functions apply an operation to each element of a list or a vector (and other more general data structures). For example, the above loop example could also be accomplished with the `lapply()` function as follows:

```
y <- lapply(x, print)
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

14 Discussion and Other Resources for Learning R

The main aim behind this chapter was to introduce R to new users. This chapter is, of course, an initial step and can hardly cover all the basics. Interested users are advised to consult other resources e.g., open access books, tutorials, cheat sheets, and package manuals for more information. An introductory book "An Introduction to R" packaged with each R installation can be accessed from the RStudio Help panel by first selecting "Show R Help" and then selecting "An Introduction to R" under "Manuals". This book covers a wide range of topics on base R programming

in great detail. The book "R for Data Science" by Hadley Wicham and Garret Grolemund provides a comprehensive tutorial on using R for data science under the tidyverse paradigm. The book is free to use and readily available online at https://r4ds.had.co.nz/. As we go further, several questions will emerge and the reader will learn by doing and by consulting the literature and help files. In doing so, the reader will build knowledge and experience that helps advance their skills.

References

1. R Core Team (2022) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna
2. Wickham H, Grolemund G (2016) R for data science: import, tidy, transform, visualize, and model data. O'Reilly Media, Inc., Sebastopol
3. Wickham H (2015) R packages: organize, test, document, and share your code. O'Reilly Media, Inc., Sebastopol
4. Epskamp S, Fried EI (2018) A tutorial on regularized partial correlation networks. Psychol Methods 23:617–634. https://doi.org/10.1037/met0000167
5. Bastian M, Heymann S, Jacomy M (2009) Gephi: an open source software for exploring and manipulating networks. In: Third international AAAI conference on weblogs and social media, pp 361–362
6. RStudio Team (2020) RStudio: integrated development environment for R. RStudio, PBC, Boston
7. Xie Y, Allaire JJ, Grolemund G (2019) R markdown: the definitive guide, 1st edn. Chapman & Hall/CRC, Boca Raton
8. Wickham H, Vaughan D, Girlich M (2023) tidyr: tidy messy data. https://CRAN.R-project.org/package=tidyr
9. Wickham H, François R, Henry L, Müller K, Vaughan D (2023) dplyr: a grammar of data manipulation. https://CRAN.R-project.org/package=dplyr
10. Müller K, Wickham H (2023) tibble: simple data frames. https://CRAN.R-project.org/package=tibble
11. Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019) Welcome to the tidyverse. J Open Source Softw 4:1686. https://doi.org/10.21105/joss.01686

An R Approach to Data Cleaning and Wrangling for Education Research

Juho Kopra, Santtu Tikka, Merja Heinäniemi, Sonsoles López-Pernas, and Mohammed Saqr

1 Introduction

When analyzing data, it is crucial that the data is in a suitable format for the tools you will be using. This makes data wrangling essential. Data preparation and cleaning, such as extracting information from raw data or removing erroneous measurements, must also be done before data is ready for analysis. Data wrangling often takes up the majority of the time spent on analysis, sometimes up to 80%. To reduce the amount of work required, it is beneficial to use tools that follow the same design paradigm to minimize the time spent on data wrangling. The `tidyverse` [1] programming paradigm is currently the most popular approach for this in R.

The `tidyverse` has several advantages that make it preferable over other alternatives such as simply using base R for your data wrangling needs. All packages in the `tidyverse` follow a consistent syntax, making it intuitive to learn and use new `tidyverse` packages. This consistency also makes the code more easier to read, and maintain, and reduces the risk of errors. The `tidyverse` also has a vast range of readily available packages that are actively maintained, reducing the need for customized code for each new data wrangling task. Further, these packages integrate seamlessly with one another, facilitating a complete data analysis pipeline.

J. Kopra (✉) · S. López-Pernas · M. Saqr
School of Computing, University of Eastern Finland, Kuopio, Finland
e-mail: juho.kopra@uef.fi

S. Tikka
Department of Mathematics and Statistics, University of Jyväskylä, Finland

M. Heinäniemi
Institute of Biomedicine, University of Eastern Finland, Kuopio, Finland

M. Saqr, S. López-Pernas (eds.), *Learning Analytics Methods and Tutorials*,
https://doi.org/10.1007/978-3-031-54464-4_4

To fully realize the benefits of the `tidyverse` programming paradigm, one must first understand the key concepts of tidy data and pivoting. Tidy data follows three rules:

1. Each variable must have its own column.
2. Each observation must have its own row.
3. Each value must have its own cell.

Let's consider examples of tidy data. For instance, if you have data from a Moodle course where two attempts of an exam for each student are located in a single column. This example data violates the first rule, because there are two variables in a single column instead of separate columns for each variable. What is needed to make this data tidy is to pivot it to a longer format. The pivoted data would have two rows for each student, both of which are different observations (exam attempts 1 and 2). Thus the pivoted data would not conflict with second rule.

Data can also be too long, but in practice, this is much more rare. This can occur if two or more variables are stored on a single column across multiple rows. A key indicators of this is if the different rows of the same column have different measurements units (e.g. lb vs. kg). It may also occur that your raw data has multiple values in a single cell. In these cases, it is necessary to split the cells to extract the necessary information. In a simple case, where you have two values systematically in a one cell, the values can be easily separated into their own columns.

Overall, using `tidyverse` and understanding the key concepts of tidy data and pivoting can streamline the data analysis process and make code easier to work with and maintain. The rest of this chapter will guide readers through the process of data cleaning and wrangling with R in the field of learning analytics. We demonstrate how data can be grouped and summarized, how to select and transform variables of interest, and how data can be rearranged, reshaped and joined with other datasets. We will strongly rely on the `tidyverse` programming paradigm for a consistent and coherent approach to data manipulation, with a focus on tidy data.

2 Reading Data into R

Data files come in many formats, and getting your data ready for analysis can often be a daunting task. The `tidyverse` offers much better alternatives to base R functions for reading data, especially in terms of simplicity and speed when reading large files. Additionally, most of the file input functions in the `tidyverse` follow a similar syntax, meaning that the user does not have to master every function for reading every type of data individually.

Often, just before the data can be read into R, user must specify the location of data files by setting a working directory. Perhaps most useful way to do that is to create a project in RStudio and then create a folder called "data" within the project folder. Data files can be put into that folder and user can refer to those files just by telling R-functions relative path of data file (e.g. "data/Final%20Dataset.csv") while the project takes care of the rest of the path. A more traditional way of

setting this, which also works without RStudio, is by using a command such as `setwd("/home/Projects/LAproject/data/Final%20Dataset.csv")`. Here, a function called `setwd()` is used to set up a folder into location mentioned in a character string given as its argument. A `getwd()` lists current working directory, which can also be seen in RStudio just above the Console output.

Some of the most common text data formats are comma-separated files or semicolon-separated files, both of which typically have the file extension .csv. These files can be read into R using the `readr` [2] package and the functions `read_csv()` and `read_csv2()`, respectively. For instance, we can read a comma-separated file R as follows

```
library("readr")
url <- "https://github.com/lamethods/data/raw/main/2_moodleEdify/"
lms <- read_csv(paste0(url, "Final%20Dataset.csv"))
```

Functions in `readr` provide useful information about how the file was read into R, which can be used to assess if the input was successful and what assumptions about the data were made during the process. In the printout above, the `read_csv()` function tells us the number of rows and columns in the data and the column specification, i.e., what type of data is contained within each column. In this case, we have 17 columns with `character` type of data, and 4 columns of `double` type of data. Functions in `readr` try to guess the column specification automatically, but it can also be specified manually when using the function. For more information about this dataset, please refer to Chapter 2 in this book [3].

Data from Excel worksheets can be read into R using the `import()` function from the `rio` [4] package. We will use synthetic data generated based on a real blended course of learning analytics for the remainder of this chapter. These data consist of three Excel files which we will first read into R.

```
library("rio")
url <- "https://github.com/lamethods/data/raw/main/1_moodleLAcourse/"
events <- import(paste0(url, "Events.xlsx"), setclass = "tibble")
results <- import(paste0(url, "Results.xlsx"), setclass = "tibble")
demographics <- import(paste0(url, "Demographics.xlsx"), setclass = "tibble")
```

The data files contain information on students' Moodle events, background information such as their name, study location and employment status, and various grades they've obtained during the course. For more information about the dataset, please refer to Chapter 2 in this book [3]. These data are read in the `tibble` [5] format, a special type of `data.frame` commonly used by `tidyverse` packages. We also load the `dplyr` [6] package which we will use for various tasks throughout this chapter.

```
library("dplyr")
```

3 Grouping and Summarizing Data

Instead of individual-level data metrics, we may be interested in specific groups as specified by some combination of data values. For example, we could compute the number of students studying in each location by gender. To accomplish this, we need to start by creating a grouped dataset with the function `group_by()`. To compute the number of students, we can use the `summarise()` function which we already used previously in Chapter 1 and the function `count()`, which simply returns the number of observations in each category of its argument.

```
demographics |>
  group_by(Gender) |>
  count(Location)
```

```
# A tibble: 4 x 3
# Groups:   Gender [2]
  Gender Location      n
  <chr>  <chr>     <int>
1 F      On campus    55
2 F      Remote       10
3 M      On campus    51
4 M      Remote       14
```

The column `n` lists the number of students in each group. When a `tibble` that contains grouped data is printed into the console, the grouping variable and the number of groups will be displayed in the console below the dimensions. Next, we will compute the total number of Moodle events of each student, which we will also use in the subsequent sections.

```
events_summary <- events |>
  group_by(user) |>
  tally() |>
  rename(Frequency.Total = n)
events_summary
```

```
# A tibble: 130 x 2
  user      Frequency.Total
  <chr>               <int>
1 00a05cc62             417
2 042b07ba1             918
3 046c35846             199
4 05b604102             199
5 0604ff3d3             436
# i 125 more rows
```

Here, the function `tally()` simply counts the number of number rows in the data related to each student, reported in the column `n` which we rename to `Frequency.Total` with the `rename()` function. We could also count the number of events by event type for each student

```
events |>
  group_by(user, Action) |>
  count(Action)
```

```
# A tibble: 1,439 x 3
# Groups:   user, Action [1,439]
  user     Action          n
  <chr>    <chr>       <int>
1 00a05cc62 Applications     2
2 00a05cc62 Assignment     121
3 00a05cc62 Course_view    103
4 00a05cc62 Feedback         7
5 00a05cc62 General          10
# i 1,434 more rows
```

4 Selecting Variables

In the `tidyverse` paradigm, selecting columns, i.e., variables from data is done using the `select()` function. The `select()` function is very versatile, allowing the user to carry out selections ranging from simple selection of a single variable to highly complicated selections based on multiple criteria. The most basic selection selects only a single variable in the data based on its name. For example, we can select the employment statuses of students as follows

```
demographics |>
  select(Employment)
```

```
# A tibble: 130 x 1
  Employment
  <chr>
1 None
2 None
3 None
4 None
5 Part-time
# i 125 more rows
```

Note that using `select()` with a single variable is not the same as using the `$` symbol to select a variable, as the result is still a `tibble`, `select()` simply produces a subset of the data, where only the selected columns are present. Select is more similar to `subset()` in base R, which can accomplish similar tasks as `select()` and `filter()` in the `tidyverse`. However, we do not recommend using `subset()`, as it may not work correctly when the working environment has variables that have the same name as columns in the data, which can lead to undesired outcomes.

To extract the values of the selected column as a vector, we can use the function `pull()`. We use the `head()` function here to limit the console output to just the first few values of the vector (default is 6 values).

```
demographics |>
  pull(Employment) |>
  head()
```

```
[1] "None"      "None"      "None"      "None"      "Part-time" "Part-time"
```

The `select()` function syntax supports several operations that are similar to base R. We can select ranges of consecutive variables using `:`, complements using `!`, and combine selections using `c()`. The following selections illustrate some of these features:

```
demographics |>
  select(user:Origin)
```

```
# A tibble: 130 x 4
  user      Name    Surname  Origin
  <chr>     <chr>   <chr>    <chr>
1 6eba3ff82 Amanda  Mora     Costa Rica
2 05b604102 Lian    Abdullah Yemen
3 111422ee7 Bekim   Krasniqi Kosovo
4 b4658c3a9 Yusuf   Kaya     Turkey
5 e6ec47f29 Zoran   Babić    Serbia
# i 125 more rows
```

```
demographics |>
  select(!Gender)
```

```
# A tibble: 130 x 7
  user      Name    Surname  Origin     Birthdate  Location  Employment
  <chr>     <chr>   <chr>    <chr>      <chr>      <chr>     <chr>
```

```
1 6eba3ff82 Amanda Mora     Costa Rica 28.2.1998  On campus None
2 05b604102 Lian   Abdullah Yemen      19.11.1996 On campus None
3 111422ee7 Bekim  Krasniqi Kosovo     30.1.1999  On campus None
4 b4658c3a9 Yusuf  Kaya     Turkey     16.6.1998  On campus None
5 e6ec47f29 Zoran  Babić    Serbia     29.10.1998 On campus Part-time
# i 125 more rows
```

```
demographics |>
  select(c(user, Surname))
```

```
# A tibble: 130 x 2
  user      Surname
  <chr>     <chr>
1 6eba3ff82 Mora
2 05b604102 Abdullah
3 111422ee7 Krasniqi
4 b4658c3a9 Kaya
5 e6ec47f29 Babić
# i 125 more rows
```

In the first selection, we select all variables starting from `user` on the left to `Origin` on the right. In the second, we select all variables except `Gender`. In the third, we select both `user` and `Surname` variables.

Sometimes, our selection might not be based directly on the variable names themselves as in the examples above but instead on vectors that contain the names of columns we wish to select. In such cases, we can use the function `all_of()`. We can consider the intersections or unions of such selections using `&` and `|`, respectively.

```
cols_a <- c("user", "Name", "Surname")
cols_b <- c("Surname", "Origin")
demographics |>
  select(all_of(cols_a))
```

```
# A tibble: 130 x 3
  user      Name   Surname
  <chr>     <chr>  <chr>
1 6eba3ff82 Amanda Mora
2 05b604102 Lian   Abdullah
3 111422ee7 Bekim  Krasniqi
4 b4658c3a9 Yusuf  Kaya
5 e6ec47f29 Zoran  Babić
# i 125 more rows
```

```
demographics |>
  select(all_of(cols_a) & all_of(cols_b))
```

```
# A tibble: 130 x 1
  Surname
  <chr>
1 Mora
2 Abdullah
3 Krasniqi
4 Kaya
5 Babić
# i 125 more rows
```

```
demographics |>
  select(all_of(cols_a) | all_of(cols_b))
```

```
# A tibble: 130 x 4
  user      Name    Surname  Origin
  <chr>     <chr>   <chr>    <chr>
1 6eba3ff82 Amanda  Mora     Costa Rica
2 05b604102 Lian    Abdullah Yemen
3 111422ee7 Bekim   Krasniqi Kosovo
4 b4658c3a9 Yusuf   Kaya     Turkey
5 e6ec47f29 Zoran   Babić    Serbia
# i 125 more rows
```

Often the names of data variables follow a similar pattern, and these patterns can be used to construct selections. Selections based on a prefix or a suffix in the variable name can be carried out with the functions `starts_with()` and `ends_with()`, respectively. The function `contains()` is used to look for a specific substring in the names of the variables, and more complicated search patterns can be defined with the function `matches()` that uses regular expressions (see `?tidyselect::matches` for further information).

```
results |>
  select(starts_with("Grade"))
```

```
# A tibble: 130 x 13
  Grade.SNA_1 Grade.SNA_2 Grade.Review Grade.Group_self Grade.Group_All
        <dbl>       <dbl>        <dbl>            <dbl>           <dbl>
1           0           0         6.67                5               4
2           8          10         6.67                1               3
```

```
3          10          10         10                  10            9.11
4           5           5          0                   1            4
5          10          10         10                  10            9.18
# i 125 more rows
# i 8 more variables: Grade.Excercises <dbl>, Grade.Project <dbl>,
#   Grade.Literature <dbl>, Grade.Data <dbl>, Grade.Introduction <dbl>,
#   Grade.Theory <dbl>, Grade.Ethics <dbl>, Grade.Critique <dbl>
```

```
results |>
  select(contains("Data"))
```

```
# A tibble: 130 x 1
  Grade.Data
       <dbl>
1          4
2          3
3          5
4          3
5          5
# i 125 more rows
```

So far, our selections have been based on variable names, but other conditions for selection are also feasible. The general-purpose helper function `where()` is used to select those variables for which a function provided to it returns `TRUE`. For example, we could select only those columns that contain `character` type data or `double` type data.

```
results |>
  select(where(is.character))
```

```
# A tibble: 130 x 1
  user
  <chr>
1 6eba3ff82
2 05b604102
3 111422ee7
4 b4658c3a9
5 e6ec47f29
# i 125 more rows
```

```
results |>
  select(where(is.double))
```

```
# A tibble: 130 x 14
  Grade.SNA_1 Grade.SNA_2 Grade.Review Grade.Group_self Grade.Group_All
        <dbl>       <dbl>        <dbl>            <dbl>           <dbl>
1           0           0         6.67                5            4
2           8          10         6.67                1            3
3          10          10        10                  10            9.11
4           5           5         0                   1            4
5          10          10        10                  10            9.18
# i 125 more rows
# i 9 more variables: Grade.Excercises <dbl>, Grade.Project <dbl>,
#   Grade.Literature <dbl>, Grade.Data <dbl>, Grade.Introduction <dbl>,
#   Grade.Theory <dbl>, Grade.Ethics <dbl>, Grade.Critique <dbl>, Final_grade
<dbl>
```

5 Filtering Observations

In contrast to selection which relates to obtaining a subset of the columns of the data, filtering refers to obtaining a subset of the rows. In the `tidyverse`, data filtering is carried out with the `dplyr` package function `filter()`, which should not be confused with the base R `filter()` function in the `stats` package. As we have attached the `dplyr` package, the base R `filter()` function is masked, meaning that when we write code that uses `filter()`, the `dplyr` version of the function will automatically be called.

Filtering is often a much simpler operation than selecting variables, as the filtering conditions are based solely on the values of the data variables. Using `filter()` is analogous to the base R subset operator `[`, but the filtering condition is given as an argument to the `filter()` function instead. It is good to remind that in R a single equal sign (=) is merely for arguments of function calls, while double equal sign (==) is needed for comparison of two values. And example of filter:

```
demographics |>
  filter(Origin == "Bosnia") |>
  select(Name, Surname)
```

```
# A tibble: 2 x 2
  Name  Surname
  <chr> <chr>
1 Hamza Hodžić
2 Davud Delić
```

The code above first filters our student demographics data to only those students whose country of origin is Bosnia. Then, we select their first and last names.

Multiple filtering conditions can be refined and combined using base R logical operators, such as `&` and `|`.

```
demographics |>
  filter(Gender == "F" & Location == "Remote")
```

```
# A tibble: 10 x 8
   user       Name   Surname      Origin      Gender Birthdate  Location Employment
   <chr>      <chr>  <chr>        <chr>       <chr>  <chr>      <chr>    <chr>
 1 d93f7f0d3 Zahra  Gul          Afghanistan F      22.11.1999 Remote   None
 2 93d1f2e82 Louise Bernard      France      F      5.9.1998   Remote   Part-time
 3 417892918 Miora  Rakotomalala Madagascar  F      9.12.1995  Remote   None
 4 f98e6e2b8 Linda  Mensah       Ghana       F      7.2.1991   Remote   None
 5 590846fe3 Lucija Horvat       Croatia     F      22.7.1998  Remote   None
 # i 5 more rows
```

Here, we filtered our data to female students who are studying remotely. The same result could also be obtained by using the `filter()` function two times

```
demographics |>
  filter(Gender == "F") |>
  filter(Location == "Remote")
```

```
# A tibble: 10 x 8
   user       Name   Surname      Origin      Gender Birthdate  Location Employment
   <chr>      <chr>  <chr>        <chr>       <chr>  <chr>      <chr>    <chr>
 1 d93f7f0d3 Zahra  Gul          Afghanistan F      22.11.1999 Remote   None
 2 93d1f2e82 Louise Bernard      France      F      5.9.1998   Remote   Part-time
 3 417892918 Miora  Rakotomalala Madagascar  F      9.12.1995  Remote   None
 4 f98e6e2b8 Linda  Mensah       Ghana       F      7.2.1991   Remote   None
 5 590846fe3 Lucija Horvat       Croatia     F      22.7.1998  Remote   None
 # i 5 more rows
```

This type of approach may improve the readability of your code especially when there are several independent filtering conditions to be applied simultaneously.

Filters can naturally be based on numeric values as well. For example, we could select those students whose final grade is higher than 8.

```
results |>
  filter(Final_grade > 8)
```

```
# A tibble: 58 x 15
   user      Grade.SNA_1 Grade.SNA_2 Grade.Review Grade.Group_self Grade.Group_All
   <chr>           <dbl>       <dbl>        <dbl>            <dbl>           <dbl>
 1 111422ee7          10          10           10               10            9.11
 2 e6ec47f29          10          10           10               10            9.18
 3 4951e7386          10          10            7                9            8.56
 4 9d744e5bf          10          10           10               10            9.29
 5 0ef305578          10          10            9.33            10            8.56
 # i 53 more rows
```

```
# i 9 more variables: Grade.Excercises <dbl>, Grade.Project <dbl>,
#   Grade.Literature <dbl>, Grade.Data <dbl>, Grade.Introduction <dbl>,
#   Grade.Theory <dbl>, Grade.Ethics <dbl>, Grade.Critique <dbl>, Final_grade <dbl>
```

Similarly, we could select students based on their total number of Moodle events.

```
events_summary |>
  filter(Frequency.Total > 100 & Frequency.Total < 500)
```

```
# A tibble: 44 x 2
  user      Frequency.Total
  <chr>               <int>
1 00a05cc62             417
2 046c35846             199
3 05b604102             199
4 0604ff3d3             436
5 0af619e4b             268
# i 39 more rows
```

6 Transforming Variables

In the best-case scenario, our data is already in the desired format after it has been read into R, but this is rarely the case with real datasets. We may need to compute new variables that were not present in the original data, convert measurements to different units, or transform text data into a numeric form. In the `tidyverse`, data transformations are carried out by the `mutate()` function of the `dplyr` package. This function can be used to transform multiple variables at the same time or to construct entirely new variables. The syntax of the function is the same in both cases: first, the name of the variable should be provided followed by an R expression that defines the variable. The transformed data is not automatically assigned to any variable, enabling transformations to be used as temporary variables within a chain of piped operations.

As a simple example, we could convert the students' locations into a factor variable.

```
demographics |>
  mutate(Location = factor(Location))
```

```
# A tibble: 130 x 8
  user      Name   Surname  Origin     Gender Birthdate  Location  Employment
  <chr>     <chr>  <chr>    <chr>      <chr>  <chr>      <fct>     <chr>
1 6eba3ff82 Amanda Mora     Costa Rica F      28.2.1998  On campus None
```

```
2 05b604102 Lian   Abdullah Yemen      F      19.11.1996 On campus None
3 111422ee7 Bekim  Krasniqi Kosovo     M      30.1.1999  On campus None
4 b4658c3a9 Yusuf  Kaya     Turkey     M      16.6.1998  On campus None
5 e6ec47f29 Zoran  Babić    Serbia     M      29.10.1998 On campus Part-time
# i 125 more rows
```

As we see from the `tibble` printout, the `Location` variable is a factor in the transformed data as indicated by the `<fct>` heading under the variable name. Note that the original `demographics` data was not changed, as we did not assign the result of the computation.

The gender and employment status of the students could also be used as factors, which we could do in a single `mutate()` call

```
demographics |>
  mutate(
    Gender = factor(Gender),
    Location = factor(Location),
    Employment = factor(Employment)
  )
```

```
# A tibble: 130 x 8
  user      Name   Surname  Origin     Gender Birthdate  Location  Employment
  <chr>     <chr>  <chr>    <chr>      <fct>  <chr>      <fct>     <fct>
1 6eba3ff82 Amanda Mora     Costa Rica F      28.2.1998  On campus None
2 05b604102 Lian   Abdullah Yemen      F      19.11.1996 On campus None
3 111422ee7 Bekim  Krasniqi Kosovo     M      30.1.1999  On campus None
4 b4658c3a9 Yusuf  Kaya     Turkey     M      16.6.1998  On campus None
5 e6ec47f29 Zoran  Babić    Serbia     M      29.10.1998 On campus Part-time
# i 125 more rows
```

However, writing out individual identical transformations manually is cumbersome when the number of variables is large. For such cases, the `across()` function can be leveraged, which applies a function across multiple columns. This function uses the same selection syntax that we already familiarized ourselves with earlier to define the columns that will be transformed. To accomplish the same three transformations into a factor format, we could write

```
demographics |>
  mutate(across(c(Gender, Location, Employment), factor))
```

```
# A tibble: 130 x 8
  user      Name   Surname  Origin     Gender Birthdate  Location  Employment
  <chr>     <chr>  <chr>    <chr>      <fct>  <chr>      <fct>     <fct>
1 6eba3ff82 Amanda Mora     Costa Rica F      28.2.1998  On campus None
2 05b604102 Lian   Abdullah Yemen      F      19.11.1996 On campus None
3 111422ee7 Bekim  Krasniqi Kosovo     M      30.1.1999  On campus None
4 b4658c3a9 Yusuf  Kaya     Turkey     M      16.6.1998  On campus None
5 e6ec47f29 Zoran  Babić    Serbia     M      29.10.1998 On campus Part-time
# i 125 more rows
```

The first argument to the `across()` function is the selection that defines the variables to be transformed. The second argument defines the transformation, in this case, a function, to be used.

Working with dates can often be challenging. When we read the student demographic data into R, the variable `Birthdate` was assumed to be a `character` type variable. If we would like to use this variable to e.g., compute the ages of the students, we need to first convert it into a proper format using the `as.Date` function. Since the dates in the data are not in any standard format, we must provide the format manually. Afterwards, we can use the `lubridate` [7] package to easily compute the ages of the students, which we will save into a new variable called `Age`. We will also construct another variable called `FullName` which formats the first and last names of the students as `"Last, First"`.

```
library("lubridate")
demographics |>
  mutate(
    Birthdate = as.Date(Birthdate, format = "%d.%m.%Y"),
    Age = year(as.period(interval(start = Birthdate, end = date("2023-03-12")))),
    FullName = paste0(Surname, ", ", Name)
  ) |>
  select(Age, FullName)
```

```
# A tibble: 130 x 2
    Age FullName
  <dbl> <chr>
1    25 Mora, Amanda
2    26 Abdullah, Lian
3    24 Krasniqi, Bekim
4    24 Kaya, Yusuf
5    24 Babić, Zoran
# i 125 more rows
```

The computation of the ages involves several steps. First, we construct a time interval object with the `interval()` function from the birthdate to the date for which we wish to compute the ages. Next, the `as.period()` function converts this interval into a time duration, from which we lastly get the number of years with the `year()` function.

Suppose that we would like to construct a new variable `AchievingGroup` that categorizes the students into top 50% achievers and bottom 50% achievers based on their final grade on the course. We leverage two functions from the `dplyr` package to construct this new variable: `case_when()` and `ntile()`. The function `case_when()` is used to transform variables based on multiple sequential conditions. The function `ntile()` has two arguments, a vector `x` and an integer `n`, and it splits `x` into `n` equal-sized groups based on the ranks of the values in `x`.

```
results <- results |>
  mutate(
    AchievingGroup = factor(
      case_when(
        ntile(Final_grade, 2) == 1 ~ "Low achiever",
        ntile(Final_grade, 2) == 2 ~ "High achiever"
      )
    )
  )
```

The syntax of `case_when()` is very simple: we describe the condition for each case followed by `~` after which we define the value that the case should correspond to. We assign the result of the computation to the `results` data, as we will be using the `AchievingGroup` variable in later chapters.

We would also like to categorize the students based on their activity level, i.e., the number of total Moodle events. Our goal is to create three groups of equal size consisting of low activity, moderate activity and high activity students. The approach we applied to categorizing the achievement level of the students is also applicable for this purpose. We name our new variable as `ActivityGroup`, and we assign the result of the computation, as we will also be using this variable in later chapters.

```
events_summary <- events_summary |>
  mutate(
    ActivityGroup = factor(
      case_when(
        ntile(Frequency.Total, 3) == 1 ~ "Low activity",
        ntile(Frequency.Total, 3) == 2 ~ "Moderate activity",
        ntile(Frequency.Total, 3) == 3 ~ "High activity"
      )
    )
  )
```

7 Rearranging Data

Sometimes we may want to reorder the rows or columns of our data, for example in alphabetical order based on the names of students on a course. The `arrange()` function from the `dplyr` package orders the rows of the by the values of columns selected by the user. The values are sorted in ascending order by default, but the order can be inverted by using the `desc()` function if desired. The variable order in the selection defines how ties should be broken when duplicate values are encountered in the previous variables of the selection. For instance, the following code would arrange the rows of our `demographics` data by first comparing the surnames of the students, and then the given names for those students with the same surname. Missing values are placed last in the reordered data.

```
demographics |>
  arrange(Surname, Name)
```

```
# A tibble: 130 x 8
  user       Name   Surname  Origin      Gender Birthdate  Location  Employment
  <chr>      <chr>  <chr>    <chr>       <chr>  <chr>      <chr>     <chr>
1 ba76ebfab Bismah Abbasi   Pakistan    F      2.4.1996   Remote    Full-time
2 05b604102 Lian   Abdullah Yemen       F      19.11.1996 On campus None
3 d2c3ce9a4 Amir   Ait      Morocco     M      19.6.1997  On campus None
4 68a377c82 Saliha Akmatova Kyrgyzstan  F      19.5.1999  Remote    None
5 7e2726f3c Kazi   Akter    Bangladesh  M      22.12.1992 On campus None
# i 125 more rows
```

A descending order based on both names can be obtained by applying the `desc()` function.

```
demographics |>
  arrange(desc(Surname), desc(Name))
```

```
# A tibble: 130 x 8
  user       Name    Surname  Origin    Gender Birthdate Location  Employment
  <chr>      <chr>   <chr>    <chr>     <chr>  <chr>     <chr>     <chr>
1 a48165ad5 Liam    Zambrano Ecuador   M      4.4.1998  On campus None
2 1115dae61 Poema   Wong     Tahiti    F      22.1.1999 Remote    Part-time
3 0ef305578 Jack    White    Australia M      22.4.1995 Remote    None
4 f753ce9bf Dechen  Wangmo   Bhutan    F      29.4.1999 On campus None
5 f87eaa00c Prasert Wang     Thailand   M      9.4.1997  On campus None
# i 125 more rows
```

Column positions can be changed with the `relocate()` function of the `dplyr` package. Like `arrange()`, we first select the column or columns we wish to move

into a different position in the data. Afterwards, we specify the position where the columns should be moved to in relation to positions of the other columns. In our `demographics` data, the user ID column `user` is the first column. The following code moves this column after the `Employment` column so that the `user` column becomes the last column in the data.

```
demographics |>
  relocate(user, .after = Employment)
```

```
# A tibble: 130 x 8
  Name    Surname  Origin     Gender Birthdate  Location  Employment user
  <chr>   <chr>    <chr>      <chr>  <chr>      <chr>     <chr>      <chr>
1 Amanda  Mora     Costa Rica F      28.2.1998  On campus None       6eba3ff82
2 Lian    Abdullah Yemen      F      19.11.1996 On campus None       05b604102
3 Bekim   Krasniqi Kosovo     M      30.1.1999  On campus None       111422ee7
4 Yusuf   Kaya     Turkey     M      16.6.1998  On campus None       b4658c3a9
5 Zoran   Babić    Serbia     M      29.10.1998 On campus Part-time  e6ec47f29
# i 125 more rows
```

The mutually exclusive arguments `.before` and `.after` of `relocate()` specify the new column position in relation to columns that were not selected. These arguments also support the `select()` function syntax for more general selections.

8 Reshaping Data

Broadly speaking, tabular data typically take one of two formats: wide or long. In the wide format, there is one row per subject, where the subjects are identified by an identifier variable, such as the `user` variable in our Moodle data, and multiple columns for each measurement. In the long format, there are multiple rows per subject, and the columns describe the type of measurement and its value. For example, the `events` data is in a long format containing multiple Moodle events per student, but the `results` and `demographics` data are in a wide format with one row per student.

In the previous section, we constructed a summary of the users' Moodle events in total and of different types. The latter data is also in a long format with multiple rows per subject, but we would instead like to have a column for each event type with one row per user, which means that we need to convert this data into a wide format. Conversion between the two tabular formats is often called *pivoting*, and the corresponding functions `pivot_wider()` and `pivot_longer()` from the `tidyr` [8] package are also named according to this convention. We will create a wide format data of the counts of different event types using the `pivot_wider()` function as follows

```
library("tidyr")
events_types <- events |>
  group_by(user, Action) |>
  count(Action) |>
  pivot_wider(
    names_from = "Action",
    names_prefix = "Frequency.",
    values_from = "n",
    values_fill = 0
  )
events_types
```

```
# A tibble: 130 x 13
# Groups:   user [130]
  user      Frequency.Applications Frequency.Assignment Frequency.Course_view
  <chr>                      <int>                <int>                 <int>
1 00a05cc62                      2                  121                   103
2 042b07ba1                      0                   62                   294
3 046c35846                      0                   41                    53
4 05b604102                      0                   44                    49
5 0604ff3d3                      0                    9                   119
# i 125 more rows
# i 9 more variables: Frequency.Feedback <int>, Frequency.General <int>,
#   Frequency.Group_work <int>, Frequency.Instructions <int>,
#   Frequency.La_types <int>, Frequency.Practicals <int>, Frequency.Social <int>,
#   Frequency.Ethics <int>, Frequency.Theory <int>
```

Here, we first specify the column name that the names of the wide format data should be taken from in the long format data with `names_from`. In addition, we specify a prefix for the new column names using `names_prefix` that helps to distinguish what these new columns will contain, but in general, the prefix is optional. Next, we specify the column that contains the values for the new columns with `values_from`. Because not every student necessarily has events of every type, we also need to specify what the value should be in cases where there are no events of a particular type by using `values_fill`. As we are considering the frequencies of the events, it is sensible to select 0 to be this value. We save the results to `events_types` as we will use the event type data in later sections and chapters.

9 Joining Data

Now that we have computed the total number of events for each student and converted the event type data into a wide format, we still need to merge these new data with the demographics and results data. Data merges are also called *joins*, and

the dplyr package provides several functions for different kinds of joins. Here, we will use the left_join() function that will preserve all observations of the first argument.

```
left_join(demographics, events_summary, by = "user")
```

```
# A tibble: 130 x 10
  user     Name  Surname Origin Gender Birthdate Location Employment Frequency.Total
  <chr>    <chr> <chr>   <chr>  <chr>  <chr>     <chr>    <chr>                <int>
1 6eba3ff~ Aman~ Mora    Costa~ F      28.2.1998 On camp~ None                   312
2 05b6041~ Lian  Abdull~ Yemen  F      19.11.19~ On camp~ None                   199
3 111422e~ Bekim Krasni~ Kosovo M      30.1.1999 On camp~ None                   532
4 b4658c3~ Yusuf Kaya    Turkey M      16.6.1998 On camp~ None                   246
5 e6ec47f~ Zoran Babić   Serbia M      29.10.19~ On camp~ Part-time              356
# i 125 more rows
# i 1 more variable: ActivityGroup <fct>
```

In essence, the above left join adds all columns from events_summary to demographics whenever there is a matching value in the by column. To continue, we can use additional left joins to add the remaining variables from the results data, and the Moodle event counts of different types from events_types to have all the student data in a single object.

```
all_combined <- demographics |>
  left_join(events_types, by = "user") |>
  left_join(events_summary, by = "user") |>
  left_join(results, by = "user")
all_combined
```

```
# A tibble: 130 x 37
  user      Name   Surname  Origin     Gender Birthdate  Location  Employment
  <chr>     <chr>  <chr>    <chr>      <chr>  <chr>      <chr>     <chr>
1 6eba3ff82 Amanda Mora     Costa Rica F      28.2.1998  On campus None
2 05b604102 Lian   Abdullah Yemen      F      19.11.1996 On campus None
3 111422ee7 Bekim  Krasniqi Kosovo     M      30.1.1999  On campus None
4 b4658c3a9 Yusuf  Kaya     Turkey     M      16.6.1998  On campus None
5 e6ec47f29 Zoran  Babić    Serbia     M      29.10.1998 On campus Part-time
# i 125 more rows
# i 29 more variables: Frequency.Applications <int>, Frequency.Assignment <int>,
#   Frequency.Course_view <int>, Frequency.Feedback <int>, Frequency.General <int>,
#   Frequency.Group_work <int>, Frequency.Instructions <int>,
#   Frequency.La_types <int>, Frequency.Practicals <int>, Frequency.Social <int>,
#   Frequency.Ethics <int>, Frequency.Theory <int>, Frequency.Total <int>,
#   ActivityGroup <fct>, Grade.SNA_1 <dbl>, Grade.SNA_2 <dbl>, ...
```

We will use this combined dataset in the following chapters as well.

10 Missing Data

Sometimes it occurs that learning analytics data has cells for which the values are missing for some reason. The Moodle event data which we have utilized in this chapter does not naturally contain missing data. Thus, to have an example, we need to create a data which does. Second, handling of missing data is a vast topic of which we can only discuss some of the key points very briefly from a practical perspective. For a more comprehensive overview, we recommend reading [9] and [10] for a hands on approach. A short overview of missingness can be found in [11].

The code below will create missing values randomly to each column of `events_types` data (`user` column is an exception). To do that, we use the `mice` [12] package which also has methods for the handling of missing data. Unfortunately, `mice` is not part of the `tidyverse`. For more information about `mice`, a good source is `miceVignettes` at https://www.gerkovink.com/miceVignettes/. Now, let's create some missing data.

```
library("mice")
set.seed(44)
events_types <- events_types |>
  rename(
    "Ethics" = "Frequency.Ethics",
    "Social" = "Frequency.Social",
    "Practicals" = "Frequency.Practicals"
  )
ampute_list <- events_types |>
  ungroup(user) |>
  select(Ethics:Practicals)|>
  as.data.frame() |>
  ampute(prop = 0.3)
events_types_mis <- ampute_list$amp |>
  as_tibble()
events_types_mis[2, "Practicals"] <- NA
```

Above, we also rename the variables that contain the frequencies of Moodle events related to ethics, social and practicals into `Ethics`, `Social` and `Practicals`, respectively. Let's now see some of the values of `events_types_mis`

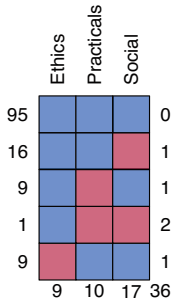
```
events_types_mis
```

```
# A tibble: 130 x 3
  Ethics Social Practicals
   <int>  <int>      <int>
1     NA     12         89
```

```
2     14     NA          NA
3      0      0          47
4      0      0          48
5      0      0          61
# i 125 more rows
```

We can see that now the data contains `NA` values in some of the cells. These are the cells in which a missing value occurs, meaning that a value for those measurements has not been recorded. A missing data pattern, that is how missing of one variable affects missingness of other variables, can be show as:

```
md.pattern(events_types_mis, rotate.names = TRUE)
```



Above, each red square indicates a missing value while blue squares stand for observed ones. We can see that there are 95 complete rows, 10 for which `Practicals` are missing, 17 have missingness on `Social` and 9 are missing on `Ethics`. Also, one row has two red squares indicating a missing value on both `Social` and `Practicals`.

Let's now discuss options of handling missing data briefly. There are four classes of statistical methods for analyzing data with missing values: complete case (CC) methods, weighting methods, imputation methods, and model-based methods. The simplest of these is complete case analysis, which leaves missing values out of the analysis and only uses observations with all variables recorded. This can be done with the `tidyr` [8] package function `drop_na()`:

```
events_types_mis |>
  drop_na()
```

```
# A tibble: 95 x 3
  Ethics Social Practicals
   <int>  <int>      <int>
1      0      0         47
```

```
2      0      0         48
3      0      0         61
4      0     24        102
5      4     18         71
# i 90 more rows
```

We can see that after using this method, our data has only 95 rows as those were the rows without any columns having missing values. This made our data much smaller! If there are a lot of missing values, the data may become too small to use for practical purposes.

A more novel group of methods are imputation methods. One of the options is using single imputation (SI) where the mean of each variable will determine the imputed value. The single mean imputation can be done as follows:

```
imp <- mice(events_types_mis, method = "mean",
m = 1, maxit = 1 , print = FALSE)
complete(imp) |>
  head()
```

```
     Ethics   Social Practicals
1  7.553719 12.00000   89.00000
2 14.000000 15.64602   74.80833
3  0.000000  0.00000   47.00000
4  0.000000  0.00000   48.00000
5  0.000000  0.00000   61.00000
6  0.000000 24.00000  102.00000
```

We can see from above that the imputed values are not integers anymore. However, if we aim to estimate means or regression coefficients (see Chapter 5 [13] for details) that is not a problem. One of the problems with mean imputation is that the variance and standard error estimates will become downward biased. A mean of `Ethics` for mean imputation is:

```
fit <- with(imp, lm(Ethics ~ 1))
summary(fit)
```

```
# A tibble: 1 x 6
  term        estimate std.error statistic  p.value  nobs
  <chr>          <dbl>     <dbl>     <dbl>    <dbl> <int>
1 (Intercept)     7.55     0.811      9.32 4.20e-16   130
```

Next, let's briefly have a look at how we can utilize multiple imputation (MI) which is an improvement over single imputation. The multiple imputation approach generates more than one imputation thus creating many complete data sets for us.

For each of these datasets, we can perform any analysis that we are interested in. After the analysis, one must pool the results from the impured datasets to get the final result. Here, we utilize a method called predictive mean matching (`method = "pmm"` in the code below), which uses the neighbour values of data as imputations.

```
imp2 <- mice(events_types_mis, method = "pmm",
m= 10, maxit = 100, print = FALSE)
fit2 <- with(imp2, lm(Ethics ~ Practicals))
pool_fit <- pool(fit2)
# Multiple imputation
summary(pool_fit)
```

```
         term   estimate  std.error statistic       df     p.value
1 (Intercept) 1.62080372 2.18285149 0.7425167 101.7304 0.459485402
2  Practicals 0.08049328 0.02616765 3.0760606 106.0802 0.002668431
```

```
# Complete cases
summary(lm(Ethics ~ Practicals, events_types_mis))["coefficients"]
```

```
$coefficients
             Estimate Std. Error  t value   Pr(>|t|)
(Intercept) 2.15459162 2.12235051 1.015191 0.31226283
Practicals  0.06220884 0.02605001 2.388054 0.01865793
```

```
# Without missingness
summary(lm(Ethics ~ Practicals, events_types))["coefficients"]
```

```
$coefficients
             Estimate Std. Error   t value     Pr(>|t|)
(Intercept) 1.05409529 2.17821479 0.4839262 0.6292651258
Practicals  0.08891892 0.02590313 3.4327482 0.0008053447
```

From the results above, we can see that in this particular case the multiple imputation performs well in comparison to CC approach. The regression coefficient for full data without any missing values is 0.089, and it is 0.080 for multiple imputation, while complete case analysis gives 0.062. As all of them have very similar standard errors, this yields that MI and full data give statistically significant p-values for significance level 0.01, while CC does not.

11 Correcting Erroneous Data

Let's imagine that our data has an error on the surname variable `Surname` and that all the names ending with "sen" should end with "ssen". What we can do is that we

can use regular expressions to detect the erroneous rows and we can also use them to replace the values. Let's first figure out which last names contain a name ending with "sen". We can use a function `str_detect()` to return TRUE/FALSE for each row from `stringr` [14] package within a `filter()` function call. We define `pattern = "sen$"` where `$` indicates the end of the string.

```
library("stringr")
demographics |>
  filter(str_detect(string = Surname, pattern = "sen$")) |>
  pull(Surname)
```

```
[1] "Nielsen"  "Johansen" "Joensen"  "Jansen"   "Olsen"
```

After pulling the filtered surnames, there seems to be five surnames ending with "sen". Next, let's try to replace "sen" with "ssen". On the next row we filter just as previously to limit output.

```
demographics |>
  mutate(Surname = str_replace(
    string = Surname, pattern = "sen$", replacement = "ssen")
  ) |>
  filter(str_detect(string = Surname, pattern = "sen$")) |>
  pull(Surname)
```

```
[1] "Nielssen"  "Johanssen" "Joenssen"  "Janssen"   "Olssen"
```

Thus, the following code updates the data so that all the surnames ending with "sen" now end with "ssen" instead.

```
demographics <- demographics |>
  mutate(Surname = str_replace(
    string = Surname, pattern = "sen$", replacement = "ssen")
  )
```

12 Conclusion and Further Reading

Data wrangling is one of the most important steps in any data analysis pipeline. This chapter introduced the `tidyverse`, tidy data, and several commonly used R packages for data manipulation and their use in basic scenarios in the context of learning analytics. However, the `tidyverse` is vast and can hardly be fully covered in a single chapter. We refer the reader to additional resources such as those found on the tidyverse website at https://www.tidyverse.org/learn/ and the book "R for Data Science" by Hadley Wicham and Garret Grolemund. The book is free to use and readily available online at https://r4ds.had.co.nz/.

References

1. Wickham H, Averick M, Bryan J, Chang W, McGowan LD, François R, Grolemund G, Hayes A, Henry L, Hester J, Kuhn M, Pedersen TL, Miller E, Bache SM, Müller K, Ooms J, Robinson D, Seidel DP, Spinu V, Takahashi K, Vaughan D, Wilke C, Woo K, Yutani H (2019) Welcome to the tidyverse. J Open Source Softw 4:1686. https://doi.org/10.21105/joss.01686
2. Wickham H, Hester J, Bryan J (2024) readr: read rectangular text data. R package version 2.1.5. https://CRAN.R-project.org/package=readr
3. López-Pernas S, Saqr M, Conde J, Del-Río-Carazo L (2024) A broad collection of datasets for educational research training and application. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, Berlin
4. Chan C, Leeper T, Becker J, Schoch D (2023) rio: a swiss-army knife for data file I/O. https://cran.r-project.org/package=rio
5. Müller K, Wickham H (2023) tibble: simple data frames. R package version 3.2.1. https://CRAN.R-project.org/package=tibble
6. Wickham H, François R, Henry L, Müller K, Vaughan D (2023) dplyr: a grammar of data manipulation. R package version 1.1.4. https://CRAN.R-project.org/package=dplyr
7. Grolemund G, Wickham H (2011) Dates and times made easy with lubridate. J Stat Softw 40(3):1–25. https://www.jstatsoft.org/v40/i03/
8. Wickham H, Vaughan D, Girlich M (2023) tidyr: tidy messy data. R package version 1.3.0. https://CRAN.R-project.org/package=tidyr
9. Little RJA, Rubin DB (2020) Statistical analysis with missing data, 3rd edn. Wiley, Hoboken
10. van Buuren S (2018) Flexible imputation of missing data. Chapman et Hall/CRC Press, Boca Raton. https://doi.org/10.1201/9780429492259
11. Kopra J (2018) Statistical modelling of selective non-participation in health examination surveys. Report/University of Jyväskylä, Department of Mathematics and Statistics
12. van Buuren S, Groothuis-Oudshoorn K (2011) mice: multivariate imputation by chained equations in R. J Stat Softw 45:1–67. https://doi.org/10.18637/jss.v045.i03
13. Tikka S, Kopra J, Heinäniemi M, López-Pernas S, Saqr M (2024) Introductory statistics with R for educational researchers. In: Saqr M, López-Pernas S (eds). Springer, Berlin
14. Wickham H (2023) stringr: simple, consistent wrappers for common string operations. R package version 1.5.1. https://CRAN.R-project.org/package=stringr

Introductory Statistics with R for Educational Researchers

Santtu Tikka, Juho Kopra, Merja Heinäniemi, Sonsoles López-Pernas, and Mohammed Saqr

1 Introduction

Learning analytics involves the practical application of statistical methods to quantitative data, which can represent various aspects of the learning process such as student engagement, progress, and outcomes. Thus, knowledge about basic statistical methods is essential. Let's start with how statistics connects with the research process and how it is also linked with philosophy of science. According to Niiniluoto [1], the research process can be described with the following eight steps:

1. Setting up a problem.
2. Disambiguation of the problem. Building a research strategy.
3. Collecting data.
4. Describing the data.
5. Analysis of data.
6. Interpreting the analyses.
7. Writing the report.
8. Publishing the results.

Steps 1 and 2 require knowledge and skills related to the applied field, but also general scientific aptitude. Knowledge about statistics is central in steps 3, 4, 5, and

S. Tikka (✉)
Department of Mathematics and Statistics, University of Jyväskylä, Jyväskylä, Finland
e-mail: santtu.tikka@jyu.fi

J. Kopra · S. López-Pernas · M. Saqr
School of Computing, University of Eastern Finland, Joensuu, Finland

M. Heinäniemi
Institute of Biomedicine, University of Eastern Finland, Kuopio, Finland

M. Saqr, S. López-Pernas (eds.), *Learning Analytics Methods and Tutorials*,
https://doi.org/10.1007/978-3-031-54464-4_5

6. Finally, steps 7 and 8 mostly require skills in writing and communication. Overall, it can be argued that a solid understanding of statistics and statistical methods is crucial for anyone conducting research with quantitative data.

This chapter of the book concentrates on steps 4, 5, and 6 of the research process. We start with descriptive statistics, which are statistics that describe the overall features of the data. In contrast, inferential statistics are used to draw conclusions and make inferences about the population under study. Afterwards, we explain the basics of statistical hypothesis testing, which is the most common—although not the only—way to analyze data. The most common statistical tests, such as Student's t-test, Chi-squared test, Analysis of variance, Levene's test, and Shapiro-Wilk test are covered in this chapter. We also explain how to interpret the results of each test. We also present the linear regression model, which is not a statistical test but one of the most powerful statistical tools. Basic understanding of linear regression is essential for anyone interested in more advanced regression techniques, including logistic regression which is covered in the final section of this chapter. For a more in-depth view on the statistical tests covered in this chapter, we refer the reader to works such as [2, 3].

2 Descriptive Statistics

Descriptive statistics are used to provide a meaningful quantitative overview of data, and to summarize potentially vast amounts of information into more easily comprehensible and manageable quantities. In general, descriptive statistics are used as a first step in a data analysis workflow. In this section we will focus on numeric descriptors while visualizations are the topic of Chapter 6 [4]. For this chapter, we will use the combined Moodle data with students' demographics, results, and summarized Moodle event activity. For more information about the dataset, please refer to Chapter 2 in this book [5]. We begin by installing all R packages that we will use in this chapter.

```
install.packages(
  c("car", "rio", "see", "dplyr", "tidyr",
    "broom", "report", "correlation", "performance")
)
```

We use the `rio` [6] package to read the data files into R via the `import()` function. The `Events` dataset contains log data on the student's Moodle activity such as Moodle event types and names. The `Demographics` dataset contains background information on the students such as their gender and location of study (categorical variables). Finally, the `Results` data contains grades on various aspects of the Moodle course including the final course grade (numeric variables). We also create a new variable called `AchievingGroup` that categorizes the students

into bottom and top 50% of achievers in terms of the final grade. We will leverage the `dplyr` [7] and `tidyr` [8] packages to wrangle the data into a single combined dataset. We begin by reading the data files and by constructing the `AchievingGroup` variable.

```
library("rio")
library("dplyr")
library("tidyr")
url <- "https://github.com/lamethods/data/raw/main/1_moodleLAcourse/"
events <- import(paste0(url, "Events.xlsx"), setclass = "tibble")
demographics <- import(paste0(url, "Demographics.xlsx"), setclass = "tibble")
results <- import(paste0(url, "Results.xlsx"), setclass = "tibble") |>
  mutate(
    AchievingGroup = factor(
      case_when(
        ntile(Final_grade, 2) == 1 ~ "Low achiever",
        ntile(Final_grade, 2) == 2 ~ "High achiever"
      )
    )
  )
```

Next, we summarize the student's engagement based on their Moodle activity into three groups: Low activity, Moderate activity and High Activity.

```
events_summary <- events |>
  group_by(user) |>
  tally() |>
  rename(Frequency.Total = n) |>
  mutate(
    ActivityGroup = factor(
      case_when(
        ntile(Frequency.Total, 3) == 1 ~ "Low activity",
        ntile(Frequency.Total, 3) == 2 ~ "Moderate activity",
        ntile(Frequency.Total, 3) == 3 ~ "High activity"
      )
    )
  )
```

We also count the different types of Moodle events.

```
events_types <- events |>
  group_by(user, Action) |>
  count(Action) |>
  pivot_wider(
    names_from = "Action",
```

```
    names_prefix = "Frequency.",
    values_from = "n",
    values_fill = 0
  )
```

Finally, we combine the data.

```
all_combined <- demographics |>
  left_join(events_types, by = "user") |>
  left_join(events_summary, by = "user") |>
  left_join(results, by = "user")
```

The various steps of the combined data construction are discussed in greater detail in Chapter 4 [9].

2.1 Measures of Central Tendency

A typical way to summarize a univariate data sample is to describe its "middle point" using an appropriate statistic depending on the measurement scale of the data. The most common statistics to describe such a value are the *mean*, the *median*, and the *mode*.

For data on the interval or ratio scales (and sometimes also on the ordinal scale), the most common option is to use the arithmetic mean, which is available via the base R function `mean()`. This function takes a vector of values as its input.

```
all_combined |>
  summarise(
    mean_grade = mean(Final_grade),
    mean_total = mean(Frequency.Total)
  )
```

```
# A tibble: 1 x 2
  mean_grade mean_total
       <dbl>      <dbl>
1       7.25       736.
```

The means are reported as a `tibble` with a single column for both variables.

For data on the ordinal scale (or interval or ratio scales), the median can be used which is defined as the value that separates the lower half from the bottom half of the data sample, i.e., the 50% quantile. The median can be computed in R using the built-in `median()` function. Similarly to `mean()`, this function also takes a vector of values as its input.

```
all_combined |>
  summarise(
    median_grade = median(Final_grade),
    median_total = median(Frequency.Total)
  )
```

```
# A tibble: 1 x 2
  median_grade median_total
         <dbl>        <dbl>
1         7.95          606
```

Just like before, the medians are reported as a `tibble` with each value in its own column.

For data on the nominal or ordinal scale, the mode is a suitable choice as it describes the category with the highest number of observations. Unfortunately, there is no readily available function in R to compute the mode, and the reader should take care not to mistakenly use the `mode()` function, which is used to determine the internal storage mode of a variable (similar to the `typeof()` function). However, we can easily write our own function to compute the statistical mode as follows:

```
stat_mode <- function(x) {
  u <- unique(x)
  u[which.max(tabulate(match(x, u)))]
}
```

Functions in R are written using the following syntax. First, we define the name of the function, just like we would define the name of a variable when assigning data into it, in this case the name is `stat_mode`. Next, we assign the function definition, which starts with the keyword `function`. Next, we describe the function arguments within the parentheses, in this case we call our argument `x`, which we assume contains the data vector we wish to compute the mode for. Next, we define the body of the function within the braces. The body determines what the function does and what its output should be. Within the body, we first determine the unique values in the data vector `x`, and assign the result to a variable `u`. Next, we need to count the number of occurrences of each unique value. To start, we first match each observed value in `x` to the unique values in `u` to get the corresponding indices, which we will then count using tabulate. We obtain the index of the value with the highest number of occurrences with the function `which.max()`, and finally the corresponding unique value by selecting it from `u` using the subset operator, i.e., the brackets. Our function will now work on all types of data.

```
all_combined |>
  summarise(
    mode_gender = stat_mode(Gender),
```

```
    mode_location = stat_mode(Location)
  )
```

```
# A tibble: 1 x 2
  mode_gender mode_location
  <chr>       <chr>
1 F           On campus
```

The output is now similar to the `mean` and `median` functions that we used earlier, showing the modes of `Gender` and `Location` as a two-column `tibble`. For nominal variables, it is common to also compute the frequencies of each category. This can easily be done with the base R function `table()`

```
table(all_combined$Gender)
```

```
 F  M
65 65
```

```
table(all_combined$Location)
```

```
On campus    Remote
      106        24
```

The function outputs the names of the categories and the frequency of each category as an integer vector.

2.2 *Measures of Dispersion*

For data on the interval and ratio scales (and sometimes also on the ordinal scale), it is also meaningful to describe how clustered or scattered the values in the sample are, i.e., how far apart the values are from one another. Commonly used measures of statistical dispersion include the *variance*, *standard deviation*, and the *interquartile range*. Typically, such measures have the value 0 when all values in the sample are identical, and the value increases as the dispersion in the data grows.

All three measures can be readily computed with built-in R functions `var()`, `sd()`, and `IQR()` respectively. Like `mean()` and `median()`, these functions accept a vector or numeric values as input.

```
all_combined |>
  summarise(
    var_grade = var(Final_grade),
```

```
    sd_grade = sd(Final_grade),
    iqr_grade = IQR(Final_grade)
  )
```

```
# A tibble: 1 x 3
  var_grade sd_grade iqr_grade
      <dbl>    <dbl>     <dbl>
1      4.81     2.19      3.34
```

The variance, standard deviation and interquartile range of the final grade are returned as a `tibble` with three columns.

2.3 Covariance and Correlation

Covariance and correlation measure the linear dependence between two variables. Correlation is a unitless measure between -1 and 1, whereas covariance is not, and its scale depends on the scale of the variables. The sign of both measures indicates the tendency of the relationship. Positive sign means that as the value of one variable increases, the value of the other variable tends to increase as well. Conversely, a negative sign indicates that the value of the second variable tends to decrease as the value of the first variable increases.

Both covariance and correlation and be computed directly in R using the functions `cov()` and `cor()`, respectively.

```
all_combined |>
  summarise(
    cov_grade_total = cov(Final_grade, Frequency.Total),
    cor_grade_total = cor(Final_grade, Frequency.Total),
  )
```

```
# A tibble: 1 x 2
  cov_grade_total cor_grade_total
            <dbl>           <dbl>
1            504.           0.513
```

We obtain the covariance and correlation between the final grade and total number of Moodle events. We will familiarize ourselves with correlations in greater depth in Sect. 4.

2.4 Other Common Statistics

The extreme values of a data sample can be found using the function `range()`, which computes the minimum and maximum values of the sample as a vector. These

values can also be computed individually with the corresponding functions `min()` and `max()`. Because `summarise()` only allows a single value as output per row, we use the `reframe()` function instead when computing the range.

```
all_combined |>
  reframe(
    range_grade = range(Final_grade)
  )
```

```
# A tibble: 2 x 1
  range_grade
        <dbl>
1           0
2          10
```

```
all_combined |>
  summarise(
    min = min(Final_grade),
    max = max(Final_grade)
  )
```

```
# A tibble: 1 x 2
    min   max
  <dbl> <dbl>
1     0    10
```

With `reframe()`, we obtain a `tibble` with two rows, the first containing the minimum value and the second the maximum value of the final grade. If we instead use `summarise()` like before, we can only obtain one value per computed variable. The `summary()` function can also be used to quickly compute several of the most common descriptive statistics for all variables of a dataset.

```
results |>
  select(Grade.SNA_1:Grade.Group_self) |>
  summary()
```

```
  Grade.SNA_1      Grade.SNA_2      Grade.Review    Grade.Group_self
 Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
 1st Qu.: 8.000   1st Qu.: 9.000   1st Qu.: 6.670   1st Qu.: 9.000
 Median : 9.000   Median :10.000   Median : 8.000   Median :10.000
 Mean   : 8.346   Mean   : 9.262   Mean   : 7.724   Mean   : 8.085
 3rd Qu.:10.000   3rd Qu.:10.000   3rd Qu.: 9.670   3rd Qu.:10.000
 Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
```

The output shows the minimum and maximum values, the quartiles, and the mean of each variable that we selected.

3 Statistical Hypothesis Testing

Statistical hypothesis testing aims to evaluate hypotheses about a population of interest using probabilistic inference. The starting point of any statistical test is a so-called *null hypothesis* (denoted by H_0), which typically corresponds to a scenario, where evidence supporting a specific hypothesis is a result of pure chance. For example, when evaluating whether a new drug is an efficient form of treatment via a randomized controlled trial, the null hypothesis could be that the drug has no effect on the response. A null hypothesis is always associated with an *alternative hypothesis* (denoted by H_1), which is typically the inverse of the null hypothesis and corresponds to the hypothesis of interest, e.g., the drug has an effect on the response.

Statistical tests operate by assuming that the null hypothesis is true, and highly unlikely events under this assumption are typically regarded as giving cause for rejecting the null hypothesis. A statistical test is associated with a *test statistic*, which is a measure of how much the observations deviate from the null hypothesis scenario. The distribution of the test statistic under the null hypothesis and the sample test statistic can be used to compute the probability of obtaining a test statistic as extreme or more extreme than the one observed, assuming that the null hypothesis is true. This probability is known as the *p-value*, which is often mischaracterized even in scientific literature. For instance, the p-value is not the probability that the null hypothesis is true or that the alternative hypothesis is false. The p-value also does not quantify the size of the observed effect, or its real-world importance.

Typically, a *confidence level* is decided before applying a statistical test (usually denoted by α), and the null hypothesis is rejected if the observed p-value is smaller than this confidence level. If the p-value is greater than the confidence level, the null hypothesis is not rejected. Traditionally, the confidence level is 0.05, but this convention varies by field, and should be understood as being arbitrary, i.e., there is nothing special about the value 0.05. If the p-value falls below the confidence level, the result is regarded as *statistically significant*.

Hypothesis testing is a powerful tool for drawing conclusions from data, but it is important to use it appropriately and to understand its limitations. Every statistical test is associated with a set of assumptions which are often related to the distribution of the data sample. If these assumptions are violated, then the results of the test may be unreliable. In the following sections, some of the most common statistical tests are introduced. We will take advantage of the `report` [10] package and the corresponding function `report()` to showcase the results of the various statistical tests. For more information on the concepts and principles related to statistical hypothesis testing, see e.g., [2, 3].

3.1 Student's t-test

Student's t-test [11] is one of the most well-known statistical tests. It compares the mean values of variables either between two populations or between a single population and a reference level and is thus applicable to continuous variables. The test assumes homogeneity of variance and that the data originates from a normal distribution. For nonhomogeneous data, the test can still be performed by using an approximation [12]. In R, all variants of the t-test test can be applied using the function `t.test()`.

Our goal is to compare the students' Moodle activity with respect to their final grade. For this purpose, we use the binary variable called `AchievingGroup` which categorizes the students into top and bottom 50% achievers in terms of the final grade.

3.1.1 One-Sample t-test

The one-sample t-test compares the mean of a data sample against a reference value, typically defined by the null hypothesis. Let us begin by testing the hypothesis that the average number of Moodle events (`Frequency.Total`) is 600 ($H_0 : \mu = 600$). The function `t.test()` can be used in various ways, but in this example we provide the function with a `formula` object `Frequency.Total ~ 1` as the first argument. The formula syntax is a standard method for defining statistical models and other dependency structures in R. The formula defines that the left-hand side of the `~` symbol is a response variable which is explained by the terms on the right-hand side. Because we're not conducting the test with respect to any other variable, the right-hand side of the formula is simply `1`, which means that it is a constant in the R formula syntax. This does not mean for example, that our null hypothesis would be that the number of Moodle events is 1. The expected value that the test is applied against (i.e., the value we assume μ to have under the null hypothesis) is defined via the argument `mu`, which by default has the value 0 for a one-sample t-test. Argument `data` defines in which environment the formula should be evaluated. By providing the `all_combined` data, we do not have to explicitly extract the `FrequencyTotal` variable from the data in the formula by writing `all_combined$Frequency.Total` or by using `pull()`. This is especially useful when the formula contains several variables from the same data.

```
ttest_one <- t.test(Frequency.Total ~ 1, data = all_combined, mu = 600)
ttest_one
```

```
    One Sample t-test

data:  Frequency.Total
t = 3.4511, df = 129, p-value = 0.0007553
```

```
alternative hypothesis: true mean is not equal to 600
95 percent confidence interval:
 657.8530 813.3163
sample estimates:
mean of x
 735.5846
```

As a result, we obtain the value of the test statistic (`t = 3.4511`), the degrees of freedom (`df = 129`), and the p-value of the test (`p-value = 0.0007553`). Because the p-value is very small (much smaller than the standard 0.05 confidence level), we reject the null hypothesis, which means that the average number of Moodle events is significantly different from 600. The output of the test result object also describes the alternative hypothesis H_1 under `alternative hypothesis`, and the confidence interval of the test statistic.

We produce a summarized report of the test results with the `report()` function.

```
report(ttest_one)
```

```
Warning: Unable to retrieve data from htest object.
  Returning an approximate effect size using t_to_d().

Effect sizes were labelled following Cohen's (1988) recommendations.

The One Sample t-test testing the difference between Frequency.Total (mean =
735.58) and mu = 600 suggests that the effect is positive, statistically
significant, and small (difference = 135.58, 95% CI [657.85, 813.32], t(129) =
3.45, p < .001; Cohen's d = 0.30, 95% CI [0.13, 0.48])
```

This produces a description of the results of the test that is easier to read and interpret than the direct output of the test result object. We note that a warning is also produced which we can safely ignore in this case. The warning occurs because the test result object `ttest_one` does not retain the original data `all_combined` which we used to carry out the test. If non-approximate effect sizes are desired, the test should be carried out by supplying the variables being compared directly without using the formula interface of the `t.test()` function. For more information on the effect size, see e.g., [13, 14].

3.1.2 Two-Sample t-test

In contrast to the one-sample t-test, the two-sample t-test compares the means of two data samples against one another. For example, suppose that we're interested in the hypothesis that the average number of Moodle events is the same for the top and bottom 50% achievers ($H_0 : \mu_1 = \mu_2$). We can once again leverage the formula syntax, but instead of the constant `1` on the right-hand side of the formula, we will now replace it with the variable `Achievement` which defines the achievement level.

```
ttest_two <- t.test(Frequency.Total ~ AchievingGroup, data = all_combined)
ttest_two
```

```
    Welch Two Sample t-test

data:  Frequency.Total by AchievingGroup
t = 4.4749, df = 95.988, p-value = 2.102e-05
alternative hypothesis:
true difference in means between group 1 and group 2 is not equal to 0
95 percent confidence interval:
 182.6427 473.8496
sample estimates:
mean in group High achiever  mean in group Low achiever
                   899.7077                    571.4615
```

The contents of the result object are mostly the same as in the case of the one-sample t-test. The result is again statistically significant (using the 0.05 confidence level) meaning that according to the test, the average number of Moodle events is higher for the top 50% achievers. The `report()` function can be used to produce a similar summary as in the case of the one-sample t-test.

```
report(ttest_two)
```

```
Warning: Unable to retrieve data from htest object.
  Returning an approximate effect size using t_to_d().

Effect sizes were labelled following Cohen's (1988) recommendations.

The Welch Two Sample t-test testing the difference of Frequency.Total by
AchievingGroup (mean in group High achiever = 899.71, mean in group Low achiever =
571.46) suggests that the effect is positive, statistically significant, and large
(difference = 328.25, 95% CI [182.64, 473.85], t(95.99) = 4.47, p < .001; Cohen's d
= 0.91, 95% CI [0.49, 1.33])
```

This produces the same warning as before in the one-sample case, but we can safely ignore it again.

3.1.3 Paired Two-Sample t-test

Instead of directly comparing the means of two groups, it may sometimes be of interest to compare differences between pairs of measurements. Such a scenario typically arises in an experiment, where subjects are paired, or two sets of measurements are taken from the same subjects. While our Moodle event data does not contain such measurement pairs, we could still imagine that our data was organized such that each student in the bottom 50% achievers was paired with a

student in the top 50% achievers and that there is a one-to-one correspondence between the two achievement groups. This dependency between the two groups is the key difference between the paired test and the two-sample test.

A more suitable approach for paired data is to test the differences between the pairs, e.g., the differences between the number of Moodle events in our scenario. We supply the `t.test()` function with the argument `paired = TRUE` so that it will take the measurement pairs into account. In this test, the null hypothesis is that the mean difference between the student pairs is zero ($H_0 : \mu_d = 0$).

```
ttest_paired <- t.test(
  Frequency.Total ~ AchievingGroup, data = all_combined, paired = TRUE
)
ttest_paired
```

```
    Paired t-test

data:  Frequency.Total by AchievingGroup
t = 4.3733, df = 64, p-value = 4.599e-05
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 178.3014 478.1910
sample estimates:
mean difference
       328.2462
```

The result is once again statistically significant, and we reject the null hypothesis. Because the mean difference between the pairs is positive, this means average number of Moodle events is higher for the top 50% achievers of the pairs.

Paired two-sample t-test is also supported by `report()`.

```
report(ttest_paired)
```

```
Warning: Unable to retrieve data from htest object.
  Returning an approximate effect size using t_to_d().

Effect sizes were labelled following Cohen's (1988) recommendations.

The Paired t-test testing the difference of Frequency.Total by AchievingGroup (mean
difference = 328.25) suggests that the effect is positive, statistically
significant, and medium (difference = 328.25, 95% CI [178.30, 478.19], t(64) =
4.37, p < .001; Cohen's d = 0.55, 95% CI [0.28, 0.81])
```

We can once again safely ignore the produced warning message.

3.2 Chi-Squared Test

The chi-squared test is used for the analysis of contingency tables; it tests whether two categorical variables are independent or not [15]. A typical use case for this test is to investigate differences between groups such as student attendance by location or gender. The basic idea of the test is to compare the observed contingency table to a table under the null hypothesis where the variables are independent. The chi-squared test is based on the cell-specific differences between these two tables. As a general rule, the test assumes that the expected value is at least 5 in at least 80% of the cells, and that no expected values are below 1. If these assumptions are violated, the results of the test may not be reliable. In such cases, Fisher's exact test [16] can be used instead via the function `fisher.test()`, but it may be computationally slow for large contingency tables. Both the chi-squared test and Fisher's exact test assume that the data is a random sample from the population.

We will use the combined Moodle data to investigate whether the achievement level and the activity level of the students are independent. First, we must create the contingency table from the individual-level data. We use the `table()` function for this purpose.

```
tab <- table(all_combined$ActivityGroup, all_combined$AchievingGroup)
tab
```

```
                  High achiever Low achiever
  High activity                27           16
  Low activity                 14           30
  Moderate activity            24           19
```

The table shows the observed frequencies in each cell, i.e., for each combination of activity and achievement. Next, we apply the chi-squared test using the `chisq.test()` function.

```
Xsq_test <- chisq.test(tab)
Xsq_test
```

```
    Pearson's Chi-squared test

data:  tab
X-squared = 9.2135, df = 2, p-value = 0.009984
```

Printing the test result object shows the test statistic (`X-squared`), the associated degrees of freedom (`df`) and the p-value (`p-value`). The p-value is very small, meaning that we reject the null hypothesis. In other words, the achievement and activity levels of the students are not independent. This is not a surprising result, as

more active students are more likely to engage with the course content and perform better in terms of the learning outcomes. We can confirm that the assumptions of the test related to the expected values of the cells were not violated by using the test result object, which contains the expected values of the cells in the element `expected`.

```
all(Xsq_test$expected >= 1)
```

```
[1] TRUE
```

```
mean(Xsq_test$expected >= 5) >= 0.80
```

```
[1] TRUE
```

All expected values were greater than one, and over 80% of the expected values were greater than 5. This means that the assumptions are satisfied for our data and thus the results are reliable. Here, we used the function `all()` which takes a `logical` vector as input and returns `TRUE` if all elements of the vector were `TRUE`. Otherwise, the function returns `FALSE`. Unfortunately, the `report()` function does not support the chi-squared test.

3.3 Analysis of Variance

Analysis of variance (ANOVA) [17] can be viewed as the generalization of Student's t-test, where instead of one or two groups, the means of a variable are compared across multiple groups simultaneously. The name of the method comes from its test statistic, which is based on a decomposition of the total variance of the variable into variance within the groups and between the groups. ANOVA makes several assumptions: the observations are independent, the residuals of the underlying linear model follow a normal distribution, and that the variance of the variable is the same across groups (homoscedasticity). If these assumptions are violated, the results of the test may not be reliable. One alternative in such instances is to use the non-parametric Kruskal-Wallis test [18] instead, which is available in R via the function `kruskal.test()`. This test uses the ranks of the observations, and the null hypothesis is that the medians are the same for each group.

We use our combined Moodle data to demonstrate ANOVA. Instead of comparing the total number of Moodle events between top and bottom 50% of achievers, this time we will compare the final grade of the students across three activity groups: low activity, moderate activity, and high activity, described by the variable `ActivityGroup`. Thus the null and alternative hypotheses are in this case:

- H_0: The expected values of the final grade are the same across the three activity groups ($\mu_1 = \mu_2 = \mu_3$),

- H_1: At least one activity group has a different expected final grade ($\mu_i \neq \mu_j$ for at least one pair $i \neq j$).

To carry out the analysis, we apply the `aov` function, which uses the same formula syntax to define the response variable and the groups as the `t.test()` function does. Next, we apply the `summary()` function to the `aov()` function return object `fit`, as the default output of `aov()` is not very informative.

```
fit <- aov(Final_grade ~ ActivityGroup, data = all_combined)
summary(fit)
```

```
                Df Sum Sq Mean Sq F value   Pr(>F)
ActivityGroup    2  175.7   87.87   25.11 6.47e-10 ***
Residuals      127  444.4    3.50
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The summary contains the following columns: `Df` describes the degrees of freedom of the F-distribution associated with the test, `Sum Sq` reports the sum of squares related to the groups and the residuals, `Mean Sq` reports the corresponding mean sum of squares, `F value` is the value of the test statistic, and finally `Pr(>F)` is the p-value of the test. For this example, the p-value is very small, which means that the null hypothesis is rejected, and there are statistically significant differences in the final grade between the groups according to the test. In the following sections we will learn how to test for the assumptions related to normality and homoscedasticity. The `report()` function can be used for the output of `aov()` as well.

```
report(fit)
```

```
The ANOVA (formula: Final_grade ~ ActivityGroup) suggests that:

  - The main effect of ActivityGroup is statistically significant and large (F(2,
127) = 25.11, p < .001; Eta2 = 0.28, 95% CI [0.17, 1.00])

Effect sizes were labelled following Field's (2013) recommendations.
```

This output also reports the degrees of freedom, the test statistic value and the p-value but in a more easily readable format.

Note that ANOVA simply measures if there are differences between the groups but does not provide information on how these differences emerge. For example, there could be a single group that is different from all the rest, or two subgroups where the means are similar within each group, but different between the subgroups. Visualizations can be a helpful tool for gaining more insight into the differences, and post-hoc pairwise tests can be carried out to compare the pairs of groups.

3.4 Levene's Test

Levene's test is used to investigate whether the variance of a variable is the same across two or more groups [19]. Compared to alternatives such as Bartlett's test [20], Levene's test is less sensitive to non-normal observations. The test is not available in base R, but it can be found in the `car` package as the function `leveneTest()`. The function uses the same formula syntax as `t.test()` and `aov()`. We will investigate the homogeneity of the variance of the final grade between the activity groups.

```
library("car")
leveneTest(Final_grade ~ ActivityGroup, data = all_combined)
```

```
Levene's Test for Homogeneity of Variance (center = median)
       Df F value   Pr(>F)
group   2  5.7204 0.004181 **
      127
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The output of `leveneTest()` is analogous to the output of the ANOVA summary, and it contains the degrees of freedom (`Df`), the value of the test statistic (`F value`) and the p-value of the test (`Pr(>F)`). The p-value is very small, so we reject the null hypothesis meaning that the variance of the final grade is not the same across the groups according to the test. This means that the assumption of homoscedasticity is violated for the analysis of variance of the final grade, and thus the results may not be reliable. The `report()` function is not supported for `leveneTest()`.

3.5 Shapiro-Wilk Test

Shapiro-Wilk test tests the null hypothesis that a data sample originated from a normal distribution [21]. The test is available in base R as the function `shapiro.test()`. Unfortunately, this function does not support the formula syntax unlike the other test functions we have used thus far. The function only accepts a single `numeric` vector as its argument. Therefore, to test the normality of multiple groups simultaneously, the data must first be split into the groups to be tested. We apply the test to the final grade in each achievement group. With the help of the `broom` package [22], we wrap the test results into a tidy format.

```
library("broom")
all_combined |>
  # Performs the computations in each activity group
  group_by(ActivityGroup) |>
```

```
  # Apply a function in each group
  group_modify(~{
    # Apply the Shapiro test in each group and create tidy output
    shapiro.test(.$Final_grade) |>
      tidy()
  }) |>
  # Selection of variables to keep in the output
  select(ActivityGroup, statistic, p.value)
```

```
# A tibble: 3 x 3
# Groups:   ActivityGroup [3]
  ActivityGroup     statistic  p.value
  <fct>                 <dbl>    <dbl>
1 High activity         0.918 0.00448
2 Low activity          0.909 0.00215
3 Moderate activity     0.862 0.000103
```

This is also a great example of the tidyverse paradigm. First, we group the data by `ActivityGroup` using `group_by()`, and then apply a function in each group using `group_modify()`. We apply the `shapiro.test()` function to the `Final_grade` variable, and then we convert the test results into a tidy tibble using the `tidy()` function from the `broom` package. We also use the special dot notation `.` to select the final grade variable from the data in each group. Finally, we select the grouping variable (`ActivityGroup`), the test statistic (`statistic`) and the p-value (`p.value`) of each test using `select()` and print the results. The resulting object is a tibble with three columns: `ActivityGroup`, `statistic` and `p.value`, the last two of which give the test statistic and p-value of the test for the activity group of the first column.

We can see that according to the test, the `Final_grade` variable is not normally distributed in any of the activity groups, as the p-values are very small. As a consequence, the results of the analysis of variance carried out earlier may not be reliable.

4 Correlation

In Sect. 2.3, we briefly described covariance and correlation, and showcased the base R functions to compute them. However, there are several powerful and user-friendly packages for the analysis and reporting of correlations, such as the `correlation` [23] package which we will demonstrate in this section.

```
library("correlation")
```

For example, the package can easily compute all pairwise correlations between the numeric variables of the data with the function `correlation()`. The argument `select` can be used to compute the correlations only for a subset of the variables.

```
corrs <- correlation(
  all_combined,
  select = c("Frequency.Total", "Grade.Theory", "Final_grade")
)
corrs
```

```
# Correlation Matrix (pearson-method)

Parameter1      |   Parameter2 |    r |       95% CI | t(128) |         p
------------------------------------------------------------------------
Frequency.Total | Grade.Theory | 0.31 | [0.15, 0.46] |   3.75 | < .001***
Frequency.Total |  Final_grade | 0.51 | [0.37, 0.63] |   6.76 | < .001***
Grade.Theory    |  Final_grade | 0.45 | [0.30, 0.58] |   5.69 | < .001***

p-value adjustment method: Holm (1979)
Observations: 130
```

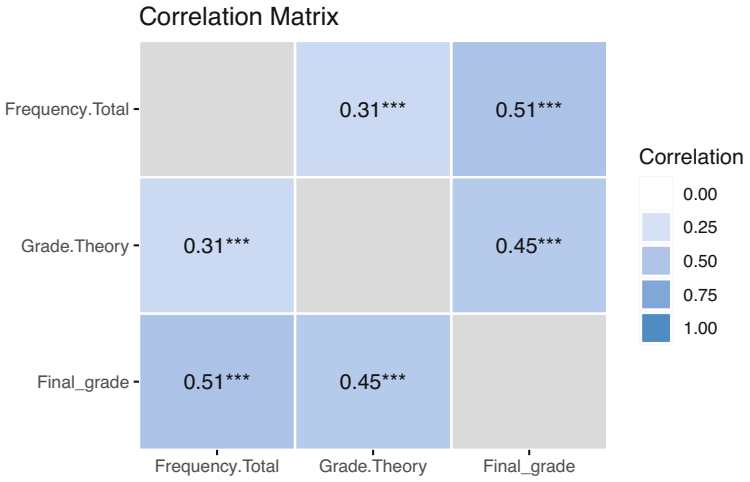
The columns `Parameter1` and `Parameter2` describe the variables that the correlation was computed for, `r` is the value of the sample correlation, and the remaining columns report the 95% confidence interval, the value of the test statistic, and the p-value of the test (a t-test for correlations) along with the statistical significance. By default, Pearson's correlation coefficient is calculated, but the package also supports many alternative correlation measures. The correlation coefficient to be computed can be selected with the argument `method` that has the value `"pearson"` by default. Selecting for example `method = "spearman"` would compute the Spearman correlation coefficient instead. We can also obtain a correlation matrix by using `summary()`

```
summary(corrs)
```

```
# Correlation Matrix (pearson-method)

Parameter       | Final_grade | Grade.Theory
-------------------------------------------
Frequency.Total |     0.51*** |      0.31***
Grade.Theory    |     0.45*** |

p-value adjustment method: Holm (1979)
```

By default, redundant correlations are omitted, but they can be obtained by setting `redundant = TRUE` in the call to `summary()`. A plot of the correlation matrix can be produced with the help of the package `see` [24].

```
library("see")
corrs |>
  # Also include redundant correlations
  summary(redundant = TRUE) |>
  plot()
```



The plot shows the strength of the correlations where darker colors imply stronger correlations. Visualizations will be covered at greater length in Chapter 6 [4].

5 Linear Regression

Linear regression is a statistical tool where one continuous variable is explained by the values of other variables. The variable of interest is said to be a dependent, while the other variables are called predictors. Predictors may also be called explanatory variables, independent variables or covariates depending on the context, applied field, and perspective.

Consider a very simple case, where we only have one predictor, which happens to be a continuous variable. In this case, fitting a linear regression model is merely the same as fitting a straight line to a scatterplot. It is assumed that deviations from this line are simply a result of random variation.

Now, let's go through the formal definition of a linear regression model. Let Y be a dependent variable with measurements $y_1 \ldots, y_n$, and let $X_1, X_2, \ldots, X_k$ be predictor variables with measurements $x_{1i}, \ldots, x_{ki}$ for all $i = 1, \ldots, n$ where i refers to an individual measurement. Then, the regression equation is

$$y_i = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki} + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma^2), \quad i = 1, \ldots, n$$

where we have the regression coefficients $\beta_0, \beta_1, \ldots \beta_k$ and the error variance σ^2. The first parameter β_0 is called the intercept that models the conditional expectation of Y when all the predictors have the value 0. From the regression equation, several assumptions become apparent. First, as the name of the model suggests, a linear relationship is assumed between the response and the predictors. Next, the variance σ^2 of the errors ε_i is constant, and does not depend on the values of the predictors (homoscedasticity). The errors are also assumed independent. The predictor variables are assumed fixed and their values perfectly measured without error.

Let's fit a linear regression model that predicts the final grade with the number of Moodle events of different types. To simplify the exposition, we will only use three types of Moodle events as predictors. We use the `lm()` function which has the same formula interface that we are already familiar with. First, we must define the dependent variable on the left-hand side of the formula, followed by the predictors on the right-hand side separated by a + sign. We must also supply the `data` argument, which tells the function where the actual values of the variables can be accessed.

```
fit <- lm(
  Final_grade ~ Frequency.Applications + Frequency.Assignment +
    Frequency.La_types,
  data = all_combined
)
summary(fit)
```

```
Call:
lm(formula = Final_grade ~ Frequency.Applications +
    Frequency.Assignment + Frequency.La_types, data = all_combined)

Residuals:
    Min      1Q  Median      3Q     Max
-7.0382 -0.8872  0.3665  1.2372  3.4422

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)           5.800211   0.405963  14.288  < 2e-16 ***
Frequency.Applications  0.076516   0.022294   3.432 0.000811 ***
Frequency.Assignment   -0.005049   0.005734  -0.881 0.380225
Frequency.La_types      0.088252   0.027314   3.231 0.001574 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.914 on 126 degrees of freedom
```

```
Multiple R-squared:  0.2559,    Adjusted R-squared:  0.2382
F-statistic: 14.45 on 3 and 126 DF,  p-value: 3.798e-08
```

The `summary()` function provides a compact overview of the model fit for `lm` objects. First, a summary of the residuals (i.e., the differences between the observed and predicted values) is provided under `Residuals`. Next, a summary of the regression coefficients β is provided under `Coefficients`, including their estimates (`Estimate`), standard errors (`Std. Error`), test statistics for t-tests that test whether the coefficients are significantly different from zero (`t value`), p-values of the tests (`Pr(>|t|)`) and statistical significance (indicated by the asterisks). For instance, we see that the number of group work events is statistically significant. The notation used for the significance levels of the tests is described following `Signif. codes`. Estimate of the square root of the error variance σ^2 is reported following `Residual standard error`. The two `R-squared` values are estimates of the proportion of variance of the data that is explained by the model. Finally, the `F-statistic` reports the results of ANOVA when applied with the same model formula that was used for the linear regression model.

The `report()` function provides a more comprehensive summary of the model fit and the regression coefficients:

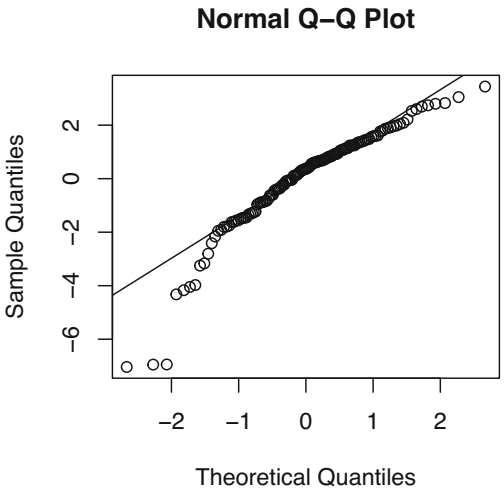
```
report(fit)
```

```
We fitted a linear model (estimated using OLS) to predict Final_grade with
Frequency.Applications, Frequency.Assignment and Frequency.La_types (formula:
Final_grade ~ Frequency.Applications + Frequency.Assignment + Frequency.La_types).
The model explains a statistically significant and moderate proportion of variance
(R2 = 0.26, F(3, 126) = 14.45, p < .001, adj. R2 = 0.24). The model's intercept,
corresponding to Frequency.Applications = 0, Frequency.Assignment = 0 and
Frequency.La_types = 0, is at 5.80 (95% CI [5.00, 6.60], t(126) = 14.29, p < .001).
Within this model:

  - The effect of Frequency Applications is statistically significant and positive
(beta = 0.08, 95% CI [0.03, 0.12], t(126) = 3.43, p < .001; Std. beta = 0.32, 95%
CI [0.13, 0.50])
  - The effect of Frequency Assignment is statistically non-significant and negative
(beta = -5.05e-03, 95% CI [-0.02, 6.30e-03], t(126) = -0.88, p = 0.380; Std. beta =
-0.08, 95% CI [-0.26, 0.10])
  - The effect of Frequency La types is statistically significant and positive (beta
= 0.09, 95% CI [0.03, 0.14], t(126) = 3.23, p = 0.002; Std. beta = 0.31, 95% CI
[0.12, 0.49])

Standardized parameters were obtained by fitting the model on a standardized
version of the dataset. 95% Confidence Intervals (CIs) and p-values were computed
using a Wald t-distribution approximation.
```

Again, the report output has condensed the information about the model fit into a format that can be read in a straightforward manner.

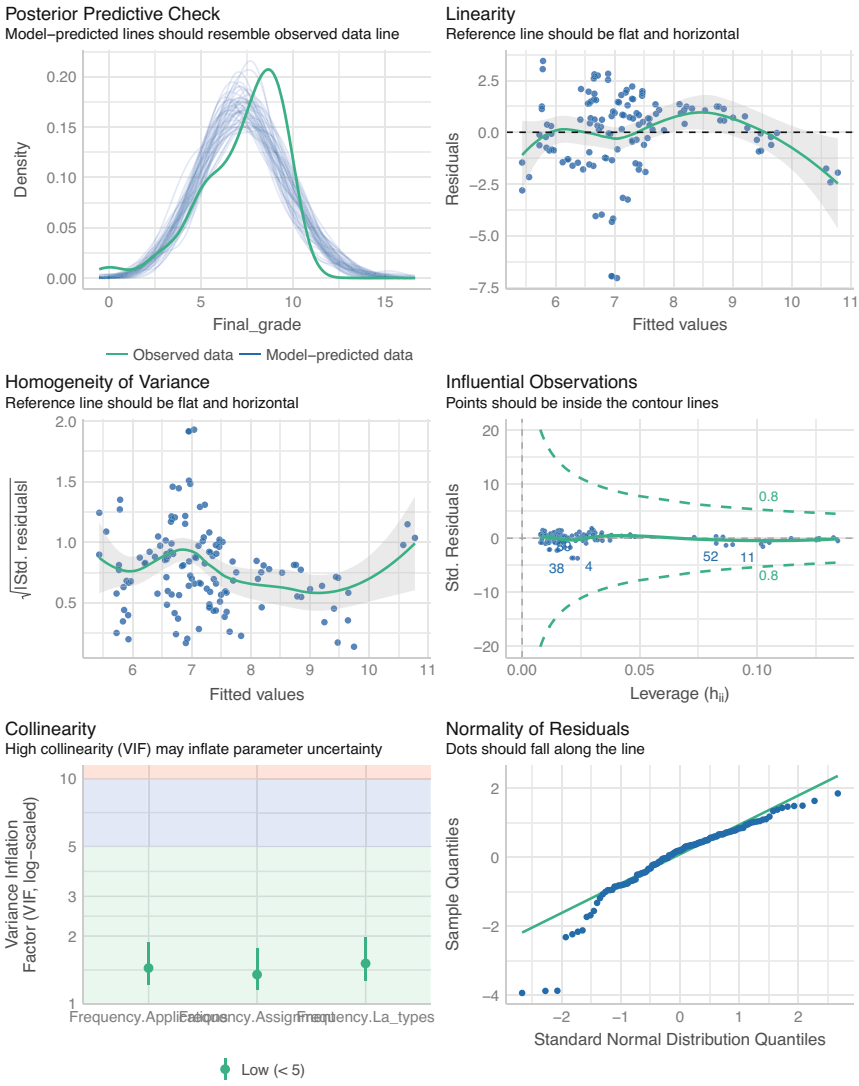
The assumption of normality of the residuals can be assessed with a quantile-quantile plot, or q-q plot for short. The residuals of the model fit can be accessed with the function `resid()`. The function `qqnorm()` draws the quantiles of the residuals against the quantiles of the normal distribution. The function `qqline()` adds a straight line through the plot that passes through the second and third quantiles, by default. Ideally, the residuals should fall on this line, and large deviations indicate that the normality assumption may not hold.

```
# Draw the quantiles of the residuals and the theoretical quantiles
qqnorm(resid(fit))
# Add a line through the theoretical quantiles
qqline(resid(fit))
```



The vast majority of residuals fall nicely onto the line for our model. Besides the q-q plot, we can obtain more model diagnostics with the help of the `performance` [25] package. This package provides a wide array of tools to assess how well models fit to the data. The general-purpose function `check_model(` provides a visual overview of the model fit using several metrics.

```
library("performance")
check_model(fit, theme = see::theme_lucid(base_size = 10))
```



The functions performs various tests related to the assumptions of the linear regression model. For example, the bottom right panel contains the same q-q plot that we previously constructed using the `qqnorm()` and `qqline()` functions. We refer the reader to the documentation of the `performance` package for more information on the remaining tests.

6 Logistic Regression

Logistic regression is a similar tool to linear regression, but with a binary outcome instead of a continuous one. Instead of modeling the outcome variable directly, a linear model is constructed for the logarithmic odds of the probability of "success" for the binary outcome, e.g., obtaining a passing grade. There is also no explicit error term ε in the model, as the uncertainty in the outcome is already captured by the success probability. Formally, the model is

$$\text{logit}\left(P(y_i = 1)\right) = \beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \cdots + \beta_k x_{ki}, \quad i = 1, \ldots, n,$$

where the logit-function is defined as $\text{logit}(x) = \log(x/(1-x))$. Here, the logit-function serves as the so-called *link function* that connects the expected value of the response to the predictors.

We fit a logistic regression model where the outcome variable is the level of achievement (`AchievingGroup`) and the predictors are the Moodle event counts of each type. The logistic regression model is a *generalized linear model*: a class of models that extend the linear regression model and that can be fitted in R with the function `glm()`. The syntax of `glm()` is analogous to `lm()`, but we must also specify the distribution of the outcome and the link function via the `family` argument. We use the function `binomial()` and supply the argument `link = "logit"` to define that the model should be a logistic regression model (in this case the `link` argument is optional, as `"logit"` is the default value). Because `AchievingGroup` is a factor, we must first convert it into a binary response that attains values 1 and 0 (or `TRUE` and `FALSE`). We can do this within the formula via the `I()` function, so that we do not have to modify our data. When used in a formula, this function will first compute its argument expression when evaluated, so that the expression is not mistaken for a variable name in the data (that does not exist). We select the high achievers as the "success" category for the outcome.

```
fit_logistic <- glm(
  # Use the I() function to construct a binary response in the formula
  I(AchievingGroup == "High achiever") ~ Frequency.Applications +
    Frequency.Assignment + Frequency.La_types,
  data = all_combined,
  # Our response is binary, so we use the binomial family with logit link
  family = binomial(link = "logit")
)
summary(fit_logistic)
```

```
Call:
glm(formula = I(AchievingGroup == "High achiever") ~ Frequency.Applications +
    Frequency.Assignment + Frequency.La_types, family = binomial(link = "logit"),
    data = all_combined)

Coefficients:
```

```
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)           -0.66272    0.62914  -1.053  0.29217
Frequency.Applications  0.30443    0.07778   3.914 9.07e-05 ***
Frequency.Assignment   -0.04477    0.01402  -3.193  0.00141 **
Frequency.La_types      0.12245    0.04710   2.600  0.00933 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 180.22  on 129  degrees of freedom
Residual deviance: 120.21  on 126  degrees of freedom
AIC: 128.21

Number of Fisher Scoring iterations: 6
```

The summary of a `glm()` function output is very similar to the output of a `lm()` summary. First, the `Call` is reported, which simply restates how the model was fitted. Next, `Coefficients` reports the estimates of the regression coefficients β, and their standard errors and statistical significance. Lastly, two deviance measures and their degrees of freedom are reported. The null deviance is twice the difference between the log-likelihood of the saturated model and the null model, and residual deviance is twice the difference between the saturated model and the model that was fitted. In simpler terms, the saturated model is a perfect model in a sense that there is a parameter for each observation. Conversely, the null model only has the intercept term. The deviance serves as a generalization of the residual sum of squares of the linear regression model, and it can be used to assess the quality of the model fit [26].

The `report()` function is applicable to models fitted with `glm()`.

```
report(fit_logistic)
```

```
We fitted a logistic model (estimated using ML) to predict AchievingGroup with
Frequency.Applications, Frequency.Assignment and Frequency.La_types (formula:
I(AchievingGroup == "High achiever") ~ Frequency.Applications +
Frequency.Assignment + Frequency.La_types). The model's explanatory power is
substantial (Tjur's R2 = 0.38). The model's intercept, corresponding to
Frequency.Applications = 0, Frequency.Assignment = 0 and Frequency.La_types = 0, is
at -0.66 (95% CI [-1.94, 0.56], p = 0.292). Within this model:

  - The effect of Frequency Applications is statistically significant and positive
(beta = 0.30, 95% CI [0.17, 0.48], p < .001; Std. beta = 1.62e-14, 95% CI
[-73784.14, 73784.14])
  - The effect of Frequency Assignment is statistically significant and negative
(beta = -0.04, 95% CI [-0.07, -0.02], p = 0.001; Std. beta = -9.05e-16, 95% CI
[-71374.92, 71374.92])
  - The effect of Frequency La types is statistically significant and positive (beta
= 0.12, 95% CI [0.04, 0.22], p = 0.009; Std. beta = -2.52e-17, 95% CI [-75547.24,
75547.24])
```

```
Standardized parameters were obtained by fitting the model on a standardized
version of the dataset. 95% Confidence Intervals (CIs) and p-values were computed
using a Wald z-distribution approximation.
```

The output describes succinctly the model that was fitted, and the effects of the predictors on the response. The `performance` package is also applicable to models fitted with the `glm()` function.

```
check_model(fit_logistic)
```

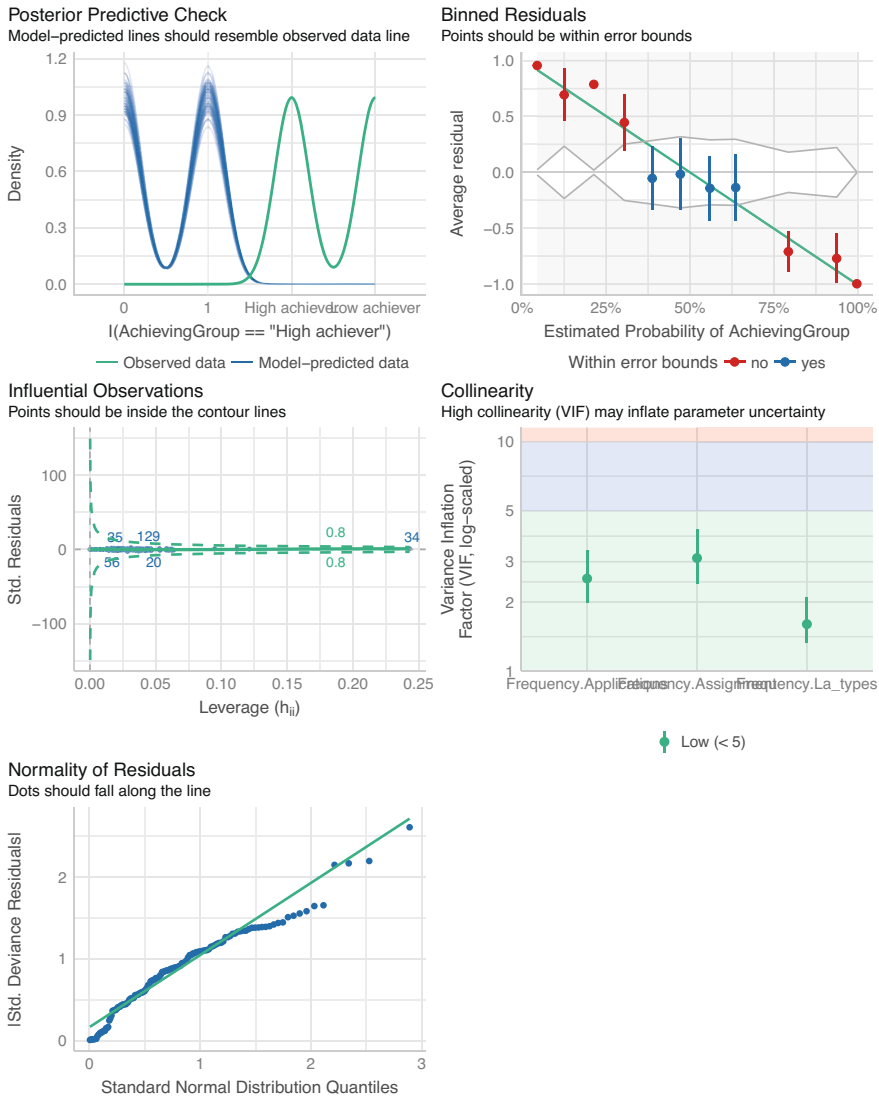


Table 1 Summary of the statistical tests, their null hypotheses, and the number of groups they compare simultaneously

Test	Null Hypothesis	Groups	R function
Student's t-test	Equal means	One, two or paired	`t.test()`
Chi-squared test	Independence	One	`chisq.test()`
Fisher's test	Independence	One	`fisher.test()`
ANOVA	Equal means	Two or more	`aov()`
Kruskal-Wallis test	Equal medians	Two or more	`kruskal.test()`
Levene's test	Homoscedasticity	Two or more	`leveneTest()`
Shapiro-Wilk test	Normality	One	`shapiro.test()`

We note that a different set of diagnostic checks is carried out for the logistic regression model compared to the linear regression model. For example, there is no assumption of homoscedasticity of variance as there is no explicit error term ε in the model. Again, we refer the reader to the documentation of the `performance` package for more details on these checks.

7 Conclusion

Basic statistics are an essential component of learning analytics. Learning analytics involves the collection, analysis, and interpretation of data related to the learning process, and statistical methods are used to identify patterns and trends in this data and to draw conclusions. Basic descriptive statistics such as measures of central tendency, variability and correlation are crucial for analyzing, interpreting, and visualizing data. Understanding these concepts is important for anyone involved in conducting research with quantitative data in the field of learning analytics. Moreover, mastery of basic statistics can facilitate the comprehension of more advanced statistical methods that are commonly used in learning analytics, such as logistic regression and cluster analysis. Table 1 contains a summary of the statistical tests that were introduced in this chapter.

We emphasize that when using any statistical test or a statistical model, it is important to keep the various assumptions related to the chosen method in mind, and to assess them beforehand whenever possible. If the assumptions are violated, the results of the method may not be reliable, and thus suitable alternatives should be considered.

8 Further Reading

This chapter scratched the surface of the full features of packages such as `correlation`, `report` and `performance` that can streamline the statistical

analysis and reporting process. We refer the reader to the documentation of these packages to gain a more thorough understanding of their features. These packages are part of a package collection called *easystats* [27] (https://github.com/easystats/easystats). There are several other packages in this collection that were not discussed in this chapter that can be useful for R users working with learning analytics. The book "Learning Statistics with R" by Danielle Navarro is freely available online and provides a comprehensive introduction to statistics using R (https://learningstatisticswithr.com/). For a general introductory text to statistical methods and inference, see e.g., [2].

References

1. Niiniluoto I (1980) Johdatus tieteenfilosofiaan: Käsitteen- ja teorianmuodostus. Otava
2. Ross SM (2010) Introductory statistics, 4th edn. Elsevier, Amsterdam
3. Lehmann EL, Romano JP (2022) Testing statistical hypotheses, 4th edn. Springer, Cham
4. López-Pernas S, Misiejuk K, Tikka S, Saqr M, Kopra J, Heinäniemi M (2024) Visualizing and reporting educational data with R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, Berlin
5. López-Pernas S, Saqr M, Conde J, Del-Río-Carazo L (2024) A broad collection of datasets for educational research training and application. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, Berlin
6. Chan C, Chan GC, Leeper TJ, Becker J (2021) rio: a swiss-army knife for data file I/O. https://cran.r-project.org/package=rio
7. Wickham H, François R, Henry L, Müller K, Vaughan D (2023) dplyr: a grammar of data manipulation
8. Wickham H, Vaughan D, Girlich M (2023) tidyr: tidy messy data
9. Kopra J, Tikka S, Heinäniemi M, López-Pernas S, Saqr M (2024) An R approach to data cleaning and wrangling for education research. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, Berlin
10. Makowski D, Lódecke D, Patil I, Thériault R, Ben-Shachar MS, Wiernik BM (2023) Automated results reporting as a practical tool to improve reproducibility and methodological best practices adoption. CRAN
11. Gosset W"Student"S (1908) The probable error of a mean. Biometrika 6:1–25. https://doi.org/10.1093/biomet/6.1.1
12. Welch BL (1947) The generalization of "student's" problem when several different population variances are involved. Biometrika 34:28–35. https://doi.org/10.1093/biomet/34.1-2.28
13. Ellis PD (2010) The essential guide to effect sizes: statistical power, meta-analysis, and the interpretation of research results. Cambridge University Press, Cambridge
14. Cohen J (1988) Statistical power analysis for the behavioral sciences. Routledge, New York
15. Pearson K (1900) On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. Philos Mag Ser 5 50:157–175. https://doi.org/10.1080/14786440009463897
16. Fisher RA (1922) On the interpretation of χ^2 from contingency tables, and the calculation of P. J R Stat Soc 85:87–94
17. Fisher RA (1921) On the "probable error" of a coefficient of correlation deduced from a small sample. Metron 1:3–32
18. Kruskal WH, Wallis WA (1952) Use of ranks in one-criterion variance analysis. J Am Stat Assoc 47:583–621. https://doi.org/10.1080/01621459.1952.10483441

19. Levene H (1960) Robust tests for equality of variances. In: Olkin I, Hotelling H, et al (eds) Contributions to probability and statistics: essays in honor of harold hotelling. Stanford University Press, Redwood City, pp 278–292
20. Bartlett MS (1937) Properties of sufficiency and statistical tests. Pro R Stat Soc Ser A 160:268–282
21. Shapiro SS, Wilk MB (1965) An analysis of variance test for normality (complete samples). Biometrika 52:591–611. https://doi.org/10.1093/biomet/52.3-4.591
22. Robinson D, Hayes A, Couch S (2023) broom: convert statistical objects into tidy tibbles
23. Makowski D, Ben-Shachar MS, Patil I, Lüdecke D (2020) Methods and algorithms for correlation analysis in R. J Open Source Softw 5:2306. https://doi.org/10.21105/joss.02306
24. Lüdecke D, Patil I, Ben-Shachar MS, Wiernik BM, Waggoner P, Makowski D (2021) See: an R package for visualizing statistical models. J Open Source Softw 6:3393. https://doi.org/10.21105/joss.03393
25. Lüdecke D, Ben-Shachar MS, Patil I, Waggoner P, Makowski D (2021) Performance: an R package for assessment, comparison and testing of statistical models. J Open Source Softw 6:3139. https://doi.org/10.21105/joss.03139
26. Dobson AJ (1990) An introduction to generalized linear models. Chapman Hall, Boca Raton
27. Lüdecke D, Ben-Shachar MS, Patil I, Wiernik BM, Bacher E, Thériault R, Makowski D (2022) easystats: framework for easy statistical modeling, visualization, and reporting. CRAN

Visualizing and Reporting Educational Data with R

Sonsoles López-Pernas, Kamila Misiejuk, Santtu Tikka, Juho Kopra, Merja Heinäniemi, and Mohammed Saqr

1 Introduction

Data visualization can be defined as "the representation and presentation of data that exploits our visual perception abilities in order to amplify cognition" [1]. It has the power to transform complex information into stories that inform and inspire action. Data visualization is an effective tool for learning analytics, as it helps to present learners' data in a way that is easily understandable and intuitive for students, teachers, researchers, and other stakeholders. Through the use of graphs, charts, and other visual aids, it is possible to quickly identify patterns, trends, and relationships within data that may not be immediately apparent through purely numerical data analysis methods.

Visualization in learning analytics has two distinct applications. On the one hand, the use of visual dashboards has become the main vehicle for putting learning analytics into practice. Presenting data in visually appealing and intuitive ways can help promote data literacy among students and other stakeholders, encouraging

S. López-Pernas (✉) · J. Kopra
School of Computing, University of Eastern Finland, Joensuu, Finland
e-mail: sonsoles.lopez@uef.fi

K. Misiejuk
Centre for the Science of Learning & Technology (SLATE), University of Bergen, Bergen, Norway

S. Tikka
Department of Mathematics and Statistics, University of Jyväskylä, Jyväskylä, Finland

M. Heinäniemi
Institute of Biomedicine, University of Eastern Finland, Kuopio, Finland

M. Saqr
School of Computing, University of Eastern Finland, Joensuu, Finland

M. Saqr, S. López-Pernas (eds.), *Learning Analytics Methods and Tutorials*,
https://doi.org/10.1007/978-3-031-54464-4_6

greater engagement with data and fostering a culture of continuous improvement. On the other hand, learning analytics scientific production heavily relies on data visualization to present research findings in a clear and accessible manner, making it easier for readers from different scholarly backgrounds to understand and act upon research insights. Regardless of the context, the power of visualization in learning analytics lies in its ability to take complex data and turn it into meaningful insights that support better decision-making and drive improvement.

In this chapter, the reader will be guided through the process of generating meaningful and aesthetically pleasing visualizations of different types of datasets using well-known R packages. Relevant plots and plot types will be demonstrated with an explanation of their usage and usage cases. Furthermore, learning-related examples will be discussed in detail. For instance, readers will learn how to visualize learners' logs extracted from learning management systems (LMSs) to show how trace data can be used to track students' learning activities. Other examples of common research scenarios in which learners' data are visualized will be illustrated throughout the chapter. In addition to creating compelling plots, readers will also be able to generate professional-looking tables with summary statistics to report descriptive statistics.

2 Visualization in Learning Analytics

Developing visualizations is a challenging task of balancing the cognitive load of users while not compromising on conveying specific insights from data [2]. Visualizations for practice in learning analytics are mostly developed for two main stakeholders: learners and instructors. Depending on the target group, a visualization or a dashboard (i.e., a collection of visualizations depicting multiple indicators) have different goals.

Learner-facing visualizations are meant to make learners aware of their own learning and to provide them with actionable feedback on their learning. Visualizations display learners' performance on a specific metric and compare it with a reference frame: other peers, desirable learning achievement, or their own progress over time [3]. Sense-making questions triggering reflection can be added to a visualization [4, 5], or some elements of the visualizations can be highlighted and described in words using layered storytelling [6, 7]. Another option is to gamify a dashboard, for example, by using badges [8]. To provide feedback to learners, visualizations can be augmented with links to recommended resources [9], information about specific topics to review to close the achievement gap [6], or explanations of the meaning of visualizations and their implications for the learner [10]. Current learner-facing dashboards mostly show resource use and assessment data [11], compare learners to their peers [12], display descriptive analytics rather than predictive or prescriptive analytics [10], and use self-regulated learning theory as their framework [12, 13]. Some reviews found a positive effect on student outcomes [10], while others reported mixed results [11, 14]. Showing

visualizations to learners can change their behavior. For example, social network analysis visualizations have resulted in fewer cross-group commenting [15], while a visualization comparing individual submission patterns with the top 25% of students in a class led to earlier homework submissions [16].

In comparison, the goal of instructor-facing visualizations is to support teachers and their decision-making process by tracking student progress. Two main types can be distinguished. Mirroring or descriptive visualizations provide insights about the learners on an aggregated or an individual level using either descriptive or comparative data. Advising or prescriptive visualizations show not only information about the learners but also alert the instructor to undertake a pedagogical action [17, 18]. Current instructor-facing visualizations mostly display course-wide information about the learners or track group work [14]. These visualizations can support teachers in facilitating student collaboration [19], planning and collecting student feedback on learning activities [20], or obtaining insights into student interactions within an online environment, such as simulations, virtual labs or online games [21, 22]. However, interpreting dashboard information is a challenging task for instructors. Although some teachers use dashboards as complementary sources of information, others act based only on the dashboard information without further investigation [23].

A common point of criticism of learning analytics dashboards is that most of them are not grounded in learning theories [13, 14]. Data-driven evaluations of dashboards focused on dashboard acceptance, usefulness, or usability are more prevalent than pedagogically-focused evaluations [24]. Some approaches were developed to mitigate these issues. The model of user-centered learning analytics systems (MULAS) presents a set of recommendations on four interconnected dimensions: theory, design, evaluation, and feedback, and can be used to guide dashboard development [14]. Another approach is an iterative five-stage Learning Awareness Tools—User eXperience (LATUX) workflow, including problem identification, low-fidelity prototyping, high-fidelity prototyping, pilot studies, and classroom use, that can be used to develop visual analytics [25]. Finally, open learner model research could be used as a source of insights while developing learning analytics visualizations, such as dashboards [9].

3 Generating Plots with `ggplot2`

In the previous section, we have seen how central visualization is to learning analytics. In the remainder of the chapter, we will learn how to create different types of visualizations that are relevant to different types of data related to teaching and learning. We will mostly rely on `ggplot2`, a popular data visualization package in R that was developed by Hadley Wickham [26]. It is based on the grammar of graphics [27], which is a systematic way of thinking about and constructing visualizations. The `ggplot2` library provides a flexible and intuitive framework for creating a wide range of graphics, from basic scatter plots to complex visualizations with multiple

layers. It is known for its ability to produce visually appealing and informative graphics with relatively few lines of code. It enables users to define aesthetics, such as color and size, and add layers, such as points and lines, to create customized and interactive plots. In addition, `ggplot2` allows for easy customization of plot features, such as titles, axis labels, and legends.

Overall, `ggplot2` is a powerful and versatile tool for data visualization in R, and is widely used by data scientists, statisticians, and researchers in a variety of fields. In this chapter, we will cover the fundamental concepts and techniques of `ggplot2`, including how to create basic plots, and customize their appearance. We will start by introducing the building blocks of a `ggplot2` plot, including aesthetics, layers, and scales. Then, we will create a plot from scratch step by step, showing how to customize its appearance, including how to change theme, colors, and scales. We will then explore the different types of plots that can be created with `ggplot2`, such as scatter plots, bar charts, and histograms.

Throughout this section, we will use datasets of students' learning data to demonstrate how to create effective visualizations for learning analytics with `ggplot2`. Please, refer to Chapter 2 of this book [28] to learn more about the datasets used. By the end of this section, you will have a solid foundation in `ggplot2` and be able to create basic, yet compelling visualizations to explore your data.

3.1 The `ggplot2` Grammar

The `ggplot2` library is based on Wilkinson's grammar of graphics [27]. The main idea is that every plot can be broken down into a set of components, each of which can be customized and combined in a flexible way. These components are:

- **Data**: This is the data we want to visualize. It can be in the form of a dataframe, tibble or any other structured data format.
- **Aesthetic mapping** (`aes`): It defines how variables in the data are mapped to visual properties of the plot, such as position, color, shape, size, and transparency.
- **Geometric object** (`geom`): It represents the actual visual elements of the plot, such as points, lines, bars, and polygons.
- **Statistical transformation** (`stat`): It summarizes or transforms the data in some way, such as by computing means, medians, or proportions, or by smoothing or summarizing data, or grouping them into bins.
- **Scale** (`scale`): It maps values in the data to visual properties of the plot, such as color, size, or position.
- **Coordinate system** (`coord`): It defines the spatial or geographic context in which the plot is displayed, such as Cartesian coordinates, polar coordinates, or maps.
- **Facet** (`facet`): It allows to split the data into subsets and display each subset in a separate panel. It often useful for visualizing data with multiple categories or groups.

Through the combination and customization of these components, we can create a wide variety of complex and informative visualizations in `ggplot2`. The idea behind the graphics grammar is to provide a consistent framework for constructing plots, allowing users to focus on the data and the message they want to convey, rather than on the technical details of the visualization. In the following section, we will create a plot from scratch step by step to become familiar with the most relevant components.

3.2 Creating Your First Plot

We will now create our first plot using `ggplot2`. Our example deals with a widely studied matter in learning analytics, which is the relationship between online activity and achievement. We will use a bar chart to represent the number of students that have low, moderate and high activity levels in each achievement group (high achievers vs. low achievers). In order to become familiar with the syntax of `ggplot2`, we will recreate the plot step by step, explaining each of the elements in the plot. Below is the final result we aim at accomplishing (Fig. 1):

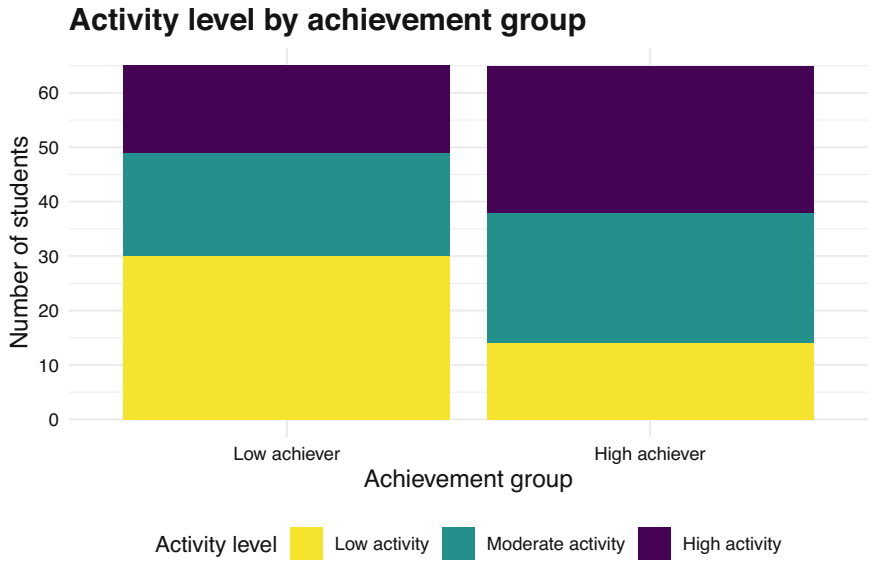


Fig. 1 First plot with ggplot2

3.2.1 Installing ggplot2

Our first step is installing the `ggplot2` library. This is usually the first step in any R script that makes use of external libraries.

```
install.packages("ggplot2")
```

To import `ggplot2` we just need to use the `library` command and specify the `ggplot2` library:

```
library(ggplot2)
```

3.2.2 Downloading the Data

Next, we need to import the data that we are going to plot. For this chapter, we are using synthetic data from a blended course on learning analytics. For more details about this dataset, refer to Chap. 2 in this book. The data is in Excel format. We can use the library `rio` since it makes it easy to read data in several formats. We first install the library:

```
install.packages("rio")
```

And import it so we can use its functions:

```
library(rio)
```

Now we can download the data using the `import` function from `rio` and assign it to a variable named `df` (short for dataframe).

```
demo_url =
"https://github.com/lamethods/data/raw/main/1_moodleLAcourse/AllCombined.xlsx"
df <- import(demo_url)
```

We can use the `head` command to get an idea of what the dataset looks like. To recreate the plot above we will need the `AchievingGroup` column —which indicates whether students' are high achievers (to 50%) or low achievers (bottom 50%), according to their final grade— and the `ActivityGroup` column —which indicates whether students have a high level of activity (top 33%), moderate activity (middle 33%), or low activity (bottom 33%), according to their total number of events in the LMS.

```
head(df)
```

```
# A tibble: 130 x 37
   User      Name   Gender ActivityGroup AchievingGroup Surname Origin Birthdate
   <chr>     <chr>  <chr>  <chr>         <chr>          <chr>   <chr>  <chr>
 1 00a05cc62 Wan    M      Low activity  Low achiever   Tan     Malay~ 12.12.19~
 2 042b07ba1 Daniel M      High activity Low achiever   Tromp   Aruba  28.5.1999
 3 046c35846 Sarah  F      Low activity  Low achiever   Schmit  Luxem~ 25.4.1997
 4 05b604102 Lian   F      Low activity  Low achiever   Abdull~ Yemen  19.11.19~
 5 0604ff3d3 Nina   F      Low activity  Low achiever   Borg    Malta  13.6.1994
 6 077584d71 Moham~ M      High activity High achiever  Gamal   Egypt  13.7.1998
 7 081b100cf Maxim~ M      Moderate act~ High achiever  Gruber  Austr~ 20.12.19~
 8 0857b3d8e Hugo   M      High activity High achiever  Pérez   Spain  22.12.19~
 9 0af619e4b Aylin  F      Low activity  Low achiever   Barat   Kazak~ 14.8.1995
10 0ec99ce96 Polina F      Moderate act~ Low achiever   Novik   Belar~ 9.10.1996
# i 120 more rows
# i 29 more variables: Location <chr>, Employment <chr>,
#   Frequency.Applications <dbl>, Frequency.Assignment <dbl>,
#   Frequency.Course_view <dbl>, Frequency.Feedback <dbl>,
#   Frequency.General <dbl>, Frequency.Group_work <dbl>,
#   Frequency.Instructions <dbl>, Frequency.La_types <dbl>,
#   Frequency.Practicals <dbl>, Frequency.Social <dbl>, ...
```

3.2.3 Creating the Aesthetic Mapping

Now that we have our data, we can pass it on to ggplot2 as follows:

```
ggplot(df)
```


Fig. 2 Empty plot

We still do not see anything because we have not selected the type of chart or the variables of the data that we want to plot (Fig. 2). First, let us specify that we want to plot the AchievingGroup column (high vs. low achievers) on the x-axis. Assigning columns of our dataset to different elements of the plot is called constructing an aesthetic mapping. We can do it by calling the aes function from ggplot2, specifying that we want to map the AchievingGroup column to the x-

axis, and then passing this call to `aes` to our plot using the second argument of `ggplot`:

3.2.4 Add the Geometry Component

```
ggplot(df, aes(x = AchievingGroup))
```

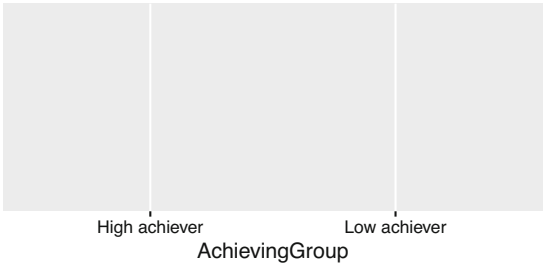


Fig. 3 Empty plot with AchievingGroup in x-axis labels

We now see that the x-axis has the two possible values of `AchievingGroup`: "High achiever" and "Low achiever" (Fig. 3). We still need to tell `ggplot2` the type of chart we want to use to plot the number of students of each type. To do that we need to add a geometrical (`geom`) component to our plot in which we specify that we want a bar chart. We do it by adding a `+` sign after our call to `ggplot` and calling `geom_bar()` (the name of the geometry that represents a bar chart).

```
ggplot(df, aes(x = AchievingGroup)) + geom_bar()
```

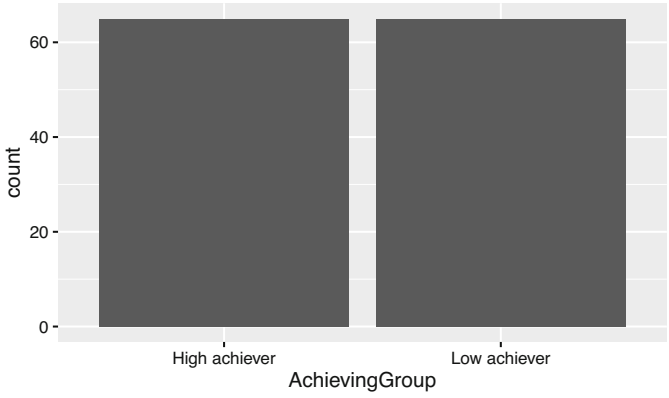


Fig. 4 Basic bar plot showing students by achievement group

Now the plot can actually be called a plot. Notice that we have not specified what we want to plot in the y-axis. When not specified, `ggplot2` assumes that we want to use the count of rows (Fig. 4).

We also notice that the bars are in the wrong order. By default, `ggplot2` orders the values in an ascending way (alphabetically in the case of text values). If we want to enforce our own order, we need to convert the `AchievingGroup` column of `df` into a factor and provide the ordered list of values to the `levels` argument.

```
df$AchievingGroup = factor(df$AchievingGroup,
                           levels = c("Low achiever", "High achiever"))
```

If we generate our plot again, we see that the bars are now in the order we want them to be (Fig. 5):

```
ggplot(df, aes(x = AchievingGroup)) + geom_bar()
```

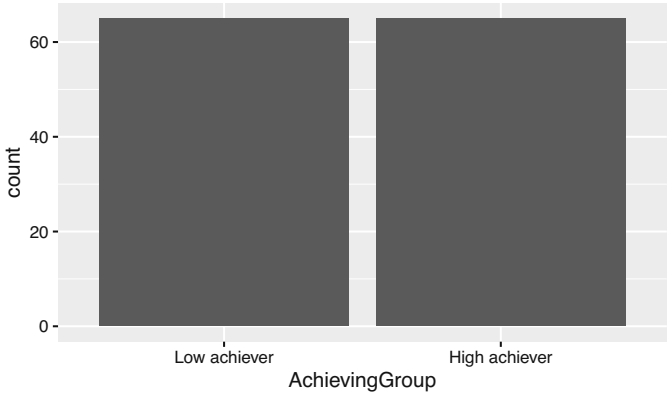


Fig. 5 Basic bar plot showing students by achievement group after transforming the x-axis variable into a factor

3.2.5 Adding the Color Scale

We still need to color our bar chart according to students' activity level. We do that by mapping the fill aesthetic to the `ActivityLevel` column inside the `aes`. When we provide the `fill` property, `ggplot` will automatically create the appropriate legend (Fig. 6).

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) + geom_bar()
```

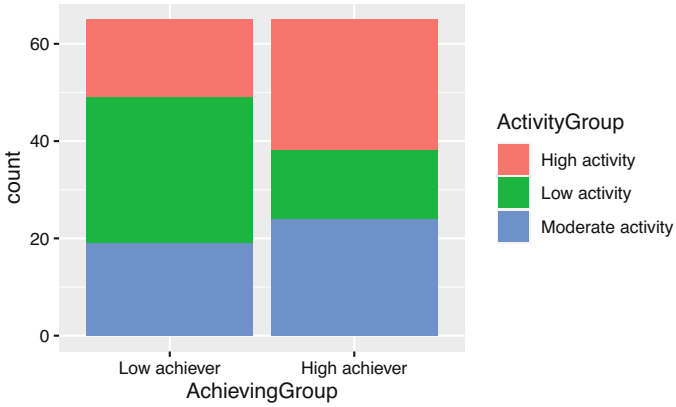


Fig. 6 Basic bar plot showing students' activity level by achievement group and colored by activity level

Again, we need to change the order of our legend so that it follows the logical semantic order for the activity levels (low-moderate-high):

```
df$ActivityGroup = factor(df$ActivityGroup,
                          levels = c("Low activity", "Moderate activity",
                          "High activity"))
```

If we generate the plot again, we see now that the legend is in the right order (Fig. 7):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) + geom_bar()
```

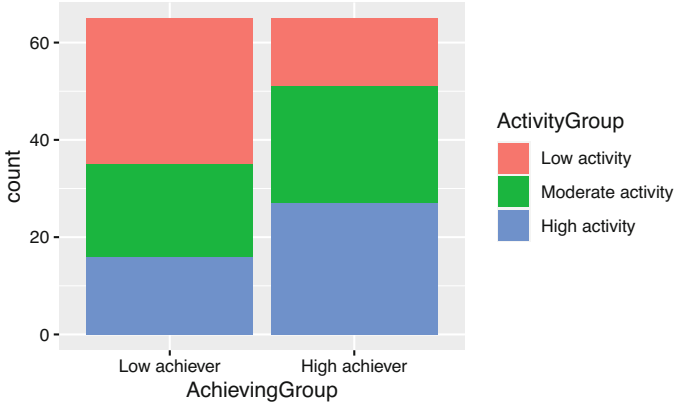


Fig. 7 Basic bar plot showing students' activity level by achievement group and colored by activity level after ordering the legend

However, the stacks are still not in the right order, being the low activity students at the top of the bar, and the high activity students at the bottom, which might be counter-intuitive. To change this, we need to reverse the position of the bar using `position = position_stack(reverse = TRUE)` inside `geom_bar` (Fig. 8):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE))
```

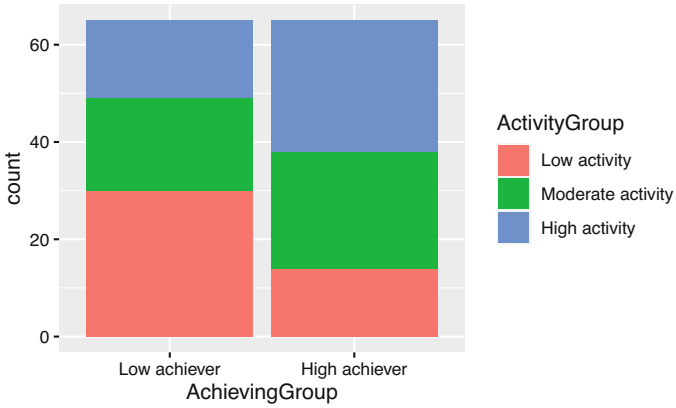


Fig. 8 Basic bar plot showing students' activity level by achievement group and colored by activity level after ordering the stacks

We are getting closer but the color scheme does not quite match our intended result. To add a color scheme to our plot we need to add a `scale` layer. In this case, the scale is for the `fill` property, which is the color of the bars in our chart. There are many ways to specify the color scheme. One option is to use sequential colors from the same palette. For that we add a new layer to our plot named `scale_fill_brewer` and we pass the palette that we want as an argument. For example, palette number 15 would look like this (Fig, 9):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_brewer(palette = 15)
```

Another option is to provide a manual scale with the colors of our choice. For that we use `scale_fill_manual` and specify a `values` vector as an argument. We need to specify as many colors as unique elements in your scale. In this case we have three activity groups (for low, moderate or high activity), so we must provide three colors. There are tons of resources online where you can find or create your own palettes (e.g., Coolors, Adobe Color or Lospec). You have to provide the

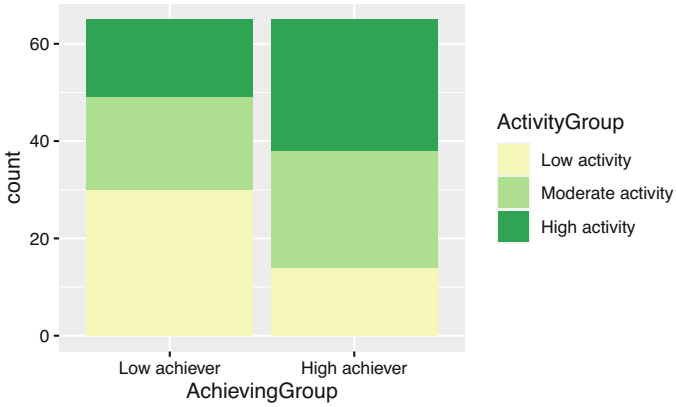


Fig. 9 Bar plot showing students' activity level by achievement group with sequential color scale

hexadecimal code of each color or the official color name recognized by R. Below is an example (Fig. 10):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_manual(values = c("#ef6461", "#7AE7C7", "#8E518D"))
```

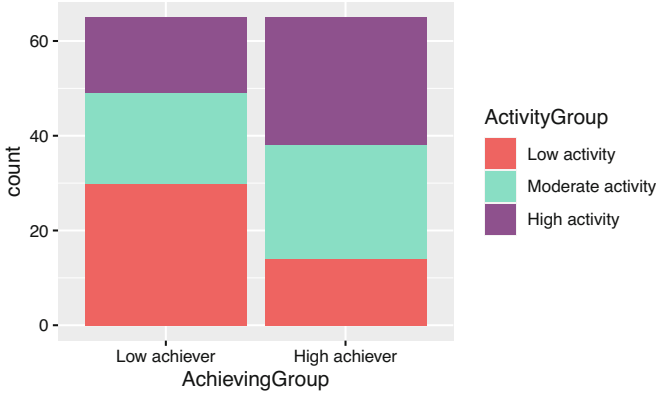


Fig. 10 Bar plot showing students' activity level by achievement group with manual color scale

Lastly, a very common color scale used is *Viridis*. It is designed to be perceived by viewers with common forms of color blindness. To use it in our plot we just add `scale_fill_viridis_d()` (Fig. 11).

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d()
```

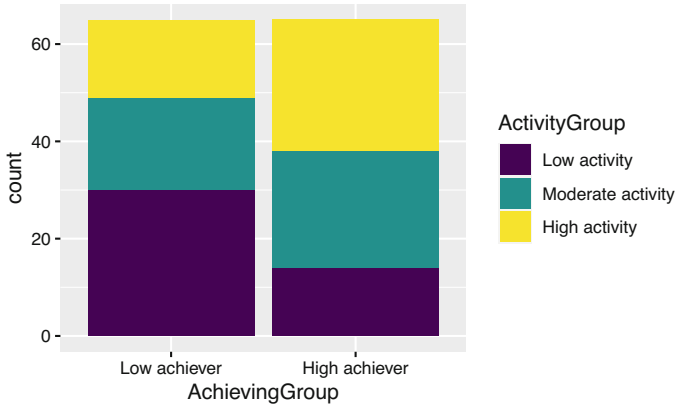


Fig. 11 Bar plot showing students' activity level by achievement group with viridis color scale

Viridis is the palette we need to replicate our target plot. However, the order of the color needs to be reversed so the most dense color represents the higher activity level. We do this by reversing the direction of the palette as follows (Fig. 12):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1)
```

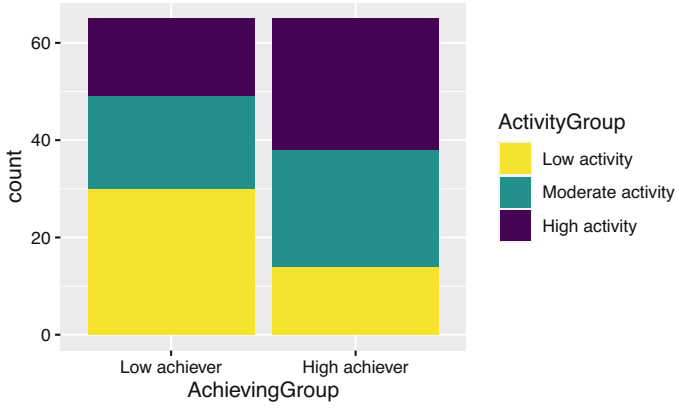


Fig. 12 Bar plot showing students' activity level by achievement group with viridis color scale

3.2.6 Working with Themes

Now that the geometry and color scheme of the bars looks like our initial plot, we notice that there are still some differences. An important one is the grey background of the plot. To change the general appearance of our plot, we may use the `ggplot2` themes. Below are some examples (Fig. 13):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1) + theme_dark()
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1) + theme_classic()
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1) + theme_void()
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1) + theme_minimal()
```

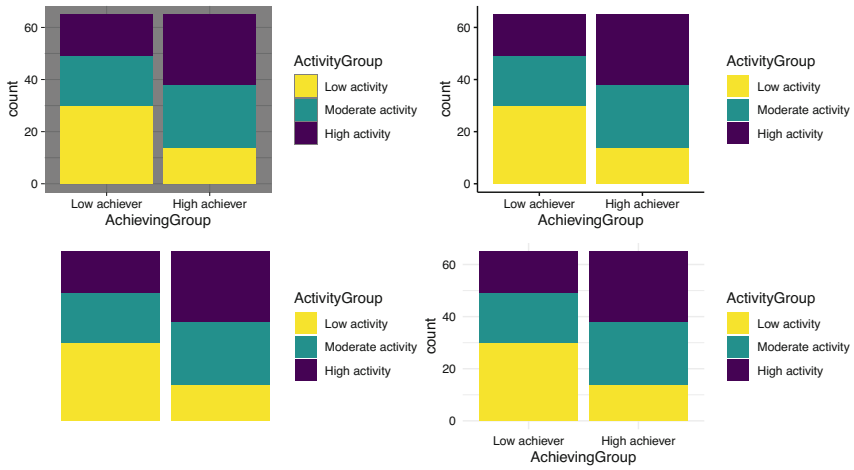


Fig. 13 Bar plot using different themes: `theme_dark` (top left), `theme_classic` (top right), `theme_void` (bottom left), and `theme_minimal` (bottom right)

We have `theme_dark` with a dark background and border, `theme_classic` with thick axes and no grid lines, `theme_void` which is completely empty, and `theme_minimal` with a minimalistic look. There are more available in the `ggplot2`

documentation and even more third-party implementations. To recreate our goal plot, we select the `theme_minimal`. To avoid having to add the theme to all of our plots from now on, we can set a default theme for our whole project by using `theme_set`:

```
theme_set(theme_minimal())
```

Notice how now we get `theme_minimal` even when we do not specify it in our code (Fig. 14):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1)
```

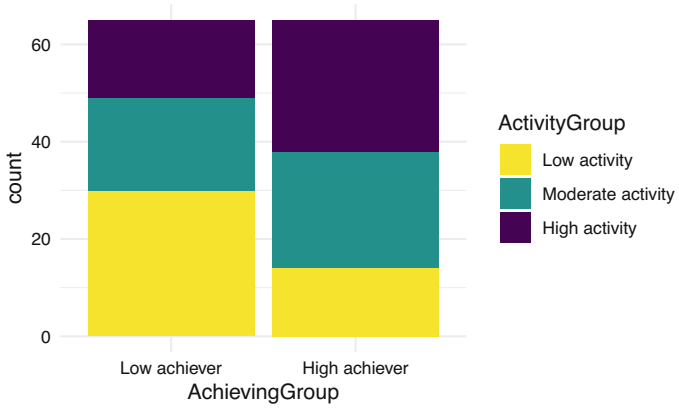


Fig. 14 Bar plot with theme `minimal` by default

3.2.7 Changing the Axis Ticks

You may have not noticed that another difference with our goal plot is the ticks in our y-axis. In the goal plot we count 10 by 10, whereas in our last plot we do so 20 by 20. Just like we modified the scale of the `fill` aesthetic when we changed the color of our bars, we can also modify the `y` aesthetic to adjust to our needs. We use the `scale_y_continuous` layer and we try different number of breaks (`n.breaks`), until we find what we like best (Fig. 15):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1) +
  scale_y_continuous(n.breaks = 15)
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1) +
  scale_y_continuous(n.breaks = 3)
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1) +
  scale_y_continuous(n.breaks = 7)
```

Fig. 15 Bar plot with different numbers of y.axis breaks: 15 (left), 3 (middle), and 7 (right)

We choose 7 breaks to obtain our desired result.

3.2.8 Titles and Labels

Our plot is still missing some slight modifications to be 100% equal to the original one. For instance, the axes' titles are not the same. To specify the y-axis label, we add a new layer to our plot named `ylab` and we pass a string with our desired label "Number of students" (Fig. 16):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1)  +
  scale_y_continuous(n.breaks = 7) +
  ylab("Number of students")
```

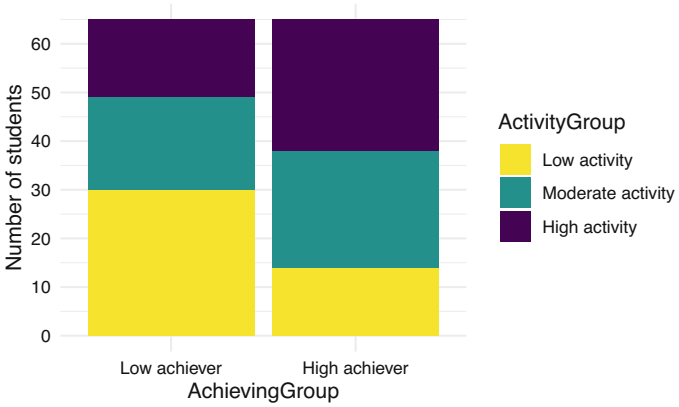


Fig. 16 Bar plot with y-axis label

We do the same for the x-axis using `xlab`, and for the legend using `labs` (Fig. 17):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1)  +
  scale_y_continuous(n.breaks = 7) +
  ylab("Number of students") +
  xlab("Achievement group") +
  labs(fill = "Activity level")
```

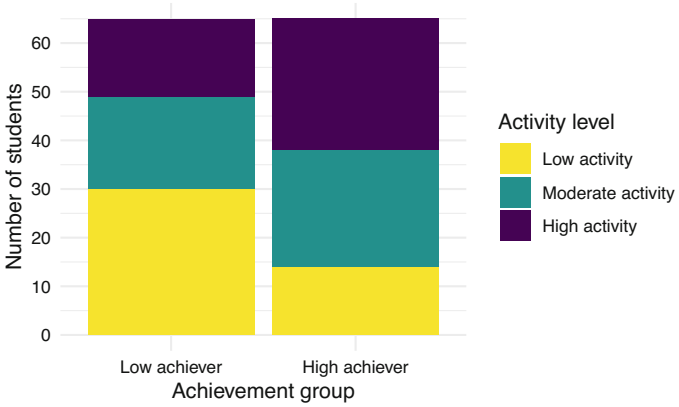


Fig. 17 Bar plot with all labels

More importantly, we are missing the overall title of the plot. To add it we use `ggtitle` and we pass our intended plot title "Activity level by achievement group". Keep in mind that, whenever possible, it is better to add a caption to the image rather than a title on the plot. A caption is more accessible for visually impaired users since it is compatible with screen readers. In scientific papers, it is also more common to have a Figure caption than a title within the plot. In social media, it is frequent to see the title on the plot as images are often shared without context. However, many social media platforms allow to provide an *alternative text* which is what screen readers will read as a substitute for the image, and that is also the case in learning analytics dashboards (Fig. 18).

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1)  +
  scale_y_continuous(n.breaks = 7) +
  ylab("Number of students") +
  xlab("Achievement group") +
  labs(fill = "Activity level") +
  ggtitle("Activity level by achievement group")
```

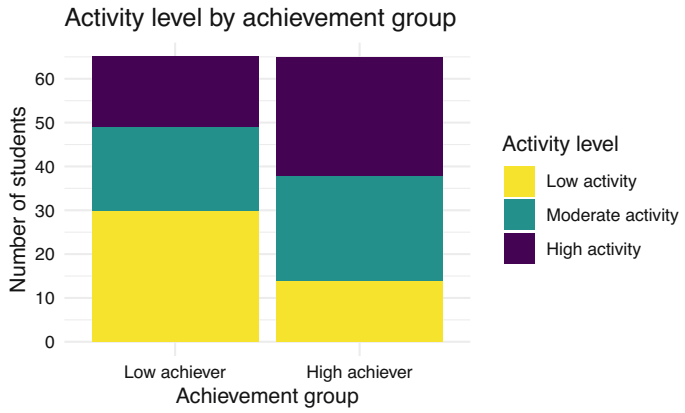


Fig. 18 Bar plot with title

3.2.9 Other Cosmetic Modifications

Lastly, we need to do some slight modifications to the overall appearance of the plot. We do this through the generic `theme` function of `ggplot2`. We first modify the position of the legend by setting `legend.position` to "bottom". We then increase the size of the axes titles, by setting `axis.title` to `element_text(size = 12)`. Finally, we make the plot title bigger as well and put it in bold by setting

`plot.title` to `element_text(size = 15, face = "bold")`). With these last changes, we have an exact replica of our original plot (Fig. 19).

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1) +
  scale_y_continuous(n.breaks = 7) +
  ylab("Number of students") +
  xlab("Achievement group") +
  labs(fill = "Activity level") +
  ggtitle("Activity level by achievement group") +
  theme(legend.position = "bottom",
        axis.title = element_text(size = 12),
        plot.title = element_text(size = 15, face = "bold"))
```

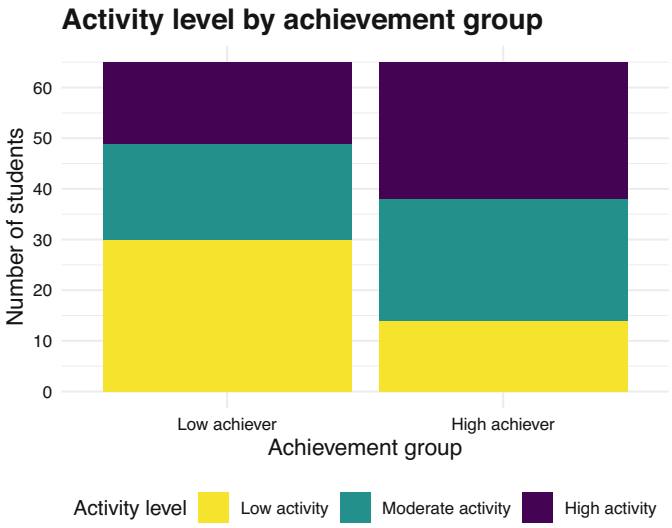


Fig. 19 Bar plot with theme modifications

3.2.10 Saving the Plot

Since we have obtained the desired result, we may now save it as an image to be able to use it elsewhere. For that, we first need to assign the plot to a variable (e.g., `myplot`).

```
myplot <- ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  geom_bar(position = position_stack(reverse = TRUE)) +
  scale_fill_viridis_d(direction = -1) +
  scale_y_continuous(n.breaks = 7) +
  ylab("Number of students") +
  xlab("Achievement group") +
  labs(fill = "Activity level") +
  ggtitle("Activity level by achievement group") +
  theme(legend.position = "bottom", axis.title = element_text(size = 12),
        plot.title = element_text(size = 15, face = "bold"))
```

We then use `ggsave` to save the plot to our filesystem. We need to specify the file path (including the extension, such as PNG, JPEG, etc.) where we want to save the plot (e.g., "bar.png") as the first argument and pass the variable where we saved our plot (`myplot`) as a second argument. If we do not do this, `ggplot2` assumes we want to save the latest plot that we created. Lastly, we may specify the width, height and resolution (dpi) of our plots. If we are submitting our figure to a scientific journal, we probably need a high resolution image. If we are using the figure in social media, we do not want the resolution to be so high as it would take a long time to load.

```
ggsave("bar.png", myplot, width = 10000, height = 5000, units = "px", dpi = 900)
```

Throughout this section, we have learned how we can create a plot from scratch using only the `ggplot2` library and a simple dataset. We have seen the many customization possibilities (theme, scales, titles) that we can achieve using the different plot components without needing to rely on external tools for retouching our final graph. In the next section we will learn about new types of plots that might be more suitable for other types of data and their customization possibilities.

3.3 Types of Plots

The `ggplot2` library offers many types of plots (or `geoms`) that you can choose from to visualize your data in several ways. In this section, we go over some of the most common types and present examples using students' learning data.

3.3.1 Bar Plot

We have seen how to construct a bar plot in the previous section as an example of how to use `ggplot2`. But when should we use a bar plot? Bar plots are useful when we want to represent counts or any numerical variable broken down by categories. The y-axis would represent the count (or other continuous numerical variable) and the x-axis would represent the categories. Keep in mind that if the categories follow a natural order, the x-axis should respect it (for example: "Morning", "Afternoon", "Evening"; or "Children", "Adults", "Elders"). Otherwise, you can just order the x-axis alphabetically or from highest to lowest value in the y-axis (Fig. 20).

```
ggplot(df, aes(x = AchievingGroup)) +
  geom_bar(position = position_stack(reverse = TRUE))
```

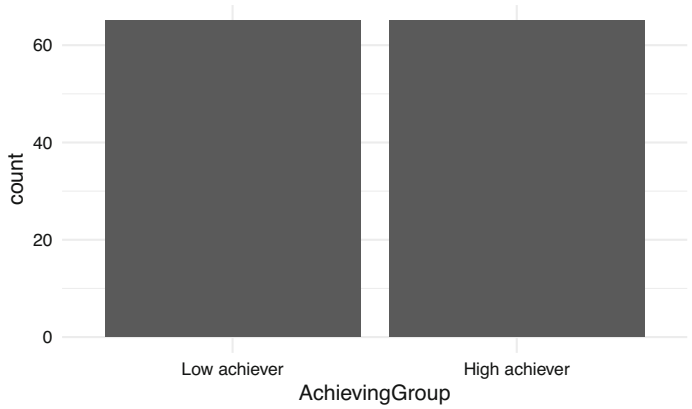


Fig. 20 Basic bar plot of students by achievement group

Remember that you can add a "third dimension" to the plot by using the `fill` property. This is known as a 'stacked' bar chart and helps highlight the proportion of, in this case, students' activity level (`ActivityGroup`) (Fig. 21).

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  scale_fill_viridis_d(direction = -1) +
  geom_bar(position = position_stack(reverse = TRUE))
```

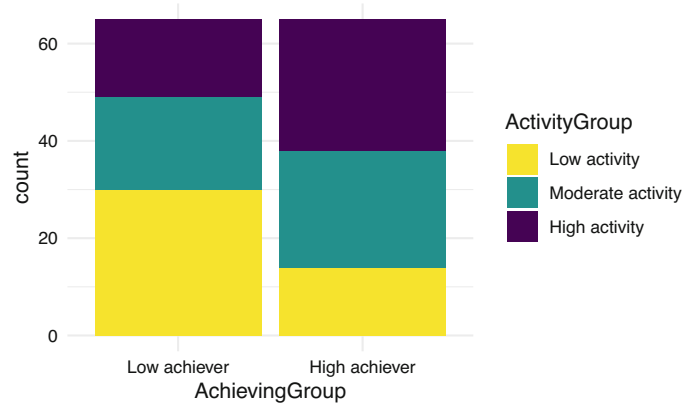


Fig. 21 Basic bar plot of students by achievement group filled by activity level

If we care more about the actual number rather than the proportion of students with each activity level, instead of a stacked bar chart we can keep each 'stack' as a whole bar of their own. This plot is very useful to compare values among categories. We accomplish this by passing the `position` argument with the value "dodge" to the `geom_bar` component (Fig. 22):

```
ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup)) +
  scale_fill_viridis_d(direction = -1) + geom_bar(position = "dodge")
```

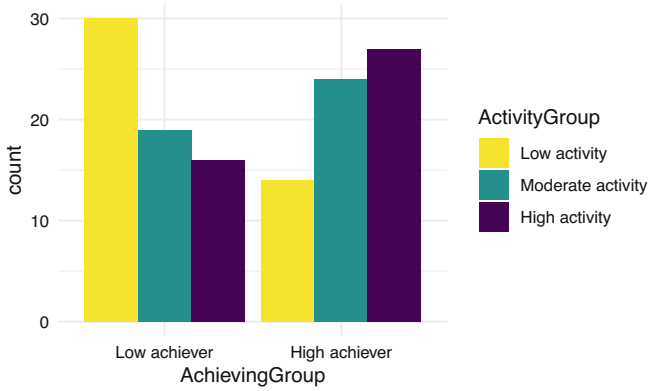


Fig. 22 Basic bar plot of students by achievement group filled by activity level with position dodge instead of stacked

We can now see that the highest group is represented by the low achievers with low activity, followed by the high achievers with high activity.

3.3.2 Histogram

Histograms allow us to represent the distribution of a single continuous variable. It is inherently a bar chart, but instead of each bar representing the count of a single category, it represents the count of a range of values in the x-axis (what is known as a bin). Let us, for example, create a histogram for students' online activity. Specifically, let us see the distribution of the number of accesses to the course main page online.

If we look at our dataset, we can see that the name of the variable that we are interested in is `Frequency.Course_view`:

```
head(df)
```

```
# A tibble: 130 x 37
   User      Name       Surname  Origin     Gender Birthdate Location Employment
   <chr>     <chr>      <chr>    <chr>      <chr>  <chr>     <chr>    <chr>
 1 00a05cc62 Wan        Tan      Malaysia   M      12.12.19~ Remote   None
 2 042b07ba1 Daniel     Tromp    Aruba      M      28.5.1999 Remote   None
 3 046c35846 Sarah      Schmit   Luxembourg F      25.4.1997 On camp~ None
 4 05b604102 Lian       Abdullah Yemen      F      19.11.19~ On camp~ None
 5 0604ff3d3 Nina       Borg     Malta      F      13.6.1994 On camp~ None
 6 077584d71 Mohamed    Gamal    Egypt      M      13.7.1998 On camp~ Part-time
 7 081b100cf Maximilian Gruber   Austria    M      20.12.19~ On camp~ None
 8 0857b3d8e Hugo       Pérez    Spain      M      22.12.19~ On camp~ None
 9 0af619e4b Aylin      Barat    Kazakhstan F      14.8.1995 On camp~ None
10 0ec99ce96 Polina     Novik    Belarus    F      9.10.1996 On camp~ None
# i 120 more rows
# i 29 more variables: Frequency.Applications <dbl>,
#   Frequency.Assignment <dbl>, Frequency.Course_view <dbl>,
#   Frequency.Feedback <dbl>, Frequency.General <dbl>,
#   Frequency.Group_work <dbl>, Frequency.Instructions <dbl>,
#   Frequency.La_types <dbl>, Frequency.Practicals <dbl>,
#   Frequency.Social <dbl>, Frequency.Ethics <dbl>, Frequency.Theory <dbl>, ...
```

To create a histogram for this variable we may use the `geom_histogram` feature of `ggplot2`. We just pass our dataset and map the `Frequency.Course_view` variable to the x axis, and we add the geometry `geom_histogram` (Fig. 23):

```
ggplot(df, mapping = aes(x = Frequency.Total)) + geom_histogram()
```

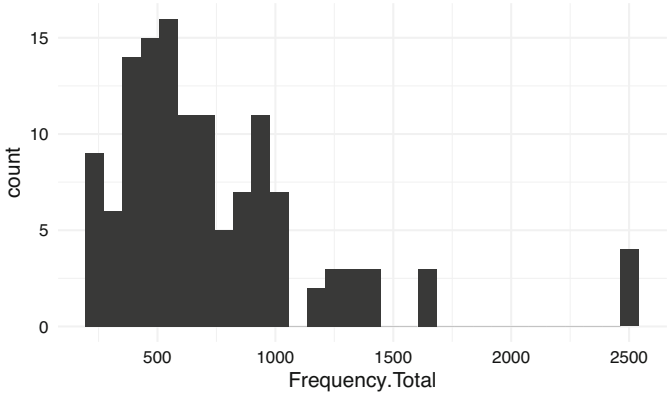


Fig. 23 Histogram of students' course page views

We can provide our own value to the `bins` argument in `geom_histogram` to personalize how many bins we want in our plot (Fig. 24):

```
ggplot(df, mapping = aes(x = Frequency.Total)) +
  geom_histogram(bins = 50)
```

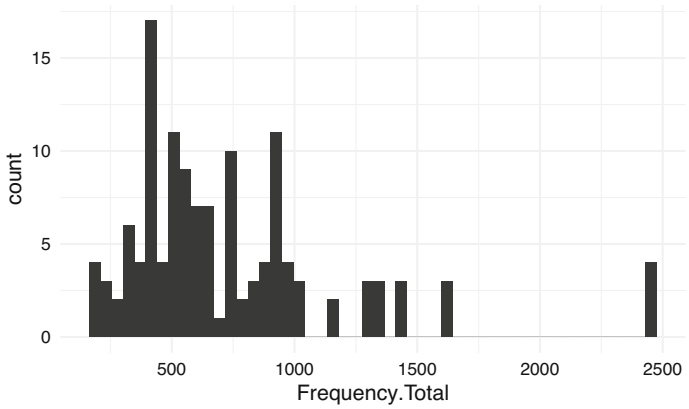


Fig. 24 Histogram of students' course page view with 50 bins

We can also personalize the color scheme using `fill` for the background of the bars (Fig. 25):

```
ggplot(df, mapping = aes(x = Frequency.Total)) +
  geom_histogram(bins = 20,  fill = "deeppink" ) +
  scale_x_continuous(n.breaks = 10)
```

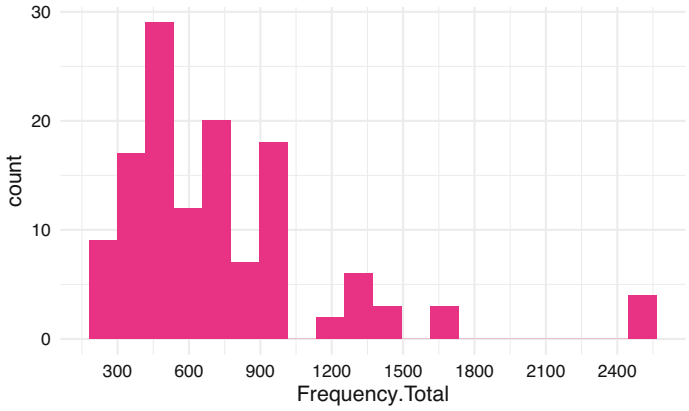


Fig. 25 Histogram of students' course page view with color, fill and linewidth

The histogram allows us to acknowledge that most students had around 400–500 events, with another peak around 900–1000. Students with more than 1000 events were rare.

3.3.3 Line Plot

Another very widely used type of plot is the line plot. Like the histogram, it is also appropriate when we have both a numerical continuous x-axis and y-axis but it gives us a bit more liberty of what we plot and it is suitable for when we want to plot several series of data together. A very common scenario for a line plot is when we deal with timelines and we wish to visualize the evolution of a certain variable over time. Let us, for instance, plot the students' daily events in the LMS throughout the course, a common plot in learning analytics dashboards. In the dataset that we have been using, we have the total count of events per user but not the timestamp of each event. We need to import the original event data from the dataset:

```
ev_url <- "https://github.com/lamethods/data/raw/main/1_moodleLAcourse/Events.xlsx"
events <- import(ev_url)
```

The `Events.xlsx` file contains all the actions that the students enrolled in this course performed in the LMS (`Action`) with their corresponding timestamp (`timecreated`): clicking on a lecture file, viewing the assignment instructions, etc.

```
head(events)
```

```
# A tibble: 95,626 x 7
   Event.context     user  timecreated         Component Event.name Log   Action
   <chr>             <chr> <dttm>              <chr>     <chr>      <chr> <chr>
 1 Assignment: Fina~ 9d74~ 2019-10-26 09:37:12 Assignme~ Course mo~ Assi~ Assig~
 2 Assignment: Fina~ 9148~ 2019-10-26 09:09:34 Assignme~ The statu~ Assi~ Assig~
 3 Assignment: Fina~ 278a~ 2019-10-18 12:05:28 Assignme~ Course mo~ Assi~ Assig~
 4 Assignment: Fina~ 53d6~ 2019-10-19 13:28:37 Assignme~ The statu~ Assi~ Assig~
 5 Assignment: Fina~ aab7~ 2019-10-15 23:38:13 Assignme~ Course mo~ Assi~ Assig~
 6 Assignment: Fina~ 82ed~ 2019-10-18 17:51:43 Assignme~ Course mo~ Assi~ Assig~
 7 Assignment: Fina~ 4178~ 2019-10-18 15:22:56 Assignme~ Course mo~ Assi~ Assig~
 8 Assignment: Fina~ 82ed~ 2019-10-22 13:46:51 Assignme~ The statu~ Assi~ Assig~
 9 Assignment: Fina~ f2e9~ 2019-10-15 14:58:17 Assignme~ Submissio~ Assi~ Assig~
10 Assignment: Fina~ 53d6~ 2019-10-19 13:28:38 Assignme~ Course mo~ Assi~ Assig~
# i 95,616 more rows
```

Instead of mapping `timecreated` directly to the `x` aesthetic, we can plot the timeline of the number of events per day by using `as.Date(timecreated)` and the `geom_line` geometry from `ggplot2`. Notice that, unlike `geom_bar`, if we do not provide a `y` aesthetic and want `ggplot2` to count the number of events per day for us, we need to make it explicit by passing the `stat` argument with value `"count"` to `geom_line` (Fig. 26).

```
ggplot(events, aes(x = as.Date(timecreated) )) + geom_line(stat = "count")
```

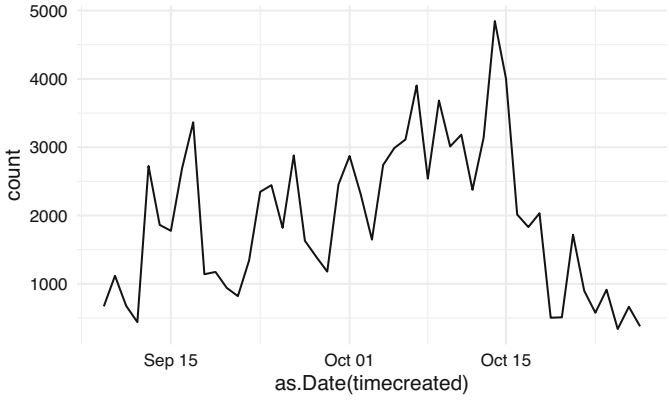


Fig. 26 Line plot of number of events per day

The line plot of students' events allows us to identify periods of increased activity. We can see that it was low at the very beginning of the course, with some peaks corresponding to the assignment deadlines and one last peak for the final project. When the course is over, activity begins to decrease.

To make our plot more aesthetically pleasing, we can customize the color and line width. We do so by tweaking the `color` and `linewidth` properties of the `geom_line`. We can also fix the axes' titles as we learned before (Fig. 27). For example:

```
ggplot(events, aes(x = as.Date(timecreated) )) +
  geom_line(stat = "count", color = "turquoise", linewidth = 2) +
  xlab ("Date") + ylab("Number of events")
```

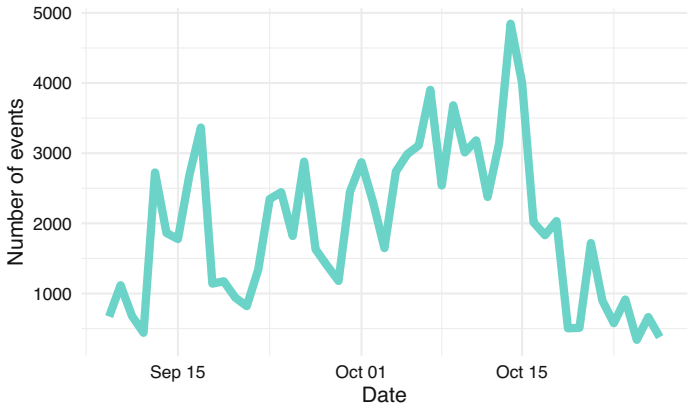


Fig. 27 Line plot of number of events per day with color, linewidth, and custom labels

We can also add a point to mark each date using `geom_point` (Fig. 28):

```
ggplot(events, aes(x = as.Date(timecreated) )) +
  geom_line(stat = "count", color = "turquoise", linewidth = 1.5)  +
  geom_point(stat = "count",  color = "purple", size = 2, stroke = 1)  +
  xlab ("Date")  +
  ylab("Number of events")
```

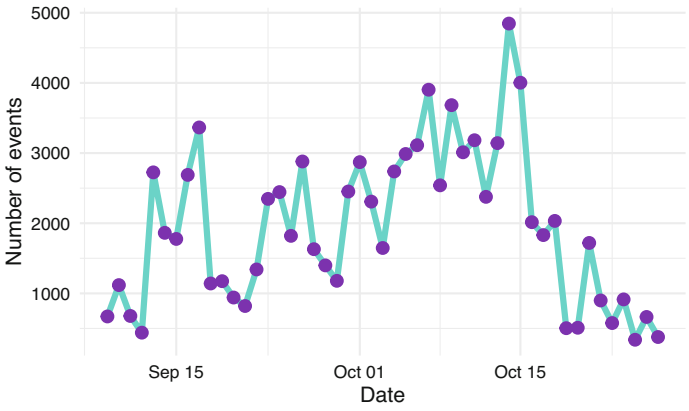


Fig. 28 Line plot of number of events per hour with points every hour

Besides visualizing the events for all the students of the course, we can pinpoint specific students to follow their progress and offer them personalized support. To do this, we would need to filter our data before handing it over to `ggplot2`. We can filter the data using the `filter` function from `dplyr`, as we learned in Chapter 4 [29]. We first install `dplyr` if we do not have it:

```
install.packages("dplyr")
```

Then, we import it as usual:

```
library(dplyr)
```

We can now filter the data and pass it on to `ggplot2` (Fig. 29):

```
events |> filter(user == "9d744e5bf") |> ggplot(aes(x = as.Date(timecreated) )) +
  geom_line(stat = "count", color = "turquoise", linewidth = 2)  +
  geom_point(stat = "count",  color = "purple", size = 2, stroke = 1)  +
  xlab ("Date")  +
  ylab("Number of events")
```

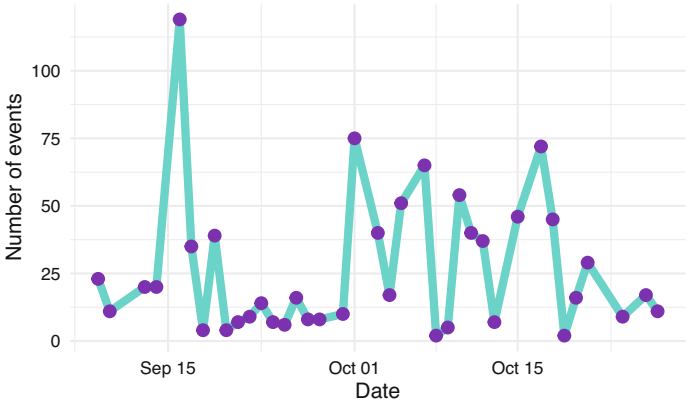


Fig. 29 Line plot of number of events per date for a single student

3.3.4 Jitter Plots

In the previous plots we have seen aggregated information for all the cohort of students as well as information for a single student. However, in some occasions, it is very useful to see the general picture while accounting for possible individual differences. For example, using our original `df` dataset, we can plot the number of events on the LMS, differentiating between high achievers and low achievers.

One option is to use `geom_point` to represent each students' count of events as a single point. To do this, we map the `Event` column to the `x` aesthetic, the `Frequency` column to the `y` aesthetic, and the `User` column to the `group` aesthetic (Fig. 30):

```
ggplot(df, aes(x = AchievingGroup, y = Frequency.Total)) +
  geom_point() +
  xlab("Achieving group") +
  ylab("Number of events") +
  theme(legend.position = "bottom",
        legend.text = element_text(size = 7),
        legend.title = element_blank())
```

However, there are many points that overlap. If we use `geom_jitter` instead, we take advantage of the horizontal gap between the event names to spread the points and avoid the overlap:

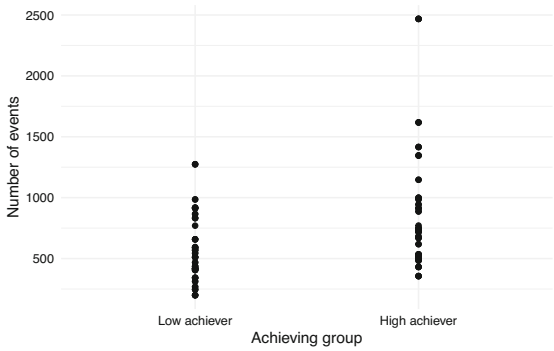


Fig. 30 Jitter plot of number of events per achievement group using `geom_point`

```
ggplot(df, aes(x = AchievingGroup, y = Frequency.Total)) +
  geom_jitter() +
  xlab("Achieving group") +
  ylab("Number of events") +
  theme(legend.position = "bottom",
        legend.text = element_text(size = 7),
        legend.title = element_blank())
```

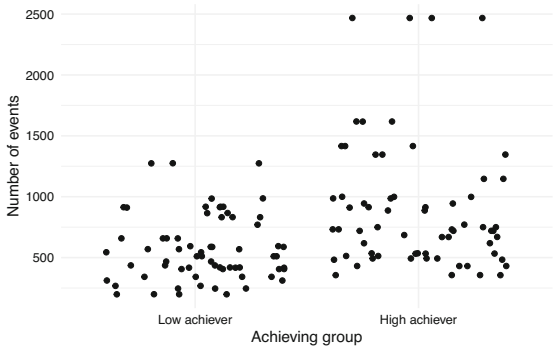


Fig. 31 Jitter plot of number of events per achievement group using `geom_jitter`

The plot shows that students that are high achievers generally have a higher number of events than low achievers (Fig. 31).

3.3.5 Box Plot

When we have too many data points, it is often more useful to visualize summary statistics instead of all the points. Box plots are very useful in summarizing data distributions. We can create a box plot for the number of events per achievement group using `geom_boxplot`:

```
ggplot(df, aes(x = AchievingGroup, y = Frequency.Total)) + geom_boxplot() +
  xlab("Achieving group") + ylab("Number of events")
```

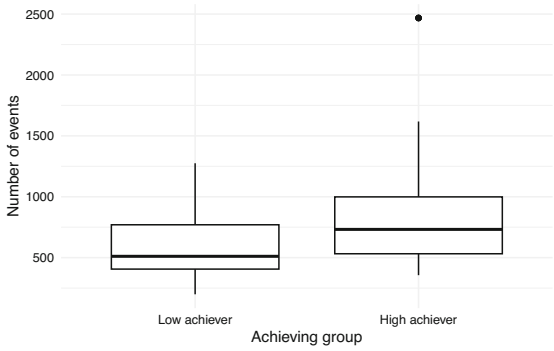


Fig. 32 Box plot of activity per achievement group

The lower hinge of each box indicates the 25% percentile, the thick middle line is the median, and the top hinge is the top 75% percentile. The upper whisker extends from the hinge up to the maximum value within 1.5 * IQR (inter-quantile range), whereas the lower whisker extends to the minimum value within 1.5 * IQR of the hinge. The points outside the whisker represent outliers in the distribution (i.e., values outside of the 1.5 * IQR range). As the jitter plot already hinted, the median number of events is higher in the high achieving group (Fig. 32).

3.3.6 Violin Plot

We can also visualize the distribution of the number of events for each group using violin plots (`geom_violin`), but these are recommended when we have a large amount of data (Fig. 33):

```
ggplot(df, aes(x = AchievingGroup, y = Frequency.Total)) + geom_violin() +
  xlab("Achieving group") + ylab("Number of events")
```

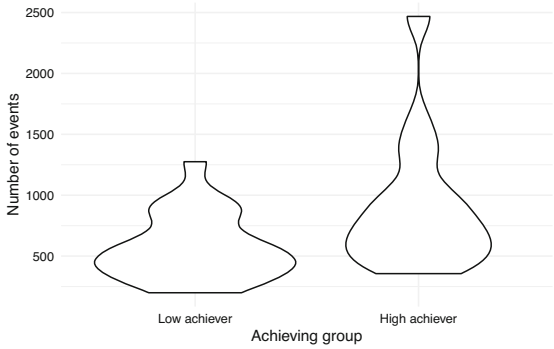


Fig. 33 Violin plot of total activity per achievement group

3.3.7 Scatter Plots

The examples we have seen so far have dealt with plotting a single variable alone or divided in categories. Another common scenario is to investigate the direct relationship between two or more variables. Scatter plots are used to visualize how two numerical variables relate to each other. For example, we can use them to see how LMS activity relates to grades (Fig. 34).

```
ggplot(df, aes(x = Frequency.Total, y = Final_grade)) +
  geom_point() +
  ylab("Final grade") + xlab("Number of events")
```

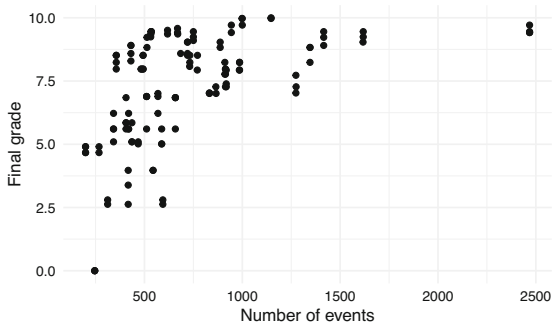


Fig. 34 Scatter plot of number of events vs. final grade

In the plot, each point represents a student. Students at the right side of the plot represent students with higher activity, while students closer to the left side of the plot, represent students with lower activity. At the same time, students with low grades are closer to the bottom of the plot, while students with high grades are closer to the top. Overall, se see an upward trend whereby students with higher activity indeed obtain better grades.

We can add another dimension by coloring points according to another variable. For example, we can color the points according to high vs. low achievers (Fig. 35), so we can now where the division between the two groups is:

```
ggplot(df, aes(x = Frequency.Total, y = Final_grade, color = AchievingGroup)) +
  geom_point() +
  ylab("Final grade") + xlab("Number of events") +
  labs(color = "Achievement")
```

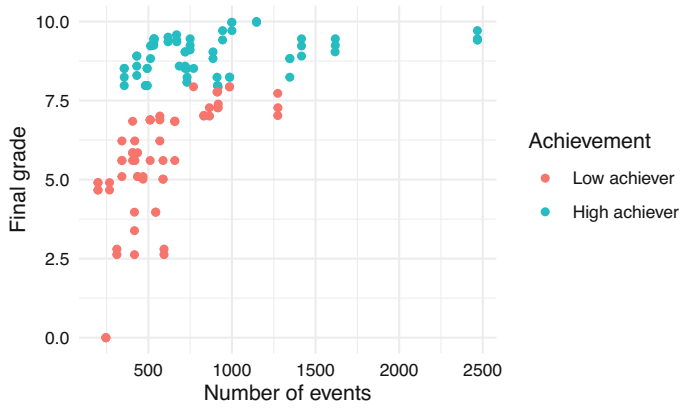


Fig. 35 Scatter plot of number of events vs. final grade colored by achievement group

We can add yet another dimension by mapping the `size` aesthetic to another variable, for example `Frequency.Group_work` which represents the number of events related to group work (Fig. 36).

```
ggplot(df, aes(x = Frequency.Total, y = Final_grade,
               fill = AchievingGroup, size = Frequency.Group_work)) +
  geom_point(color = "black", pch = 21) +
  scale_size_continuous(range = c(1, 7)) +
  ylab("Final grade") + xlab("Number of events") +
  labs(size = "Group work", fill = "Achievement")
```

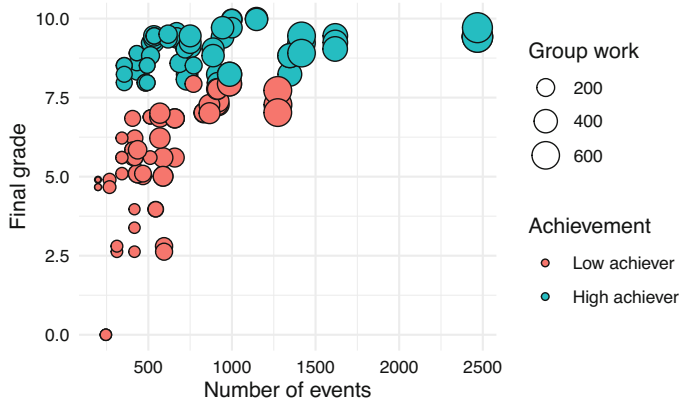


Fig. 36 Scatter plot of number of events vs. final grade colored by achievement group and sized by frequency of group work

3.4 Advanced Features

3.4.1 Plot Grids

Sometimes, adding all the information in a single plot can be overwhelming and hard to interpret. For example, take a look at the following line plot that shows the number of events per day for each of the course online components (Fig. 37):

```
ggplot(events, aes(x = as.Date(timecreated), color = Action )) +
  scale_fill_viridis_d() +
  geom_line(stat = "count") +
  xlab("Date") +
  ylab("Number of events")
```

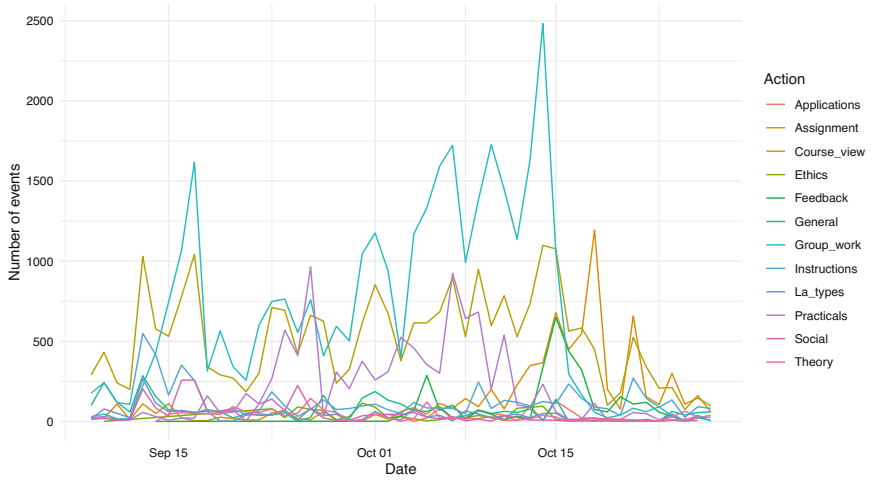


Fig. 37 Multiple series line plot

If we had only a few (2–5) lines, the plot would probably look good, but as the number of categories grow, the plot becomes unintelligible. Instead of showing all the lines together, the plot would be easier to understand if each component had their own plot. To do this, instead of mapping the `Action` column to the `color` aesthetic, we add a new component to our plot using `facet_wrap` and we pass the name of the column as a character string (`"Action"`). We can change the `geom_line` to a `geom_area` to enhance the visualization (Fig. 38).

```
ggplot(events, aes(x = as.Date(timecreated))) +
  geom_area(stat = "count", fill = "turquoise", color = "black") +
  facet_wrap("Action") +
  xlab("Date") +
  ylab("Number of events")
```

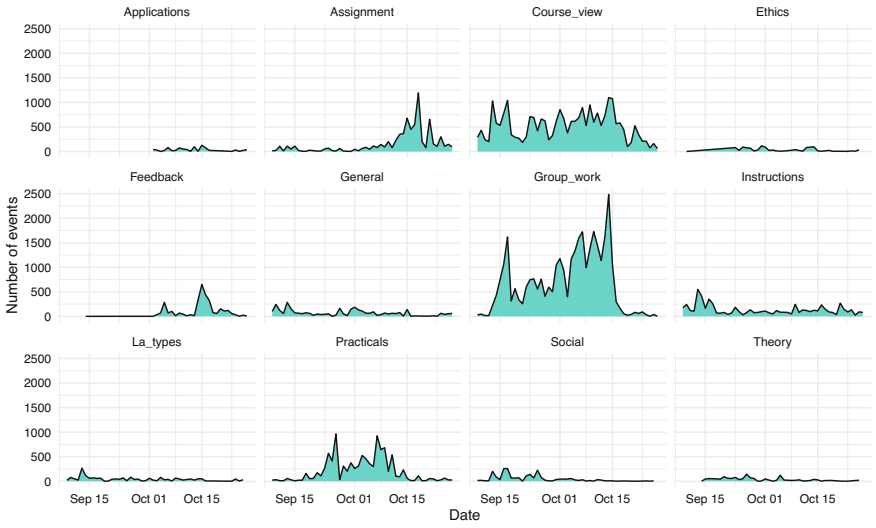


Fig. 38 Grid of multiple plots

3.4.2 Combining Multiple Plots

In the previous example, we saw how to split a plot into multiple plots. But what happens if we want to combine multiple independent plots? For that purpose, we can use the library `patchwork`. Install it first if you do not have it already:

```
install.packages("patchwork")
```

We import the `patchwork` library:

```
library(patchwork)
```

We have to create the plots that we want to combine and assign each of them to a different variable. We can use previous examples from this chapter and assign them to variables named p1, p2, and p3.

```
p1 <- ggplot(df, aes(x = Frequency.Total, y = Final_grade)) +
  geom_point() + ylab("Grade") +
  xlab("Total number of events")

p2 <- ggplot(df, aes(x = AchievingGroup, fill = ActivityGroup )) +
  geom_bar(position = position_fill(reverse = T)) +
  scale_fill_viridis_d(direction = -1) +
  xlab("Achievement group") +
  ylab("Number of events") +
  labs(fill = "Activity level")

p3 <- ggplot(events, aes(x = as.Date(timecreated) )) +
  geom_line(stat = "count", color = "turquoise", linewidth = 1.5)  +
  geom_point(stat = "count",  color = "purple", size = 2, stroke = 1)  +
  xlab ("Date")  +
  ylab("Number of events")
```

Now, if we add the three variables together separated by the + sign, the plots will be placed horizontally next to each other (Fig. 39):

```
p1 + p2 + p3
```

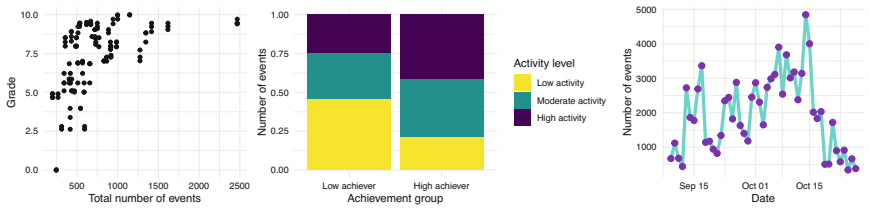


Fig. 39 Multiple plots stacked horizontally

If we use the / character side instead, we lay them out vertically (Fig. 40):

```
p1 / p2 / p3
```

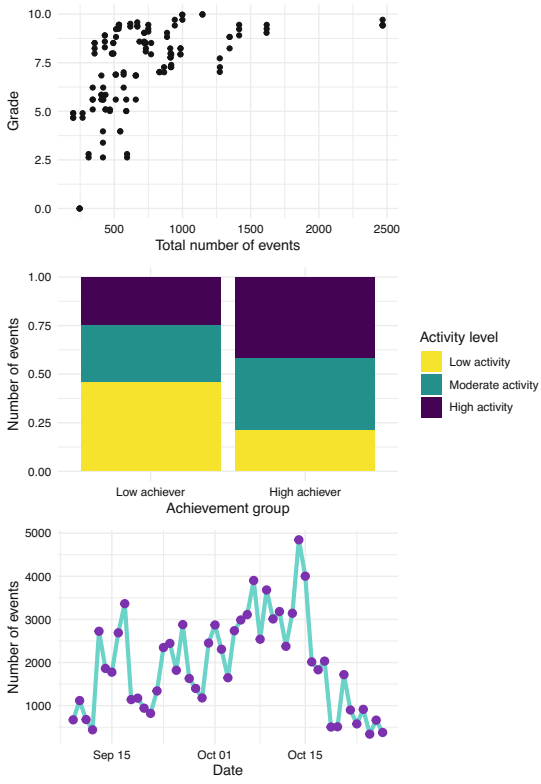


Fig. 40 Multiple plots stacked vertically

We can use combinations of both signs and even leave blank spaces as follows (Fig. 41):

```
(p1 + p2) / ( p3 + plot_spacer())
```

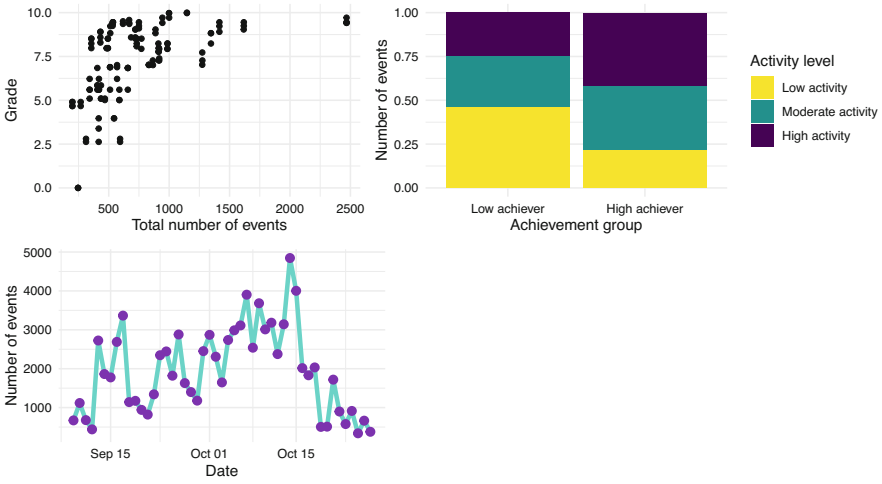


Fig. 41 Multiple plots in a grid

Putting plots side by side can be very useful to compare datasets and discuss the differences. Some publication venues limit the number of figures or pages of their articles, so combining several plots together can be very useful to overcome this limitation.

4 Creating Tables with `gt`

We have seen earlier in this chapter multiple types of visualizations that are suitable for diverse scenarios in learning analytics. However, we must not forget the other main way of reporting results or metrics, i.e., tables. When we display a data frame in Rstudio, it is by default presented as a table, but we need to be able to extract this table and display it in a dashboard, a report or a scientific article. The library `gt` can help us with this endeavor. First, install it if you do not have it yet:

```
install.packages("gt")
```

We then import it, as usual:

```
library(gt)
```

Let us create a table, for example, to display the descriptive statistics of students' events in the LMS. Using the `events` dataset, we first count the number of events of each type (`Event.name`) per student (`user`) using `group_by` and `count` from

`dplyr`. We then group by `Event.name` only and use the `summarize` function, also from `dplyr`, to create the mean, and standard deviation of the number of events of each type per student, as we learned in Chapter 5 [30].

```
events |>
  group_by(user, Action) |>
  count() |>
  group_by(Action) |>
  summarize(Mean = mean(n), SD = sd(n))
```

```
# A tibble: 12 x 3
   Action        Mean     SD
   <chr>        <dbl>  <dbl>
 1 Applications  11.1   9.83
 2 Assignment    56.7  34.1
 3 Course_view  195.  152.
 4 Ethics        11.7  10.7
 5 Feedback      24.7  16.2
 6 General       25.7  21.4
 7 Group_work   252.  163.
 8 Instructions  49.8  40.3
 9 La_types      14.5   7.58
10 Practicals    77.1  33.8
11 Social        18.1  19.0
12 Theory        11.1   6.92
```

Now that we have a data frame with the shape that we like, we can use `gt` to create the formatted table by simply adding `gt` to the pipeline of operations (Table 1):

```
events |>
  group_by(user, Action) |>
  count() |>
  group_by(Action) |>
  summarize(Mean = mean(n), SD = sd(n)) |>
  gt()
```

Table 1 Table created with gt

Action	Mean	SD
Applications	11.07143	9.825022
Assignment	56.68462	34.129492
Course_view	194.56154	151.656947
Ethics	11.68182	10.669050
Feedback	24.71429	16.243082
General	25.73846	21.390991
Group_work	251.90769	162.899810
Instructions	49.80000	40.272213
La_types	14.54615	7.583245
Practicals	77.07692	33.751627
Social	18.10744	19.034093
Theory	11.10484	6.922120

We might add some tweaks by forcing the numerical columns to have two decimals and the first column to be aligned left. You can also apply themes to the table using the library `gtExtras` (Table 2).

```
events |>
  group_by(user, Action) |>
  count() |>
  group_by(Action) |>
  summarize(Mean = mean(n), SD = sd(n)) |>
  gt() |>
  fmt_number(decimals = 2, columns = where(is.numeric)) |>
  cols_align(align = "left", columns = 1)
```

Table 2 Table created with gt with formatting

Action	Mean	SD
Applications	11.07	9.83
Assignment	56.68	34.13
Course_view	194.56	151.66
Ethics	11.68	10.67
Feedback	24.71	16.24
General	25.74	21.39
Group_work	251.91	162.90
Instructions	49.80	40.27
La_types	14.55	7.58
Practicals	77.08	33.75
Social	18.11	19.03
Theory	11.10	6.92

5 Discussion

The use of data visualization in the context of learning analytics has the potential to greatly enhance our understanding of student behavior and performance. Using tools such as `ggplot2`, instructors and researchers can create informative and visually appealing plots that highlight important patterns and trends in student activity, providing insights into factors that may be impacting student success and therefore inform instructional decisions and improve student outcomes.

As we have already seen throughout the chapter, we often use different plots when dealing with categorical variables or numerical variables; when plotting a single variable or two (or more), etc. Moreover, on some occasions when we need very detailed information, a table might be more informative compared to a figure. As a summary for the possible visualizations, Table 3 gathers the most commonly used visualization types that we have seen throughout this chapter according to the number of variables and the data type. It also points to the `ggplot2` geometry that is used to create each visualization.

Table 3 Summary of the types of visualization for each data type and number of variables

Number of variables	Variable types	Type of visualization	`ggplot2` geometry
One variable	Continuous	Histogram	`geom_hist()`
	Discrete	Bar chart	`geom_bar()`
Two or more variables	Both continuous	Scatter plot	`geom_point()`
	One discrete time and one continuous	Line chart	`geom_line()`
		Area chart	`geom_area()`
	One discrete and one continuous	Bar chart	`geom_bar()`
		Box plot	`geom_boxplot()`
		Jitter plot	`geom_jitter()`
		Violin plot	`geom_violin()`
	Both discrete	Stacked bar chart	`geom_bar()`

Another way to decide which visualization to use is to think what kind of story we want to tell or which aspect of our data we want to highlight. Figure 42 shows a flowchart that can help choose the most suitable visualization for our data. There are many other decision charts online made for this purpose. For example, "From Data to Viz"[1] leads you to the most appropriate graph for your data and also links to the code to build it and lists common caveats you should avoid.

Throughout the rest of the book, we will see other forms of data visualization that are inherent to specific learning analytics methods. For example, in Chapter 15 [31], we will learn how to represent students' discussions in the form of

[1] Data to Viz https://www.data-to-viz.com/.

Fig. 42 Flowchart to decide the most appropriate visualization for your data

social networks, and in Chapter 10 [32], we will represent students' sequences of activities using sequence analysis. The foundations learned in this chapter are key to understanding more complex visualizations in learning analytics and are, of course, transferable to other fields as well. We encourage readers to expand their knowledge of data visualization by referring to the recommended resources in the next section. Especially readers that would like to take their visualizations to the next step should consider using `shiny`,[2] a web framework for R that allows creating fully interactive web apps for data analyses such as dashboards.

6 Additional Material

- Wilke, Claus. 2019. *Fundamentals of Data Visualization*. O'Reilly. https://clauswilke.com/dataviz/.
- Rahlf, Thomas. 2019 *Data visualisation with R: 111 Examples.* Springer. https://doi.org/10.1007/978-3-030-28444-2.
- Wickham, Hadley, Danielle Navarro, and Thomas Lin Pedersen. 2019. *ggplot2: Elegant Graphics for Data Analysis (Use R)* https://ggplot2-book.org/index.html.
- Sahin, Muhittin and Dirk Ifenthaler. 2021. *Visualizations and Dashboards for Learning Analytics*. Springer. https://doi.org/10.1007/978-3-030-81222-5.

[2] Shiny https://mastering-shiny.org/.

- Dougherty, Jack and Ilya Ilyankou. 2021. *Hands-On Data Visualization: Interactive Storytelling from Spreadsheets to Code* https://handsondataviz.org/spreadsheet.html.
- *From Data to Viz*. https://www.data-to-viz.com/about.html
- Wickham, Hadley. 2021. *Mastering shiny*. O'Reilly. https://mastering-shiny.org/.

References

1. Kirk A (2012) Data visualization: a successful design process. Packt Publishing, Birmingham
2. Demmans Epp C, Bull S (2015) Uncertainty representation in visualizations of learning analytics for learners: current approaches and opportunities. IEEE Trans Learn Technol 8:242–260. https://doi.org/10.1109/tlt.2015.2411604
3. Jivet I, Scheffel M, Drachsler H, Specht M (2017) Awareness is not enough: pitfalls of learning analytics dashboards in the educational practice. Springer, Berlin, pp 82–96
4. Park Y, Jo I-H (2019) Factors that affect the success of learning analytics dashboards. Edu Technol Res Develop 67:1547–1571. https://doi.org/10.1007/s11423-019-09693-0
5. Jivet I, Wong J, Scheffel M, Valle Torre M, Specht M, Drachsler H (2021) Quantum of choice: how learners' feedback monitoring decisions, goals and self-regulated learning skills are related. In: LAK21: 11th international learning analytics and knowledge conference. https://doi.org/10.1145/3448139.3448179
6. Sedrakyan G, Malmberg J, Verbert K, Järvelä S, Kirschner PA (2020) Linking learning behavior analytics and learning science concepts: designing a learning analytics dashboard for feedback to support learning regulation. Comput Human Behav 107:105512. https://doi.org/10.1016/j.chb.2018.05.004
7. Martinez-Maldonado R, Echeverria V, Fernandez Nieto G, Buckingham Shum S (2020) From data to insights: a layered storytelling approach for multimodal learning analytics. In: Proceedings of the 2020 CHI conference on human factors in computing systems. https://doi.org/10.1145/3313831.3376148
8. de Freitas S, Gibson D, Alvarez V, Irving L, Star K, Charleer S, Verbert K (2017) How to use gamified dashboards and learning analytics for providing immediate student feedback and performance tracking in higher education. In: Proceedings of the 26th international conference on world wide web companion - WWW'17 companion. https://doi.org/10.1145/3041021.3054175
9. Bodily R, Kay J, Aleven V, Jivet I, Davis D, Xhakaj F, Verbert K (2018) Open learner models and learning analytics dashboards. In: Proceedings of the 8th international conference on learning analytics and knowledge. https://doi.org/10.1145/3170358.3170409
10. Susnjak T, Ramaswami GS, Mathrani A (2022) Learning analytics dashboard: a tool for providing actionable insights to learners. Int J Edu Technol Higher Edu 19:12. https://doi.org/10.1186/s41239-021-00313-7
11. Bodily R, Verbert K (2017) Review of research on student-facing learning analytics dashboards and educational recommender systems. IEEE Trans Learn Technol 10:405–418. https://doi.org/10.1109/tlt.2017.2740172
12. Valle N, Antonenko P, Dawson K, Huggins-Manley AC (2021) Staying on target: a systematic literature review on learner-facing learning analytics dashboards. British J Edu Technol https://doi.org/10.1111/bjet.13089
13. Perez-Alvarez R, Jivet I, Perez-Sanagustin M, Scheffel M, Verbert K (2022) Tools designed to support self-regulated learning in online learning environments: a systematic review. IEEE Trans Learn Technol 15:508–522. https://doi.org/10.1109/tlt.2022.3193271

14. Matcha W, Uzir NA, Gasevic D, Pardo A (2020) A systematic review of empirical studies on learning analytics dashboards: a self-regulated learning perspective. IEEE Trans Learn Technol 13:226–245. https://doi.org/10.1109/tlt.2019.2916802
15. Cheng J, Lei J (2020) A description of students' commenting behaviours in an online blogging activity. E-Learn Digit Media 18:209–225. https://doi.org/10.1177/2042753020954971
16. Duan X, Wang C, Rouamba G (2022) Designing a learning analytics dashboard to provide students with actionable feedback and evaluating its impacts. In: Proceedings of the 14th international conference on computer supported education. https://doi.org/10.5220/0011116400003182
17. van Leeuwen A, Rummel N (2020) Comparing teachers' use of mirroring and advising dashboards. In: Proceedings of the tenth international conference on learning analytics & knowledge. https://doi.org/10.1145/3375462.3375471
18. Isaias P, Backx Noronha Viana A (2020) On the design of a teachers' dashboard: requirements and insights. Springer, Berlin, pp 255–269
19. Verbert K, Govaerts S, Duval E, Santos JL, Van Assche F, Parra G, Klerkx J (2013) Learning dashboards: an overview and future research opportunities. Person Ubiq Comput 18:1499–1514. https://doi.org/10.1007/s00779-013-0751-2
20. Chavan P, Mitra R (2022) Tcherly. J Learn Anal 9:125–151. https://doi.org/10.18608/jla.2022.7555
21. López-Pernas S, Gordillo A, Barra E, Quemada J (2021) Escapp: a web platform for conducting educational escape rooms. IEEE Access 9:38062–38077. https://doi.org/10.1109/access.2021.3063711
22. López Tavares D, Perkins K, Kauzmann M, Aguirre Velez C (2019) Towards a teacher dashboard design for interactive simulations. J Phys Conf Ser 1287:012055. https://doi.org/10.1088/1742-6596/1287/1/012055
23. Li Y, Zhang M, Su Y, Bao H, Xing S (2022) Examining teachers' behavior patterns in and perceptions of using teacher dashboards for facilitating guidance in CSCL. Edu Technol Res Develop 70:1035–1058. https://doi.org/10.1007/s11423-022-10102-2
24. Jivet I, Scheffel M, Specht M, Drachsler H (2018) License to evaluate. In: Proceedings of the 8th international conference on learning analytics and knowledge. https://doi.org/10.1145/3170358.3170421
25. Martinez-Maldonado R, Pardo A, Mirriahi N, Yacef K, Kay J, Clayphan A (2015) The LATUX workflow. In: Proceedings of the fifth international conference on learning analytics and knowledge. https://doi.org/10.1145/2723576.2723583
26. Wickham H (2016) ggplot2: elegant graphics for data analysis. Springer, New York
27. Wilkinson L (1999) The grammar of graphics. Springer, New York
28. López-Pernas S, Saqr M, Conde J, Del-Río-Carazo L (2024) A broad collection of datasets for educational research training and application. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, Berlin
29. Kopra J, Tikka S, Heinäniemi M, López-Pernas S, Saqr M (2024) Data cleaning and wrangling. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, Berlin
30. Tikka S, Kopra J, Heinäniemi M, López-Pernas S, Saqr M (2024) Introductory statistics with R for educational researchers. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, Berlin
31. Saqr M, López-Pernas S, Conde MÁ, Hernández-García Á (2024) Social network analysis: a primer, a guide and a tutorial in R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, Berlin
32. Saqr M, López-Pernas S, Helske S, Durand M, Murphy K, Studer M, Ritschard G (2024) Sequence analysis in education: principles, technique, and tutorial with R. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: a practical guide using R. Springer, Berlin

Part II
Machine Learning

Predictive Modelling in Learning Analytics: A Machine Learning Approach in R

Jelena Jovanovic, Sonsoles López-Pernas, and Mohammed Saqr

1 Introduction

Prediction of students' performance has been a central theme within the field of learning analytics (LA) since the early days [1]. In fact, the initial conceptualization of the field has highlighted the use of digital data collected from learners to predict their success—among other usages. Such predictions hold the promise to help identify those who are at risk of low achievement, in order to proactively offer early support and appropriate intervention strategies based on insights derived from learners' data [1, 2]. Nevertheless, the prediction of students' performance is not unique to LA and was an important theme in related fields even before LA, e.g., academic analytics [3], educational data mining [4], and even far earlier in education research at large [5].

Such widespread, longstanding and continuous centrality of early and accurate prediction of students' performance lends itself to the premise that detection of early signs could allow a timely prevention of, e.g., dropout, low achievement, or undesired outcomes in general [6]. More importantly, identifying the predictors could help inform interventions, explain variations in outcomes and inform educators of why such outcomes happened—a form of predictive modelling that is often referred to as explanatory modelling [7]. Noticeable is the famous example of the *Signal* system at the University of Purdue, where predictions were based on digital data collected from an online learning platform [8]. *Signal* produced predictions and classified students into three categories according to "safety" and presented the students with traffic-light-inspired dashboard signs, where at-risk

J. Jovanovic (✉)
University of Belgrade, Belgrade, Serbia

S. López-Pernas · M. Saqr
School of Computing, University of Eastern Finland, Joensuu, Finland

M. Saqr, S. López-Pernas (eds.), *Learning Analytics Methods and Tutorials*,
https://doi.org/10.1007/978-3-031-54464-4_7

students get a red light. However, the influence of *Signal* on retention rates is unclear and often debated [9]. Several other systems were designed, built and applied in practice, e.g., *OU analyse* at the Open University, where the system offers informative dashboards to students and teachers as well as predictive models to forecast students' performance [10].

Successful prediction of students' performance has been demonstrated repeatedly in several LA studies across the years [11]. In general, the majority of the published studies used features extracted from logged learning trace data (i.e., data about students' interactions with online learning activities and resources) and achieved accurate predictions for a considerable number of students. Yet, most of such studies examined a single course or what is often referred to as a convenience sample (i.e., a course with a sufficiently large and accessible dataset) [11]. Studies that attempted to apply predictive modelling across several courses have not found similar success [12–15]. For instance, Finnegan et al. [15] examined 22 courses across three academic domains using student log-trace data recorded from the learning management system. The authors found considerable differences among predictive models developed for individual courses regarding their predictive power as well as the significance of features. Similar results were reported by Gašević et al. [12] who used data from nine undergraduate courses in different disciplines to examine how instructional variations affected the prediction of academic success. Gašević et al. [12] found that predictors were remarkably different across courses with no consistent pattern that would allow for having one model applicable across all courses. Similarly, Conijn et al. [13] examined 17 courses across several subjects and confirmed the considerable variability of the indicators and predictive models across courses.

Studies within the same domain have also found significant differences in predictors and predictive models. For instance, a recent study [14] examined 50 courses with a similar course design and homogeneous pedagogical underpinning. The authors found variations among different offerings of the same course, that is, the same predictor was statistically significantly correlated with performance in one course offering, but not in the same course offered to similar students in the next year. Furthermore, some predictors were more consistent than others e.g., the frequency of online sessions was more consistent than the frequency of lectures. In a similar vein, Jovanović et al. [16] applied mixed-effect linear modelling to data from fifty combined courses and developed several predictive models with different combinations of features. All predictive models in the work by Jovanović et al. [16] were able to explain only a limited proportion of variations in students' grades. The intraclass correlation coefficient (a measure of source of variability) of all models revealed that the main source of variability were students themselves, that is, students' specific features not captured in the logged data, pointing to the importance of taking students' international conditions into account.

The goal of this chapter is to introduce the reader to predictive LA. The next section is a review of the existing literature, including the main objectives, indicators and algorithms that have been operationalized in previous works. The remainder of the chapter is a step-by-step tutorial of how to perform predictive LA using R. The

tutorial describes how to predict student success using students' online trace log data extracted from a learning management system. The reader is guided through all the required steps to perform prediction, including the data preparation and exploration, the selection of the relevant indicators (i.e., feature engineering) and the actual prediction of student success.

2 Predictive Modelling: Objectives, Features, and Algorithms

Extensive research in the LA field has been devoted to the prediction of different measures of student success, as proven by the existence of multiple reviews and meta-analyses on the topic [17–20]. Among the measures of student success that have been examined in the literature are student retention [21], grades [22], and course completion [23]. Predicting lack of success has also been a common target of predictive analytics, mostly in the form of dropout [24], with special interest in the early prediction of at-risk students [25, 26].

To predict student success, numerous indicators from varying data sources have been examined in the literature. Initially, indicators were derived from students' demographic data and/or academic records. Some examples of such indicators are age, gender, and previous grades [27]. More recent research has focused on indicators derived from students' online activity in the learning management system (LMS) [17, 20]. Many of such indicators are derived directly from the raw log data such as the number of total clicks, number of online sessions, number of clicks on the learning materials, number of views of the course main page, number of assignments completed, number of videos watched, number of forum posts [13, 14, 28–31]. Other indicators are related to time devoted to learning, rather than to the mere count of clicks, such as login time, login frequency, active days, time-on-task, average time per online session, late submissions, and periods of inactivity [13, 14, 32–35]. More complex indicators are often derived from the time, frequency, and order of online activities, such as regularity of online activities, e.g., regularity of accessing lecture materials [16, 36, 37], or regularity of active days [14, 16]. Network centrality measures derived from network analysis of interactions in collaborative learning settings were also considered, as they compute how interactions relate to each other and their importance [38]. Research has found that predictive models with generic indicators are only able to explain just a small portion of the overall variability in students' performance [36]. Moreover, it is important to take into account learning design as well as quality and not quantity of learning [17, 20].

The variety of predictive algorithms that have been operationalized in LA research is also worth discussing. Basic algorithms, such as linear and logistic regression, or decision trees, have been used for their explainability, which allows teachers to make informed decisions and interventions related to the students

"at risk" [37]. Other machine learning algorithms have also been operationalized such as kNN or random forest [39, 40], although their interpretability is less straightforward. Lastly, the most cutting-edge techniques in the field of machine learning have also made their way to LA, such as XGBoost [41] or Neural Networks [42]. Despite the fact that the accuracy achieved by these complex algorithms is often high, their lack of interpretability is often pointed out as a reason for teachers to avoid making decisions based on their outcomes [7, 43].

It is beyond the scope of this review to offer a comprehensive coverage of the literature. Interested readers are encouraged to read the cited literature and the literature reviews on the topics [11, 14, 17–20]

3 Predicting Students' Course Success Early in the Course

3.1 Prediction Objectives and Methods

The overall objective of this section is to illustrate predictive modelling in LA through a typical LA task of making early-in-the-course predictions of the students' course outcomes based on the logged learning-related data (e.g., making predictions of the learners' course outcomes after log data has been gathered for the first 2–3 weeks). The course outcomes will be examined and predicted in two distinct ways: (1) as success categories (high vs. low achievement), meaning that the prediction task is approached with classification models; (2) as success score (final grades), in which case the development of regression models is required.

To meet the stated objectives, the following overall approach will be applied: create several predictive models, each one with progressively more learning trace data (i.e., logged data about the learners' interactions with course resources and activities), as they become available during the course. In particular, the first model will be built using the learning traces available at the end of the first week of the course; the second model will be built using the data available after the completion of the second week of the course (i.e., the data logged over the first 2 weeks); then, the next one will be built by further accumulating the data, so that we have learning traces for the first 3 weeks, and so on. In all these models, the outcome variable will be the final course outcome (high/low achievement for classification models, that is, the final grade for regression models). We will evaluate all the models on a small set of properly chosen evaluation metrics and examine when (that is, how early in the course) we can make reasonably good predictions of the course outcome. In addition, we will examine which learning-related indicators (i.e., features of the predictive models) had the highest predictive power.

3.2 Context

The context of the predictive modelling presented in this chapter is a postgraduate course on learning analytics (LA), taught at University of Eastern Finland. The course was 6 weeks long, though some assignments were due in the week after the official end of the course. The course covered several LA themes (e.g., Introductory topics, Learning theories, Applications, Ethics), and each theme was covered roughly in 1 week of the course. Each theme had a set of associated learning materials, mostly slides, and reading resources. The course reading resources included seminal articles, book chapters, and training materials for practical work. The course also contained collaborative project work (referred to as group projects). In the group project, students worked together in small groups to design an LA system. The group project was continuous all over the course and was designed to align with the course themes. For instance, when students learned about LA data collection, they were required to discuss the data collection of their own project. The group project has two grades, one for the group project as a whole and another for the individual contribution to the project. It is important to note here that the dataset is based on a synthetic anonymized version of the original dataset and was augmented to three times the size of the original dataset. For more details on the course and the dataset, please refer to the dataset chapter [44] of the book.

3.3 An Overview of the Required Tools (R Packages)

In addition to a set of tidyverse packages that facilitate general purpose data exploration, wrangling, and analysis tasks (e.g., `dplyr`, `tidyr`, `ggplot2`, `lubridate`), in this chapter, we will also need a few additional R packages relevant for the prediction modelling tasks:

- The **`caret`** (Classification And REgression Training) package [45] offers a wide range of functions that facilitate the overall process of development and evaluation of prediction models. In particular, it includes functions for data preprocessing, feature selection, model tuning through resampling, estimation of feature importance, and the like. Comprehensive documentation of the package, including tutorials, is available online.[1]

- The **`randomForest`** package [46] provides an implementation of the Random Forest prediction method [47] that can be used both for the classification and regression tasks.
- The **`performance`** package [48] offers utilities for computing indices of model quality and goodness of fit for a range of regression models. In this chapter, it

[1] https://topepo.github.io/caret/.

will be used for estimating the quality of linear regression models. The package documentation, including usage examples, is available online.[2]

- The `corrplot` package [49] allows for seamless visual exploration of correlation matrices and thus facilitates understanding of connections among variables in high dimensional datasets. A detailed introduction to the functionality the package offers is available online.[3]

3.4 Data Preparation and Exploration

The data that will be used for predictive modelling in this chapter originates from the LMS of a blended course on LA. The dataset is publicly available in a GitHub repository,[4] while its detailed description is given in the book's chapter on datasets [44]. In particular, we will make use of learning trace data (stored in the `Events.xlsx` file) and data about the students' final grades (available in the `Results.xlsx` file).

We will start by familiarising ourselves with the data through exploratory data analysis.

```
library(tidyverse)
library(lubridate)
library(rio)
```

After loading the required packages, we will load the data from the two aforementioned data files:

```
events = import(
  "https://github.com/lamethods/data/raw/main/1_moodleLAcourse/Events.xlsx")
results = import(
  "https://github.com/lamethods/data/raw/main/1_moodleLAcourse/Results.xlsx")
```

We will start by exploring the events data, and looking first into its structure:

```
glimpse(events)
```

```
Rows: 95,626
Columns: 7
$ Event.context <chr> "Assignment: Final Project", "Assignment: Final Project"~
$ user          <chr> "9d744e5bf", "91489f7a9", "278a75edf", "53d6ab60c", "aab~
```

[2] https://easystats.github.io/performance/.

[3] https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html.

[4] https://github.com/sonsoleslp/labook-data/tree/main/1_moodleLAcourse.

```
$ timecreated   <dttm> 2019-10-26 09:37:12, 2019-10-26 09:09:34, 2019-10-18 12~
$ Component     <chr> "Assignment", "Assignment", "Assignment", "Assignment", ~
$ Event.name    <chr> "Course module viewed", "The status of the submission ha~
$ Log           <chr> "Assignment: Final Project", "Assignment: Final Project"~
$ Action        <chr> "Assignment", "Assignment", "Assignment", "Assignment", ~
```

Since we intend to build separate predictive models for each week of the course, we need to be able to organise the events data into weeks. Therefore, we will extend the events data frame with additional variables that allow for examining temporal aspects of the course events from the weekly perspective. To that end, we first order the events data based on the events' timestamp (`timecreated`) and then add three auxiliary variables for creating the course_week variable: weekday of the current event (`wday`), weekday of the previous event (`prev_wday`), and indicator variable for the start of a new week (`new_week`). The assumption applied here is that each course week starts on Monday and the beginning of a new week (`new_week`) can be identified by the current event being on Monday (`wday=="Mon"`) while the previous one was on any day other than Monday (`prev_wday!="Mon"`) :

```
events |>
  arrange(timecreated) |>
  mutate(wday = wday(timecreated,
                     label = TRUE,
                     abbr = TRUE,
                     week_start = 1)) |>
  mutate(prev_wday = lag(wday)) |>
  mutate(new_week = ifelse((wday == "Mon") & (is.na(prev_wday) | prev_wday != "Mon"),
                           yes = TRUE, no = FALSE)) |>
  mutate(course_week = cumsum(new_week)) -> events
```

Having created the variable that denotes the week of the course (`course_week`), we can remove the three auxiliary variables, to keep our data frame tidy:

```
events |> select(-c(wday, prev_wday, new_week)) -> events
```

We can now explore the distribution of the events across the course weeks. The following code will give us the count and proportion of events per week (with proportions rounded to the fourth decimal):

```
events |>
  count(course_week) |>
  mutate(prop = round(n/nrow(events), 4))
```

The output of the above lines show that we have data for 7 weeks: 6 weeks of the course plus one more week, right after the course officially ended but students were still able to submit assignments. We can also observe that the level of students' interaction with course activities steadily increased up until week 5 and then started going down.

Let us now move to examining the factor variables that represent different types of actions and logged events. First, we can check how many distinct values each of these variables has:

```
events |>
  summarise(across(c(Event.context, Component:Action), n_distinct))
```

```
  Event.context Component Event.name Log Action
1            80         13         27  80     12
```

We can also examine unique values of each of the four variables, but it is better to examine them together, that will help us better understand how they relate to one another and get a better idea of the semantics of events they denote. For example, we can examine how often distinct *Component*, *Event*, and *Action* values co-occur (Table 1):

```
events |>
  count(Component,Event.name, Action) |>
  arrange(Component, desc(n))
```

Likewise, we can explore how Action and Log values are related (i.e., co-occur) (Table 2):

Table 1 Count of all combinations of Component, Event, and Action

	Component	Event.name	Action	n
1	Assignment	Course module viewed	Practicals	3463
2	Assignment	Course module viewed	Assignment	2926
3	Assignment	The status of the submission has been viewed	Practicals	2698
4	Assignment	The status of the submission has been viewed	Assignment	2427
5	Assignment	Course module viewed	Group_work	676
6...102				
103	Zoom meeting	Clicked join meeting button	Group_work	25

Table 2 Count of all combinations of Log and Action

	Action	Log	n
1	Applications	File: Case studies	141
2	Applications	File: Features students really expect from learning analytics	153
3	Applications	File: OU Analyse: Analysing at-risk students at The Open University	161
4	Applications	File: Whitelock-Wainwright et al-2019-Journal of Computer Assisted Learning	42
5	Applications	URL: E2Coach	90
6..80			
81	Theory	URL: Theory video lecture	207

```
events |>
  count(Action, Log) |>
  arrange(Action)
```

Having explored the four categorical variables that capture information about the students' interactions with course resources and activities, we will select the Action variable as the most suitable one for further analysis. The reason for choosing the Action variable is twofold: (1) it is not overly granular (it has 12 distinct values), and thus allows for the detection of patterns in the learning trace data; (2) it captures sufficient information about the distinct kinds of interaction the events refer to. In fact, the Action variable was manually coded by the course instructor to offer a more nuanced way of analysis. The coding was performed to group actions that essentially indicate the same activities under the same label. For instance, logs of viewing feedback from the teacher were grouped under the label *feedback*. Practical activities (Social network analysis or Process mining) were grouped under the label *practicals*. In the same way, accessing the group work forums designed for collaboration, browsing, reading others' comments, or writing were all grouped under the label *group_work* [50].

We will rename some of the `Action` values to make it clear that they refer to distinct topics of the course materials:

```
topical_action <- c("General", "Applications", "Theory",  "Ethics", "Feedback", "La_types")

events |>
  mutate(action = ifelse(test = Action %in% topical_action,
                         yes = str_glue("Materials_{Action}"),
                         no = Action),
         .keep = "unused") -> events
```

Let us now visually examine the distribution of events across different action types and course weeks:

```
# Compute event counts across action types and course weeks
events |>
  count(course_week, action) |>
  arrange(course_week, desc(n)) -> action_dist_across_weeks

# Visualise the event distribution
action_dist_across_weeks |>
  mutate(Action = as.factor(action)) |>
  ggplot(aes(x = course_week, y = n, fill = action)) +
  geom_col(position = position_fill()) +
  scale_fill_brewer(palette = 'Paired') +
  scale_x_continuous(breaks = seq(1,7)) +
  labs(x = "\nCourse week", y = "Proportion\n", fill ="Action") +
  theme_minimal()
```

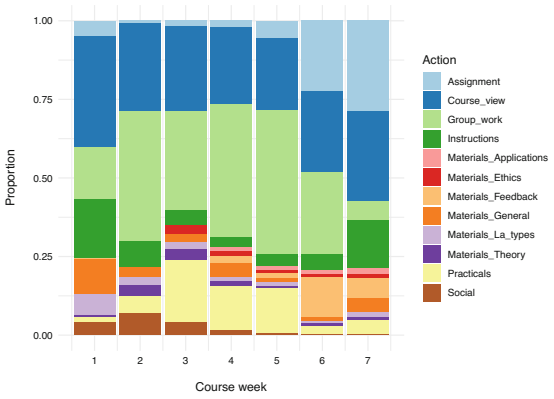


Fig. 1 Distribution of action types across the course weeks

From the plot produced by the above lines of code (Fig. 1), we can observe, for example, that group work (`Group_work`) was the most represented type of actions from week 2 till the end of the course (week 6). It is followed by browsing the main page of the course containing the course materials, announcements and updates (`Course_view`) and working on practical tasks (`Practicals`). We can also note that the assignment-related actions (`Assignment`) are present mostly towards the end of the course.

Now that we have familiarised ourselves with the events data and done some initial data preparation steps, we should do some final 'polishing' of the data and store it to have it ready for further analysis.

```
# Keep only the variables to be used for further analysis and
# Rename some of the remaining ones to keep naming consistent
events |>
  select(user, timecreated, course_week, action) |>
  rename(week = course_week, ts = timecreated) -> events

# Save the prepared data in the R native format
dir.create("preprocessed_data")
saveRDS(events, "preprocessed_data/events.RDS")
```

The next step is to explore the grades data that we previously loaded into the results data frame

```
glimpse(results)
```

```
Rows: 130
Columns: 15
$ user              <chr> "6eba3ff82", "05b604102", "111422ee7", "b4658c3a9",~
$ Grade.SNA_1       <dbl> 0, 8, 10, 5, 10, 7, 9, 10, 10, 10, 7, 10, 9, 9, 9, ~
$ Grade.SNA_2       <dbl> 0, 10, 10, 5, 10, 10, 9, 10, 10, 10, 8, 10, 10, 10,~
$ Grade.Review      <dbl> 6.67, 6.67, 10.00, 0.00, 10.00, 9.67, 6.67, 7.00, 1~
```

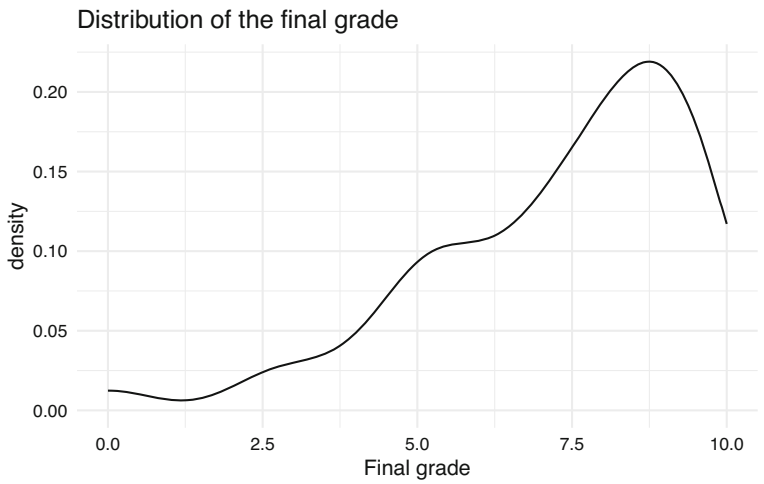


Fig. 2 Distribution of the final course grade

```
$ Grade.Group_self    <dbl> 5, 1, 10, 1, 10, 6, 10, 9, 10, 10, 6, 10, 10, 10, 9~
$ Grade.Group_All     <dbl> 4.00, 3.00, 9.11, 4.00, 9.18, 4.00, 8.56, 8.56, 9.2~
$ Grade.Excercises    <dbl> 10.00, 10.00, 10.00, 10.00, 10.00, 3.33, 10.00, 10.~
$ Grade.Project       <dbl> 0.00, 7.00, 9.33, 6.00, 5.33, 7.67, 0.00, 9.33, 10.~
$ Grade.Literature    <dbl> 6.67, 6.67, 10.00, 4.33, 10.00, 9.67, 5.00, 6.67, 1~
$ Grade.Data          <dbl> 4, 3, 5, 3, 5, 5, 1, 4, 5, 4, 5, 4, 4, 5, 3, 4, 5, ~
$ Grade.Introduction <dbl> 6, 6, 10, 4, 10, 10, 4, 8, 10, 8, 10, 10, 6, 8, 6, ~
$ Grade.Theory        <dbl> 2, 2, 10, 2, 10, 10, 2, 8, 6, 2, 8, 2, 2, 8, 8, 6, ~
$ Grade.Ethics        <dbl> 2, 8, 10, 2, 10, 10, 4, 6, 10, 6, 10, 10, 6, 8, 4, ~
$ Grade.Critique      <dbl> 4, 4, 10, 6, 10, 10, 2, 2, 10, 6, 10, 10, 6, 6, 6, ~
$ Final_grade         <dbl> 2.626970, 4.670169, 9.244600, 0.000000, 8.238179, 5~
```

Even though the results dataset includes the students' grades on individual assignments, we will be able to use just the final grade (`Final_grade`) since we do not have information when during the course the individual assignment grades became available.

To get an overall understanding of the final grade distribution, we will compute the summary statistics and plot the density function for the `Final_grade` variable:

```
summary(results$Final_grade)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.000   5.666   7.954   7.254   9.006  10.000
```

```
ggplot(results, aes(x = Final_grade)) +
  geom_density() +
  labs(x = "Final grade",
       title = "Distribution of the final grade") +
  theme_minimal()
```

We can clearly notice both in the summary statistics and the distribution plot (Fig. 2) that the final grade is not normally distributed, but skewed towards higher grade values.

As noted in Sect. 3.1, we will build two kinds of prediction models: models that predict the final grade (regression models) as well as models that predict whether a student belongs to the group of high or low achievers (classification models). For the latter group of models, we need to create a binary variable (e.g., Course_outcome) indicating if a student is in the high or low achievement group. Students whose final grade is above the 50th percentile (i.e., above the median) will be considered as being high achievers in this course (High), the rest will be considered as having low course achievement (Low):

```
results |>
  mutate(Course_outcome = ifelse(test = Final_grade > median(Final_grade),
                                 yes = "High", no = "Low")) |>
  mutate(Course_outcome = factor(Course_outcome)) -> results
```

Now that we have prepared the outcome variables both for regression and classification models (`Final_grade` and `Course_outcome`, respectively), we can save them for later use in model building:

```
results |>
  select(user, Final_grade, Course_outcome) |>
  saveRDS("preprocessed_data/final_grades.RDS")
```

3.5 *Feature Engineering*

After the data has been preprocessed, we can focus on feature engineering, that is, the creation of new variables (features) to be used for model development. This step needs to be informed by the course design and any learning theory that underpins the course design, so that the features we create and use for predictive modelling are able to capture relevant aspects of the learning process in the given learning settings. In addition, we should consult the literature on predictive modelling in LA (see Sect. 2), to inform ourselves about the kinds of features that were good predictors

in similar learning settings. Following such an approach, we have identified the following event-based features as potentially relevant:

A. Features based on learning action counts

   1. Total number of each type of learning actions
   2. Average number of actions (of any type) per day
   3. Entropy of action counts per day

B. Features based on learning sessions:

   1. Total number of learning sessions
   2. Average (median) session length (time)
   3. Entropy of session length

C. Features based on number of active days (= days with at least one learning session)

   1. Number of active days
   2. Average time distance between two consecutive active days

In addition to the course specific features (A1), the feature set includes several course-design agnostic (i.e., not directly related to a specific course design) features (e.g., A2 and A3) that proved as good predictors in similar (blended) learning settings [14, 16, 36, 51]. Furthermore, the chosen features allow for capturing both the amount of engagement with the course activities (features A1, A2, B1, B2, C1) and regularity of engagement (features A3, B3, C2) at different levels of granularity (actions, sessions, days).

To compute features based on action counts per day (group A), we need to extend the `events` dataset with date as an auxiliary variable:

```
events |> mutate(date = as.Date(ts)) -> events
```

To compute features based on learning sessions, we need to add sessions to the events data. It is often the case that learning management systems and other digital learning platforms do not explicitly log beginning and end of learning sessions. Hence, LA researchers have used heuristics to detect learning sessions in learning events data. An often used approach to session detection consists of identifying overly long periods of time between two consecutive learning actions (of the same student) and considering them as the end of one session and beginning of the next one [14, 16, 36]. To determine such overly long time periods that could be used as "session delimiters", LA researchers would examine the distribution of time periods between consecutive events in a time-ordered dataset, and set the delimiter to the value corresponding to a high percentile (e.g., 85th or 90th percentile) of the time distance distribution. We will rely on this approach to add sessions to the event data.

First, we need to compute time distance between any two consecutive actions of each student:

```
events |>
  group_by(user) |>
  arrange(ts) |>
  mutate(ts_diff = ts - lag(ts)) |>
  ungroup() -> events
```

Next, we should examine the distribution of time differences between any two consecutive actions of each student, to set up a threshold for splitting action sequences into sessions:

```
events |> pull(ts_diff) -> ts_diff

ts_diff_hours = as.numeric(ts_diff, units = 'hours')

summary(ts_diff_hours)
```

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
  0.00000   0.00028   0.00028   1.40238   0.01694 307.05028       130
```

As summary statistics is not sufficiently informative, we should examine the upper percentiles:

```
quantile(ts_diff_hours, probs = seq(0.8, 1, 0.01), na.rm = TRUE) |> round(3)
```

```
    80%     81%     82%     83%     84%     85%     86%     87%     88%     89%
  0.017   0.017   0.034   0.034   0.034   0.050   0.067   0.084   0.117   0.167
    90%     91%     92%     93%     94%     95%     96%     97%     98%     99%
  0.234   0.350   0.617   1.000   1.834   3.367   7.434  14.300  22.035  39.767
   100%
307.050
```

Considering the computed percentile values, on one hand, and the expected length of learning activities in the online part of the course (which included long forums discussions), we will set 1.5 hours (value between 93th and 94th percentile) as the threshold for splitting event sequences into learning sessions:

```
events |>
  mutate(ts_diff_hours = as.numeric(ts_diff, units = 'hours')) |>
  group_by(user) |>
  arrange(ts) |>
  mutate(new_session = (is.na(ts_diff_hours)) | (ts_diff_hours >= 1.5)) |>
  mutate(session_nr = cumsum(new_session))|>
  mutate(session_id = paste0(user,"_", "session_",session_nr)) |>
  ungroup() -> events_with_sessions
```

We will also add session length variable, which can be computed as the difference between the first and last action in each session, as it will be required for the computation of some of the features (B2 and B3):

```
events_with_sessions |>
  group_by(session_id) |>
  mutate(session_len = as.numeric(max(ts) - min(ts), units = "secs")) |>
  ungroup() -> events_with_sessions
```

After adding the necessary variables for feature computation, we can tidy up the dataset before proceeding to the feature computation. In particular, we will keep only the variables required for feature computation:

```
events_with_sessions <- events_with_sessions |>
  select(user, ts, date, week, action, session_nr, session_id, session_len)
```

All functions for computing the event-based features outlined above are given in the *feature_creation* R script. In the following, we give a quick overview of those functions, while the script with further explanations is available at the book's GitHub repository:

- the `total_counts_per_action_type` function computes the set of features labelled as A1, that is, counts of each type of learning action, up to the current week
- the `avg_action_cnt_per_day` function computed feature A2, that is, average (median) number of learning actions per day, up to the current week
- the `daily_cnt_entropy` function computes feature A3, namely entropy of action counts per day, up to the current week
- the `session_based_features` function computes all session-based features up to the current week: total number of sessions (B1), average (median) session length (B2), and entropy of session length (B3)
- the `active_days_count` function computes the number of active days (C1), up to the current week
- the `active_days_avg_time_dist` function computes avg. (median) time distance between two consecutive active days (C2), up to the current week
- finally, the `create_event_based_features` function makes use of the above functions to compute all event-based features, up to the given week

Having defined the feature set and functions for feature computation, cumulatively for each week of the course, we can proceed to the development of predictive models. In the next section (Sect. 3.6), we will present the creation and evaluation of models for predicting the overall course success (high/low), whereas Sect. 3.7 will present models for predicting the final course grade.

3.6 Predicting Success Category

To build predictive (classification) models, we will use Random Forest [47]. This decision was motivated by the general high performance of this algorithm on a

variety of prediction tasks [52] as well as its high performance on prediction tasks specific to the educational domain [43].

Random forest (RF) is a very efficient machine learning method that can be used both for classification and regression tasks. It belongs to the group of ensemble methods, that is, machine learning methods that build and combine multiple individual models to do the prediction. In particular, RF builds and combines the output of several decision or regression trees, depending on the task at hand (classification or regression). The way it works can be briefly explained as follows: the method starts by creating a number of bootstrapped training samples to be used for building a number of decision trees (e.g. 100). When building each tree, each time a split in a tree is to be made, instead of considering all predictors, a random sample of predictors is chosen as split candidates from the full set of predictors (typically the size of the sample is set to be equal to the square root of the number of predictors). The reason for choosing a random sample of predictors is to make a diverse set of trees, which has proven to increase the performance. After all the trees have been built, each one is used to generate a prediction, and those predictions are then aggregated into the overall prediction of the RF model. In case of a classification task, the aggregation of predictions is done through majority vote, that is, the class voted (i.e., predicted) by the majority of the classification trees is the final prediction. In case of a regression task, the aggregation is done by averaging predictions of individual trees. For a thorough explanation of the RF method (with examples in R), an interested reader is referred to Chapter 8 of [53].

We will first load the additional required R packages as well as R scripts with functions for feature computation and model building and evaluation:

```
library(caret)
library(randomForest)

source("feature_creation.R")
source("model_develop_and_eval.R")
```

The following code snippet shows the overall process of model building and evaluation, one model for each week of the course, starting from week 1 to week 5. Note that week 5 is set as the last week for prediction purposes since it is the last point during the course when some pedagogical intervention, informed by the model's output, can be applied by the course instructors.

```
models <- list()
eval_measures <- list()

for(k in 1:5) {

  ds <- create_dataset_for_course_success_prediction(events_with_sessions,
                                                     k, results)
  set.seed(2023)
  train_indices <- createDataPartition(ds$Course_outcome,
```

```
                                        p = 0.8, list = FALSE)
  train_ds <- ds[train_indices,] |> select(-user)
  test_ds <- ds[-train_indices,] |> select(-user)

  rf <- build_RF_classification_model(train_ds)
  eval_rf <- get_classification_evaluation_measures(rf, test_ds)

  models[[k]] <- rf
  eval_measures[[k]] <- eval_rf
}
```

The process consists of the following steps, each of which will be explained in more detail below:

(1) Creation of a dataset for prediction of the course outcomes, based on the logged events data (`events_with_sessions`) up to the given week (`k`) and the available course outcomes data (`results`)
(2) Splitting of the dataset intro the part for training the model (`train_ds`) and evaluating the model's performance (`test_ds`)
(3) Building a RF model based on the training portion of the dataset
(4) Evaluating the model based on the test portion of the dataset

All built models and their evaluation measures are stored (in `models` and `eval_measures` lists) so that they can later be compared.

Going now into details of each step, we start with the creation of a dataset to be used for predictive modelling in week *k*. This is done by first computing all features based on the logged events data (`events_data`) up to the week *k*, and then adding the course outcome variable (`Course_outcome`) from the dataset with course results (`grades`):

```
create_dataset_for_course_success_prediction <- function(events_data,
                                                         current_week,
                                                         grades) {
  features <- create_event_based_features(events_data, current_week)
  grades |>
    select(user, Course_outcome) |>
    inner_join(features, by = "user")
}
```

Next, to be able to properly evaluate the performance of the built model, we need to test its performance on a dataset that the model "has not seen". This requires the splitting of the overall feature set into two parts: one for training the model (training set) and the other for testing its performance (test set). This is done in a way that a larger portion of the dataset (typically 70–80%) is used for training the model, whereas the rest is used for testing. In our case, we use 80% of the feature set for training (`train_ds`) and 20% for evaluation purposes (`test_ds`). Since observations (in this case, students) are randomly selected for the training and

test sets, to assure that we can replicate the obtained results, we initiate the random process with an (arbitrary) value (`set.seed`).

In the next step, we use the training portion of the dataset to build a RF model, as shown in the code snippet below. We train a model by tuning its `mtry` hyper-parameter and choose the model with optimal `mtry` value based on the Area under the ROC curve (AUC ROC) metric. The `mtry` hyper-parameter defines the number of features that are randomly chosen at each step of tree branching, and thus controls how much variability will be present among the trees that RF will build. It is one of the key hyper-parameters for tuning RF models and its default value (`default_mtry`) is equal to the square root of the number of features (`n_features`). Hence, we create a grid that includes the default value and a few values around it.

```
build_RF_classification_model <- function(dataset) {

  #defining the model hyperparameter (mtry) that we want to tune
  n_features <- ncol(dataset)-1
  default_mtry <- round(sqrt(n_features))
  grid <- expand.grid(.mtry = (default_mtry-1):(default_mtry+1))

  #setting that we want to train the model through 10-fold cross-validation
  ctrl <- trainControl(method = "CV",
                       number = 10,
                       classProbs = TRUE,
                       summaryFunction = twoClassSummary)

  # initiating the training process and setting the evaluation measure
  # (ROC) for choosing the best value of the tuned hyperparameter
  rf <- train(x = dataset |> select(-Course_outcome),
              y = dataset$Course_outcome,
              method = "rf",
              metric = "ROC",
              tuneGrid = grid,
              trControl = ctrl)

  rf$finalModel
}
```

The parameter tuning is done through 10-fold cross-validation (CV). K-fold CV is a widely used method for tuning parameters of machine learning models. It is an iterative process, consisting of k iterations, where the training dataset is randomly split into k folds of equal size, and in each iteration, k-1 folds are used for training the model whereas the k-th fold is used for evaluating the model on the chosen performance measure (e.g., ROC AUC, as in our case). In particular, in each iteration, a different fold is used for evaluation purposes, whereas the remaining k-1 folds are used for training. When this iterative process is finished, the models' performance, computed in each iteration, are averaged, thus giving a more stable estimate of the performance for a particular value of the parameter being tuned. CV is often done in 10 iterations, hence the name 10-fold CV.

The final step is to evaluate each model based on the test data. To that end, we compute four standard evaluation metrics for classification models—Accuracy, Precision, Recall, and F1—as shown in the code snippet below. These four metrics are based on the so-called confusion matrix, which is, in fact, a cross-tabulation of the actual and predicted counts for each value of the outcome variable (i.e., class).

```
get_classification_evaluation_measures <- function(model, test_data) {

  # use the model to make predictions on the test set
  predicted_vals <- predict(model,
                            test_data |> select(-Course_outcome))
  actual_vals <- test_data$Course_outcome

  # create the confusion matrix (see Fig. 3)
  cm <- table(actual_vals, predicted_vals)

  TP <- cm[2,2]
  TN <- cm[2,2]
  FP <- cm[1,2]
  FN <- cm[2,1]

  # compute evaluation measures based on the confusion matrix
  accuracy = sum(diag(cm)) / sum(cm)
  precision <- TP / (TP + FP)
  recall <- TP / (TP + FN)
  F1 <- (2 * precision * recall) / (precision + recall)

  c(Accuracy = accuracy,
    Precision = precision,
    Recall = recall,
    F1 = F1)
}
```

In our case, the confusion matrix has the structure as shown on Fig. 3. In rows, it has the counts of the *actual* number of students in the high and low achievement groups, whereas the columns give the *predicted* number of high and low achievers. We consider low course achievement as the positive class, since we are primarily interested in spotting those students who might benefit from a

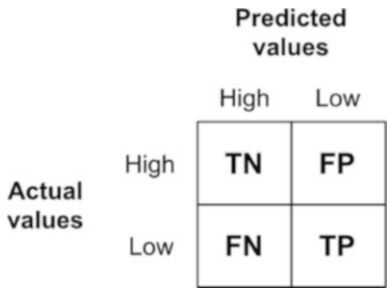


Fig. 3 Confusion matrix for the prediction of the students' overall course success

pedagogical intervention (to prevent a poor course outcome). Hence, TP (True Positive) is the count of students who had low course achievement and were predicted by the model as such. TN (True Negative) is the count of those who were high achieving in the course and the model predicted they would be high achievers. FP (False Positive) is the count of those who were high achievers in the course, but the model falsely predicted that they would have low achievement. Finally, FN (False Negative) is the count of students who were predicted to have high achievement in the course, but actually ended up in the low achievement group. These four count-based values forming the confusion matrix serve as the input for computing the aforementioned standard evaluation measures (Accuracy, Precision, Recall, and F1) based on the formuli given in the code snippet above.

After the predictive models for weeks 1–5 are built and evaluated, we combine and compare their performance measures:

```
eval_df <- bind_rows(eval_measures)
eval_df |>
  mutate(week = 1:5) |>
  mutate(across(Accuracy:F1, \(x) round(x, digits = 4))) |>
  select(week, Accuracy:F1)
```

Table 3 shows the resulting comparison of the built models. According to all measures, models 2 and 3, that is, models with the data from the first two and first 3 weeks of the course, are the best. In other words, the students' interactions with the course activities in the first 2–3 weeks are the most predictive of their overall course success. In particular, the accuracy of these models is 84%, meaning that for 84 out of 100 students, the models will correctly predict if the student would be a high or low achiever in this course. These models have precision of 75%, meaning that out of all the students for whom the models predict will be low achievers in the course, 75% will actually have low course achievement. In other words, the models will underestimate students' performance in 25% of predictions they make, by wrongly predicting that students would have low course achievement. The two best models have perfect recall (100%), meaning that the models would identify all the students who will actually have low course performance. These models outperform the other three models also in terms of the F1 measure, which was expected considering that this measure combines precision and recall giving them equal relevance. Interestingly, the studies exploring predictive models on weekly

Table 3 Comparison of prediction models for successive course weeks

Week	Accuracy	Precision	Recall	F1
1	0.7917	0.7500	0.8182	0.7826
2	0.8400	0.7500	1.0000	0.8571
3	0.8400	0.7500	1.0000	0.8571
4	0.7692	0.7333	0.8462	0.7857
5	0.7692	0.7333	0.8462	0.7857

basis have found similar high predictive power for models developed around the second week of the course [54].

RF allows for estimating the relevance of features used for model building. In a classification task, RF estimates feature relevance as the total decrease in the impurity (measured by the Gini index) of leaf nodes from splitting on a particular feature, averaged over all the trees that a RF model builds [46]. We will use this RF's functionality to compute and plot the importance of features in the best model. The function that does the computation and plotting is given below.

```
compute_and_plot_variable_importance <- function(rf_model) {
  importance(rf_model, type = 2) |>
    as.data.frame() |>
    (function(x) mutate(x, variable = rownames(x)))() -> var_imp_df

  row.names(var_imp_df) <- 1:nrow(var_imp_df)
  colnames(var_imp_df) <- c("importance","variable")

  ggplot(var_imp_df,
         aes(x = reorder(variable, importance), y = importance)) +
    geom_col(width = 0.35) +
    labs(x = "", y = "", title = "Feature importance") +
    coord_flip() +
    theme_minimal()
}
```

The plot produced by this function for one of the best models (Model 2) is given in Fig. 4.

```
compute_and_plot_variable_importance(models[[2]])
```

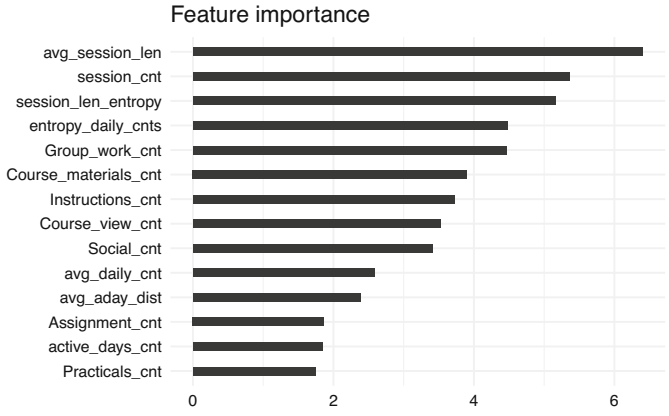


Fig. 4 The importance of features in the best course outcome prediction model, as estimated by the RF algorithm

As Fig. 4 shows, features denoting the overall level of activity (`avg_session_len`, `session_cnt`) are those with the highest predictive power. They are followed by entropy-based features, that is, features reflective of the regularity of study. These findings are in line with the LA literature (e.g., [14, 36, 37]. It should be also noted that the feature reflective of the level of engagement in the group work (`Group_work_cnt`) is among the top 5 predictors, which can be explained by the prominent role of group work in the course design.

3.7 Predicting Success Score

To predict the students' final grades, we will first try to build linear regression models, since linear regression is one of the most often used regression methods in LA [43]. To that end, we will first load a few additional R packages:

```
library(performance)
library(corrplot)
```

Considering that a linear regression model can be considered valid only if it satisfies a set of assumptions that linear regression, as a statistical method, is based upon (linearity, homogeneity of variance, normally distributed residuals, and absence of multicollinearity and influential points), we will first examine if our data satisfies these assumptions. In particular, we will compute the features based on the events data from the first week of the course, build a linear regression model using the computed features, and examine if the resulting model satisfies the assumptions. Note that we limit our initial exploration to the logged events data over the first week of the course since we aim to employ a regression method that can be applied to any number of course weeks; so, if the data from the first course week allow for building a valid linear regression model, we can explore the same method further; otherwise, we need to choose a more robust regression method, that is, method that is not so susceptible to imperfections in the input data.

Having created the dataset for final grade prediction based on the week 1 events data, we will split it into training and test sets (as done for the prediction of the course outcome, Sect. 3.6), and examine correlations among the features. The latter step is due to the fact that one of the assumptions of linear regression is the absence of high correlation among predictor variables. In the code below, we use the `corrplot` function to visualise the computed correlation values (Fig. 5), so that highly correlated variables can be easily observed.

```
ds <- create_dataset_for_grade_prediction(events_with_sessions, 1, results)

set.seed(2023)
train_indices <- createDataPartition(ds$Final_grade, p = 0.8, list = FALSE)
train_ds <- ds[train_indices,] |> select(-user)
```

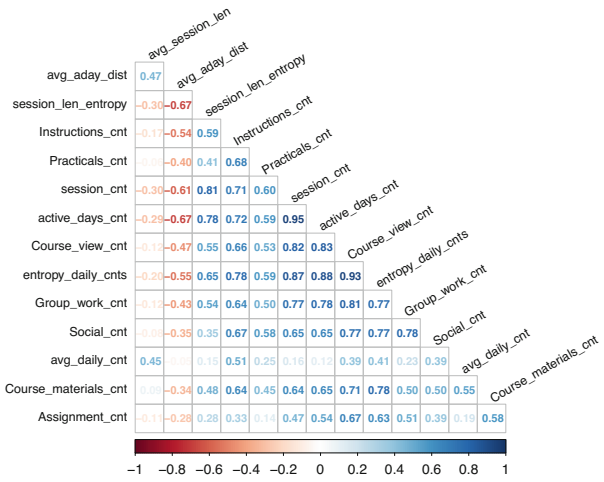


Fig. 5 Correlations among variables in the feature set

```
test_ds <- ds[-train_indices,] |> select(-user)

# examine correlations among the variables: for a linear regression model,
# they must not be highly mutually correlated
corrplot(train_ds |> select(-Final_grade) |> cor(),
         method = "number", type = "lower",
         diag = FALSE, order = 'hclust',
         tl.cex = 0.75, tl.col = 'black', tl.srt = 30, number.cex = 0.65)
```

Figure 5 indicates that there are a couple of features that are highly mutually correlated. These will be removed before proceeding with the model building. While there is no universal agreement on the correlation threshold above which features should be considered overly correlated, correlation coefficients of 0.75 and −0.75 are often used as the cut-off values [53].

```
train_ds |> select(-c(session_cnt, Course_view_cnt,
                      active_days_cnt, entropy_daily_cnts)) -> train_ds_sub
```

We can now build a model and check if it satisfies the assumptions of linear regression:

```
lr <- lm(Final_grade ~ ., data = train_ds_sub)
check_model(lr)
```

The check_model function from the *performance* R package [48] allows for seamless, visual verification of whether the assumptions are met. The output of this function when applied to our linear regression model (`lr`) is shown on Fig. 6.

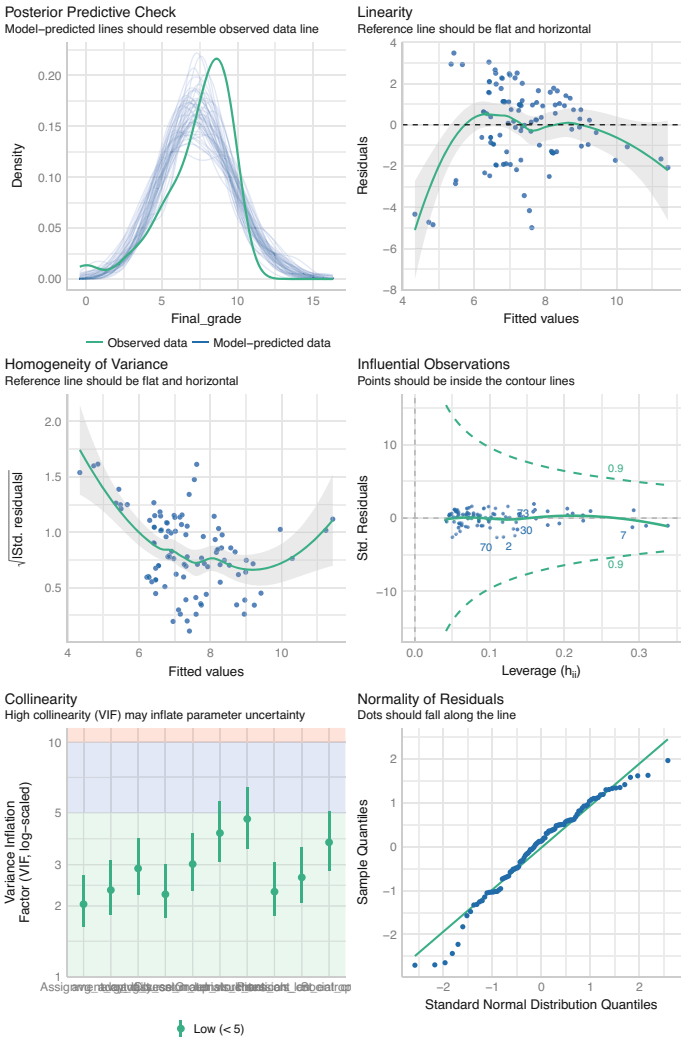


Fig. 6 The output of the check_model function enables visual verification of the assumptions that linear regression is based upon

As the figure shows, two important assumptions of linear models are not met, namely linearity and homoscedasticity (i.e. homogeneity of variance). Therefore, linear regression cannot be used with the given feature set. Instead, we have to use a regression method that does not impose such requirements on the data distribution. Since Random forest is such a method and it has already proven successful with our dataset on the classification task (Sect. 3.6), we will use it to build regression models that predict students' final grades.

Before moving to regression with Random forest, it is worth noting that, in addition to checking all model assumptions at once, using the `check_model` function, one can also check each assumption individually using appropriate functions from the `performance` R package. For example, from Fig. 6, one can not clearly see the X-axis of the collinearity plot and might want to explore this assumption more closely. That can be easily done using the `check_collinearity` function:

```
check_collinearity(lr)
```

```
# Check for Multicollinearity

Low Correlation

                Term  VIF   VIF 95% CI Increased SE Tolerance Tolerance 95% CI
 Course_materials_cnt 3.02 [2.32, 4.08]         1.74      0.33     [0.25, 0.43]
       Group_work_cnt 4.10 [3.09, 5.59]         2.02      0.24     [0.18, 0.32]
     Instructions_cnt 4.70 [3.52, 6.44]         2.17      0.21     [0.16, 0.28]
       Practicals_cnt 2.30 [1.81, 3.08]         1.52      0.43     [0.32, 0.55]
          Social_cnt 3.74 [2.83, 5.09]         1.93      0.27     [0.20, 0.35]
       Assignment_cnt 2.04 [1.63, 2.71]         1.43      0.49     [0.37, 0.61]
        avg_daily_cnt 2.89 [2.23, 3.90]         1.70      0.35     [0.26, 0.45]
      avg_session_len 2.24 [1.77, 3.00]         1.50      0.45     [0.33, 0.56]
 session_len_entropy 2.65 [2.06, 3.56]         1.63      0.38     [0.28, 0.49]
       avg_aday_dist 2.34 [1.84, 3.13]         1.53      0.43     [0.32, 0.54]
```

From the function's output, we can clearly see the VIF (Variance Inflation Factor) values for all the features and a confirmation that the assumption of the absence of multicollinearity is satisfied. The documentation of the `performance`[5] package provides the whole list of functions for different ways of checking regression models.

To build and compare regression models in each course week, we will follow a similar procedure to the one applied when building classification models (Sect. 3.6); the code that implements it is given below. The differences are in the way that the dataset for grade prediction is built (`create_dataset_for_grade_prediction`), the way that regression models are built (`build_RF_regression_model`) and evaluated (get_regression_evaluation_measures), and these will be explained in more detail below.

```
regression_models <- list()
regression_eval <- list()
for(k in 1:5) {
  print(str_glue("Starting computations for week {k} as the current week"))

  ds <- create_dataset_for_grade_prediction(events_with_sessions, k, results)

  set.seed(2023)
  train_indices <- createDataPartition(ds$Final_grade, p = 0.8, list = FALSE)
  train_ds <- ds[train_indices,] |> select(-user)
  test_ds <- ds[-train_indices,] |> select(-user)
```

[5] https://easystats.github.io/performance/reference/index.html.

```
  rf <- build_RF_regression_model(train_ds)
  eval_rf <- get_regression_evaluation_measures(rf, train_ds, test_ds)

  regression_models[[k]] <- rf
  regression_eval[[k]] <- eval_rf
}
```

To create a dataset for final grade prediction in week k, we first compute all features based on the logged events data (events_data) up to the week k, and then add the final grade variable (`Final_grade`) from the dataset with course results (`grades`):

```
create_dataset_for_grade_prediction <- function(events_data, current_week, grades) {
  features <- create_event_based_features(events_data, current_week)
  grades |>
    select(user, Final_grade) |>
    inner_join(features, by = "user")
}
```

As can be observed in the code snippet below, building a RF regression model is very similar to building a RF classification model. The main difference is in the evaluation measure that is used for selecting the optimal *mtry* value in the cross-validation process - here, we are using RMSE (Root Mean Squared Error), which is a standard evaluation measure for regression models [53]. As its name suggests, RMSE is the square root of the average squared differences between the actual and predicted values of the outcome variable on the test set.

```
build_RF_regression_model <- function(dataset) {

  n_features <- ncol(dataset)-1
  default_mtry <- round(sqrt(n_features))
  grid <- expand.grid(.mtry = (default_mtry-1):(default_mtry+1))

  ctrl <- trainControl(method = "CV",
                       number = 10)

  rf <- train(x = dataset |> select(-Final_grade),
              y = dataset$Final_grade,
              method = "rf",
              metric = "RMSE",
              tuneGrid = grid,
              trControl = ctrl)

  rf$finalModel
}
```

Finally, to evaluate each model on the test data, we compute three standard evaluation metrics for regression models, namely RMSE, MAE (Mean Absolute

Error), and R^2. MAE is the average value of the absolute differences between the actual and predicted values of the outcome variable (final grade) on the test set. Finally, R^2 (R-squared) is a measure of variability in the outcome variable that is explained by the given regression model. The computation of the three evaluation measures is shown in the code below.

```
get_regression_evaluation_measures <- function(model, train_ds, test_data) {

  predicted_vals <- predict(model,
                            test_data |> select(-Final_grade))
  actual_vals <- test_data$Final_grade

  # R2 = 1 - RSS/TSS
  # RSS - Residual Sum of Squares
  RSS <- sum((predicted_vals - actual_vals)^2)
  # TSS - Total Sum of Squares
  TSS <- sum((median(train_ds$Final_grade) - actual_vals)^2)
  R2 <- 1 - RSS/TSS

  # RMSE = sqrt(RSS/N)
  RMSE <- sqrt(RSS/nrow(test_ds))

  # MAE = avg(abs(predicted - actual))
  MAE <- mean(abs(predicted_vals - actual_vals))

  c(R2 = R2, RMSE = RMSE, MAE = MAE)
}
```

After the regression models for weeks 1–5 are built and evaluated, we combine and compare their performance measures, with the results reported in Table 4.

```
regression_eval_df <- bind_rows(regression_eval)
regression_eval_df |>
  mutate(WEEK = 1:5) |>
  mutate(across(R2:MAE, \(x) round(x, digits = 4))) |>
  select(WEEK, R2, RMSE, MAE)
```

As shown in Table 4, in this case, we do not have a clear situation as it was with the classification task (Table 3), since the three evaluation measures point to

Table 4 Comparison of grade prediction models for successive course weeks

WEEK	R2	RMSE	MAE
1	0.7315	0.8123	0.6614
2	0.8423	0.6215	0.4531
3	0.8387	0.6285	0.4474
4	0.9067	0.7625	0.5915
5	0.9179	0.7151	0.5777

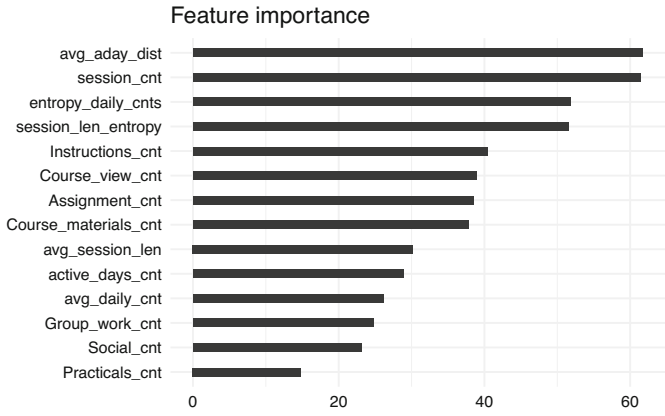


Fig. 7 The importance of features in the best final grade prediction model, as estimated by the RF algorithm

different models as potentially the best ones. In particular, according to R^2, the best model would be model 5 (i.e., the model based on the data from the first 5 weeks), whereas the other two measures point to the second or third model as the best. Considering that (1) RMSE and MAE measures are considered more important than R^2 when evaluating the predictive performance of regression models [13] and (2) RMSE and MAE values for models 2 and 3 are very close, while the second model is better in terms of R^2, we will conclude that the second model, that is, the model based on the logged event data from the first 2 weeks of the course is the best model. This model explains 84.23% of variability in the outcome variable (final grade), and predicts it with an average absolute error of 0.4531, which can be considered a small value with respect to the value range of the final grade [0–10].

To estimate the importance of features in the best regression model, we will again leverage the RF's ability. We will use the same function as before (`compute_and_plot_variable_importance`) to estimate and plot feature importance. The only difference will be that the importance function (from the *randomForest* package) will internally use residual sum of squares as the measure of node impurity when estimating the features importance. Figure 7 shows that, as in the case of predicting the overall course success (Fig. 4), regularity of study features (`avg_aday_dist`, `entropy_daily_cnts`, `session_len_entropy`) are among the most important ones. In addition, the number of learning sessions (`session_cnt`), as an indicator of overall activity in the course, is also among the top predictors.

```
compute_and_plot_variable_importance(regression_models[[2]])
```

4 Concluding Remarks

The results of predictive modelling presented in the previous section show that, in the examined postgraduate course on LA, we can make fairly accurate predictions of the students' course outcomes already in the second week of the course. In fact, both classification and regression models, that is, prediction of the students' overall course success and final grades, proved to be the most accurate when based on the logged learning events data from the first two or three course weeks. That students' learning behaviour in the first part of the course is highly predictive of their course performance, which is in line with related research on predictive modelling (e.g., [54–56]). It should be also noted that the high performance of the presented predictive models can be partially explained by the well chosen feature set and the used algorithm (Random forest) that generally performs well on prediction tasks. However, it may also be due to the relatively large dataset. As noted in Sect. 3.1, we used a synthetic anonymized version of the original dataset that is three times larger than the original dataset.

Considering the features that proved particularly relevant for predicting the students' course performance, we note that in both kinds of predictive tasks—course success and final grade prediction—features reflective of regularity of study stand out. In addition, features denoting the overall level of engagement with online learning activities and resources also have high predictive power. It is also worth noting that the highly predictive features are session level features, suggesting that learning session is the right level of granularity (better than actions or active days) for predictive modelling in the given course. In fact, this finding is in line with earlier studies that examined predictive power of a variety of features derived from learning traces [13, 14, 36]. Note that due to the purpose of this chapter to serve as introductory reading to predictive modelling in LA, we based the feature set on relatively simple features and used only one data source for feature creation. For more advanced and diverse feature creation options, interested readers are referred to, for example [57–59].

The algorithm used for building predictive models, namely Random forest, offers the advantage of flexibility in terms of the kinds of data it can work with (unlike, for example, linear regression which is based on several assumptions about data distribution) as well as fairly good prediction results it tends to produce. On the other hand, the algorithm is not as transparent as simpler algorithms are (e.g., linear regression or decision trees) and thus its use might raise issues of teachers' trust and willingness to rely on the models' output. On the positive side, the algorithm offers an estimate of feature importance thus shedding some light on the underlying "reasoning" process that led to its output (i.e., predictions).

To sum up, predictive modelling, as applied in LA, can bring about important benefits in the form of early in the course detection of students who might be struggling with the course and pointing out indicators of learning behaviour that are associated with poor course outcomes. With such insights available, teachers can make better informed decisions as to the students who need support and the kind of

support they might benefit from. However, predictive modelling is also associated with challenges, especially practical challenges related to the development and use of such models, including availability and access to the data, interpretation of models and their results, and the associated issue of trust in the models' output.

5 Suggested Readings

- Max Kuhn & Julia Silge (2022). *Tidy Modeling with R: A Framework for Modeling in the Tidyverse*. O'Reilly. https://www.tmwr.org/
- Bradley Boehmke & Brandon Greenwell (2020). *Hands-On Machine Learning with R*. Taylor & Francis. https://bradleyboehmke.github.io/HOML/
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An Introduction to Statistical Learning: With Applications in R. 2nd Edition*. Springer US. https://doi.org/10.1007/978-1-0716-1418-1

References

1. Siemens G, Long P (2011) Penetrating the fog: Analytics in learning and education. EDUCAUSE Rev 46:30
2. Siemens G (2013) Learning analytics: The emergence of a discipline. Am Behav Sci 57:1380–1400. https://doi.org/10.1177/0002764213498851
3. Campbell JP, DeBlois PB, Oblinger DG (2007) Academic analytics. Educause Rev 42:40–57
4. Baker RS, Yacef K, et al (2009) The state of educational data mining in 2009: A review and future visions. J. Educ. Data Mining 1:3–17
5. Cornog J, Stoddard GD (1925) Predicting performance in chemistry. J. Chem. Educ. 2:701. https://doi.org/10.1021/ed002p701
6. Tomcsik D, Joksimovic J, Juhász J, Mihályi K (2014) Early warning systems in six European countries desk research report on study visit countries in the frame of CROCOOS-cross-sectoral cooperation focused solutions for the prevention of early school leaving project interim report. https://cpi.si/wp-content/uploads/2020/08/12-Sistemi-zgodnjega-opozarjanja-v-EU-EN.pdf
7. Brooks C, Thompson C (2017) Predictive modelling in teaching and learning. In: Handbook of learning analytics. Society for Learning Analytics Research (SoLAR), pp 61–68
8. Arnold KE, Pistilli MD (2012) Course signals at Purdue. In: Proceedings of the 2nd International Conference on Learning Analytics and Knowledge - LAK '12 267–267. https://doi.org/10.1145/2330601.2330666
9. Caulfield M (2013) What the course signals "kerfuffle" is about, and what it means to you. EDUCAUSE edu
10. Kuzilek J, Hlosta M, Herrmannova D, Zdrahal Z, Vaclavek J, Wolff A (2015) OU analyse: Analysing at-risk students at the open university. Learn Anal Rev LAK15-1:1–16
11. Ifenthaler D, Yau JYK (2020) Utilising learning analytics to support study success in higher education: A systematic review. Educ Technol Res Dev ETR & D. https://doi.org/10.1007/s11423-020-09788-z
12. Gašević D, Dawson S, Rogers T, Gasevic D (2016) Learning analytics should not promote one size fits all: The effects of instructional conditions in predicting academic success. Internet High Educ 28:68–84. https://doi.org/10.1016/j.iheduc.2015.10.002

13. Conijn R, Snijders C, Kleingeld A, Matzat U (2017) Predicting student performance from LMS data: A comparison of 17 blended courses using moodle LMS. IEEE Trans Learn Technol 10:17–29. https://doi.org/10.1109/TLT.2016.2616312
14. Saqr M, Jovanović J, Viberg O, Gašević D (2022) Is there order in the mess? A single paper meta-analysis approach to identification of predictors of success in learning analytics. Stud High Educ 47:2370–2391. https://doi.org/10.1080/03075079.2022.2061450
15. Finnegan C, Morris LV, Lee K (2008) Differences by course discipline on student behavior, persistence, and achievement in online courses of undergraduate general education. J College Student Retention Res Theory Pract 10:39–54. https://doi.org/10.2190/CS.10.1.d
16. Jovanović J, Saqr M, Joksimović S, Gašević D (2021) Students matter the most in learning analytics: The effects of internal and instructional conditions in predicting academic success. Comput Educ 172:104251. https://doi.org/10.1016/j.compedu.2021.104251
17. Ahmad A, Schneider J, Griffiths D, Biedermann D, Schiffner D, Greller W, Drachsler H (2022) Connecting the dots – a literature review on learning analytics indicators from a learning design perspective. J Comput Assisted Learn. https://doi.org/10.1111/jcal.12716
18. Albreiki B, Zaki N, Alashwal H (2021) A systematic literature review of student' performance prediction using machine learning techniques. Educ Sci 11:552. https://doi.org/10.3390/educsci11090552
19. Shafiq DA, Marjani M, Habeeb RAA, Asirvatham D (2022) Student retention using educational data mining and predictive analytics: A systematic literature review. IEEE Access 10:72480–72503. https://doi.org/10.1109/ACCESS.2022.3188767
20. Wang Q, Mousavi A (2023) Which log variables significantly predict academic achievement? A systematic review and meta-analysis. Br J Educ Technol J Council Educ Technol 54:142–191. https://doi.org/10.1111/bjet.13282
21. Gray CC, Perkins D (2019) Utilizing early engagement and machine learning to predict student outcomes. Comput Educ 131:22–32. https://doi.org/10.1016/j.compedu.2018.12.006
22. Hussain S, Khan MQ (2021) Student-performulator: Predicting students' academic performance at secondary and intermediate level using machine learning. Ann Data Sci. https://doi.org/10.1007/s40745-021-00341-0
23. Nouri J, Larsson K, Saqr M (2019) Identifying factors for master thesis completion and non-completion through learning analytics and machine learning. In: Lecture notes in computer science. Springer International Publishing, Cham, pp 28–39
24. Sani NS, Fikri A, Ali Z, Zakree M, Nadiyah K (2020) Drop-out prediction in higher education among B40 students. Int J Adv Comput Sci Appl. IJACSA 11. https://doi.org/10.14569/ijacsa.2020.0111169
25. Adnan M, Habib A, Ashraf J, Mussadiq S, Raza AA, Abid M, Bashir M, Khan SU (2021) Predicting at-risk students at different percentages of course length for early intervention using machine learning models. IEEE Access 9:7519–7539. https://doi.org/10.1109/ACCESS.2021.3049446
26. Bañeres D, Rodríguez ME, Guerrero-Roldán AE, Karadeniz A (2020) An early warning system to detect at-risk students in online higher education. NATO Adv Sci Inst Ser E Appl Sci 10:4427. https://doi.org/10.3390/app10134427
27. Jorgensen S, Ferraro V, Fichten C, Havel A (2009) Predicting college retention and dropout: Sex and disability. ERIC Clearinghouse
28. Joksimović S, Gašević D, Kovanović V, Riecke BE, Hatala M (2015) Social presence in online discussions as a process predictor of academic performance. J Comput Assisted Learn 31:638–654. https://doi.org/10.1111/jcal.12107
29. Ober TM, Hong MR, Rebouças-Ju DA, Carter MF, Liu C, Cheng Y (2021) Linking self-report and process data to performance as measured by different assessment types. Comput Educ 167:104188. https://doi.org/10.1016/j.compedu.2021.104188
30. Scheffel M, Drachsler H, Kraker J de, Kreijns K, Slootmaker A, Specht M (2017) Widget, widget on the wall, am I performing well at all? IEEE Trans Learn Technol 10:42–52. https://doi.org/10.1109/TLT.2016.2622268

31. Wu Z, Zhao B, Wang Y (2021) Analysis of students' learning behavior under network learning environment. In: 2021 IEEE 3rd international conference on computer science and educational informatization (CSEI), pp 46–50
32. Stadler M, Hofer S, Greiff S (2020) First among equals: Log data indicates ability differences despite equal scores. Comput Hum Behav 111:106442. https://doi.org/10.1016/j.chb.2020.106442
33. Tempelaar D, Rienties B, Nguyen Q (2020) Subjective data, objective data and the role of bias in predictive modelling: Lessons from a dispositional learning analytics application. PloS One 15:e0233977. https://doi.org/10.1371/journal.pone.0233977
34. You JW (2016) Identifying significant indicators using LMS data to predict course achievement in online learning. Internet High Educ 29:23–30. https://doi.org/10.1016/j.iheduc.2015.11.003
35. Zarrabi F, Bozorgian H (2020) EFL students' cognitive performance during argumentative essay writing: A log-file data analysis. Comput Compos 55:102546. https://doi.org/10.1016/j.compcom.2020.102546
36. Jovanovic J, Mirriahi N, Gašević D, Dawson S, Pardo A (2019) Predictive power of regularity of pre-class activities in a flipped classroom. Comput Educ 134:156–168. https://doi.org/10.1016/j.compedu.2019.02.011
37. Saqr M, Fors U, Tedre M (2017) How learning analytics can early predict under-achieving students in a blended medical education course. Medical Teacher 39:757–767. https://doi.org/10.1080/0142159X.2017.1309376
38. Agudo-Peregrina ÁF, Iglesias-Pradas S, Conde-González MÁ, Hernández-García Á (2014) Can we predict success from log data in VLEs? Classification of interactions for learning analytics and their relation with performance in VLE-supported F2F and online learning. Comput Hum Behav 31:542–550. https://doi.org/10.1016/j.chb.2013.05.031
39. Ho LC, Jin Shim K (2018) Data mining approach to the identification of at-risk students. In: 2018 IEEE international conference on big data (big data), pp 5333–5335
40. Jokhan A, Sharma B, Singh S (2019) Early warning system as a predictor for student performance in higher education blended courses. Stud High Educ 44:1900–1911. https://doi.org/10.1080/03075079.2018.1466872
41. Asselman A, Khaldi M, Aammou S (2021) Enhancing the prediction of student performance based on the machine learning XGBoost algorithm. Interactive Learn Environ 1–20. https://doi.org/10.1080/10494820.2021.1928235
42. Badal YT, Sungkur RK (2023) Predictive modelling and analytics of students' grades using machine learning algorithms. Educ Inf Technol 28:3027–3057. https://doi.org/10.1007/s10639-022-11299-8
43. Sghir N, Adadi A, Lahmer M (2022) Recent advances in predictive learning analytics: A decade systematic review (2012–2022). Educ Inf Technol 1–35. https://doi.org/10.1007/s10639-022-11536-0
44. López-Pernas S, Saqr M, Conde J, Del-Río-Carazo L (2024, this volume) A broad collection of datasets for educational research training and application. In: Saqr M, López-Pernas S (eds) Learning analytics methods and tutorials: A practical guide using R. Springer
45. Kuhn M (2008) Building predictive models in r using the caret package. J Stat Softw 28:1–26. https://doi.org/10.18637/jss.v028.i05
46. Liaw A, Wiener M (2002) Classification and regression by randomForest. R news 2:18–22
47. Breiman L (2001) Random forests. Mach Learn 45:5–32. https://doi.org/10.1023/A:1010933404324
48. Lüdecke D, Ben-Shachar M, Patil I, Waggoner P, Makowski D (2021) Performance: An r package for assessment, comparison and testing of statistical models. J Open Source Softw 6:3139. https://doi.org/10.21105/joss.03139
49. Wei T, Simko V (2021). R package corrplot: Visualization of a Correlation Matrix. (Version 0.92). https://github.com/taiyun/corrplot
50. Saqr M, López-Pernas S (2024) Why learning and teaching learning analytics is hard: An experience from a real-life LA course using LA methods. In: Proceedings of the eleventh international conference on technological ecosystems for enhancing multiculturality (TEEM'23). Springer, in press

51. Gitinabard N, Xu Y, Heckman S, Barnes T, Lynch CF (2019) How widely can prediction models be generalized? Performance prediction in blended courses. IEEE Trans Learn Technol 12:184–197. https://doi.org/10.1109/TLT.2019.2911832
52. Fernandez-Delgado M, Cernadas E, Barro S, Amorim D (2014) Do we need hundreds of classifiers to solve real world classification problems? J Mach Learn Res JMLR 15:3133–3181
53. James G, Witten D, Hastie T, Tibshirani R (2021) An introduction to statistical learning: With applications in r. Springer US
54. Saqr M, Nouri J (2020) High resolution temporal network analysis to understand and improve collaborative learning. In: Proceedings of the tenth international conference on learning analytics & knowledge. ACM, New York, NY, USA, pp 314–319
55. Chen W, Brinton CG, Cao D, Mason-Singh A, Lu C, Chiang M (2019) Early detection prediction of learning outcomes in online short-courses via learning behaviors. IEEE Trans Learn Technol 12:44–58. https://doi.org/10.1109/TLT.2018.2793193
56. Jovanović J, Dawson S, Joksimović S, Siemens G (2020) Supporting actionable intelligence: Reframing the analysis of observed study strategies. In: Proceedings of the tenth international conference on learning analytics & knowledge. Association for Computing Machinery, New York, NY, USA, pp 161–170
57. Bulut O, Gorgun G, Yildirim-Erbasli SN, Wongvorachan T, Daniels LM, Gao Y, Lai KW, Shin J (2023) Standing on the shoulders of giants: Online formative assessments as the foundation for predictive learning analytics models. Br J Educ Technol J Council Educ Technol 54:19–39. https://doi.org/10.1111/bjet.13276
58. Deeva G, De Smedt J, De Weerdt J (2022) Educational sequence mining for dropout prediction in MOOCs: Model building, evaluation, and benchmarking. IEEE Trans Learn Technol 15:720–735. https://doi.org/10.1109/TLT.2022.3215598
59. Marras M, Vignoud JTT, Kaser T (2021) Can feature predictive power generalize? Benchmarking early predictors of student success across flipped and online courses. In: Proceedings of the 14th international conference on educational data mining, pp 150–160

Dissimilarity-Based Cluster Analysis of Educational Data: A Comparative Tutorial Using R

Keefe Murphy, Sonsoles López-Pernas, and Mohammed Saqr

1 Introduction

Cluster analysis is a term used to describe a broad range of techniques which have the goal of uncovering groups of observations in a data set. The typical aim of cluster analysis is to categorise observations into groups in such a way that observations within the same group are in some sense more similar to each other, while being relatively dissimilar to those in other groups [1]. In other words, clustering methods uncover group structure in heterogeneous populations by identifying more homogeneous groupings of observations which may represent distinct, meaningful subpopulations. Using machine learning terminology, cluster analysis corresponds to unsupervised learning, whereby groups are identified by relying solely on the intrinsic characteristics (typically dissimilarities), without guidance from unavailable ground truth group labels. Indeed, foreknowledge of the fixed number of groups in a data set is characteristic of supervised learning and the distinct field of classification analysis.

It is important to note that there is no universally applicable definition of what constitutes a cluster [2, 3]. Indeed, in the absence of external information in the form of existing "true" group labels, different clustering methods can reveal different things about the same data. There are many different ways to cluster the same data set, and different methods may yield solutions with different assignments, or even differ in the number of groups they identify. Consequently, we present several cluster analysis algorithms in this chapter; namely, K-means

K. Murphy (✉)
Department of Mathematics and Statistics, Hamilton Institute, Maynooth University, Maynooth, Ireland
e-mail: keefe.murphy@mu.ie

S. López-Pernas · M. Saqr
School of Computing, University of Eastern Finland, Joensuu, Finland

M. Saqr, S. López-Pernas (eds.), *Learning Analytics Methods and Tutorials*,
https://doi.org/10.1007/978-3-031-54464-4_8

[4] in Sect. 3.1, a generalisation thereof, K-medoids [5], in Sect. 3.1.2.3, and agglomerative hierarchical clustering [6] in Sect. 3.2. We apply each method in a comparative study in our tutorial using R [7], with applications to a data set about the participants from discussion forum of a massive open online course (MOOC) for teachers.

The clustering methods we review in this chapter are designed to find mutually exclusive, non-overlapping groups in a data set, i.e., they recover a hard partition whereby each observation belongs to one group only. This is in contrast to soft clustering approaches under which each observation is assigned a probability of belonging to each group. One such example of soft clustering is the model-based clustering paradigm, which is discussed in the later Chapter 9 [8]. While model-based clustering methods, as the name suggests, are underpinned by the assumption of generative probabilistic models [9], the more traditional dissimilarity-based methods on which this chapter is focused are purely algorithmic in nature and rely on heuristic criteria regarding the pairwise dissimilarities between objects.

Heuristic clustering algorithms can be further divided into two categories; partitional clustering (e.g., K-means and K-medoids) and hierarchical (of which we focus on the agglomerative variant). Broadly speaking, partitional clustering methods start with an initial grouping of observations and iteratively update the clustering until the "best" clustering is found, according to some notion of what defines the "best" clustering. Hierarchical clustering methods, on the other hand, is more sequential in nature; a tree-like structure of nested clusterings is built up via successive mergings of similar observations, according to a defined similarity metric. In our theoretical expositions and our applications in the R tutorial, we provide some guidelines both on how to choose certain method-specific settings to yield the best performance and how to determine the optimal clustering among a set of competing methods.

The remainder of this chapter proceeds as follows. In Sect. 2, we review relevant literature in which dissimilarity-based clustering methods have been applied in the realm of educational research. In Sect. 3, we describe the theoretical underpinnings of each method in turn and discuss relevant practical guidelines which should be followed to secure satisfactory performance from each method throughout. Within Sect. 4, we introduce the data set of our case study in Sect. 4.1 and give an overview of some required pre-processing steps in Sect. 4.1.1, and then present a tutorial using R for each clustering method presented in this chapter in Sect. 4.2, with a specific focus on identifying the optimal clustering solution in Sect. 4.2.4. Finally, we conclude with a discussion and some recommendations for related further readings in Sect. 5, with a particular focus on some limitations of dissimilarity-based clustering which are addressed by other frameworks in the broader field of cluster analysis.

2 Clustering in Education: Review of the Literature

In education, clustering is among the oldest and most common analysis methods, predating the field of learning analytics and educational data mining by several decades. Such early adoption of clustering is due to the immense utility of cluster analysis in helping researchers to find patterns within data, which is a major pursuit of education research [10]. Interest was fueled by the increasing attention to heterogeneity and individual differences among students, their learning processes, and their approaches to learning [11, 12]. Finding such patterns or differences among students allows teachers and researchers to improve their understanding of the diversity of students and tailor their support to different needs [13]. Finding subgroups within cohorts of students is a hallmark of so-called "person-centered methods", to which clustering belongs [8].

A person-centered approach stands in contrast to the variable-centered methods which consider that most students belong to a coherent homogeneous group with little or negligible differences [14]. Variable-centered methods assume that there is a common average pattern that represents all students, that the studied phenomenon has a common causal mechanism, and that the phenomenon evolves in the same way and results in similar outcomes amongst all students. These assumptions are largely understood to be unrealistic, "demonstrably false, and invalidated by a substantial body of uncontested scientific evidence" [15]. In fact, several analytical problems necessitate clustering, e.g., where opposing patterns exist [16]. For instance, in examining attitudes towards an educational intervention using variable-centered methods, we get an average that is simply the sum of negative and positive attitudes. If the majority of students have a positive attitude towards the proposed intervention—combined with a minority against—the final result will imply that students favour such intervention. The conclusions of this approach are not only wrong but also dangerous as it risks generalising solutions to groups to whom it may cause harm.

Therefore, clustering has become an essential method in all subfields of education (e.g., education psychology, education technology, and learning analytics) having operationalised almost all quantitative data types and been integrated with most of the existing methods [10, 17]. For instance, clustering became an essential step in sequence analysis to discover subsequences of data that can be understood as distinct approaches or strategies of students' behavior [18]. Similar applications can be found in social network analysis data to identify collaborative roles, or in multimodal data analysis to identify moments of interest (e.g., synchrony). Similarly, clustering has been used with survey data to find attitudinal patterns or learning approaches to mention a few [17]. Furthermore, identifying patterns within students' data is a prime educational interest in its own right and therefore, has been used extensively as a standalone analysis technique to identify subgroups of students who share similar interests, attitudes, behaviors, or background.

Clustering has been used in education psychology for decades to find patterns within self-reported questionnaire data. Early examples include the work of Beder

and Valentine [19] who used responses of a motivational questionnaire to discover subgroups of students with different motivational attitudes. Similarly, Clément et al. [20] used the responses of a questionnaire that assessed anxiety and motivation towards learning English as a second language to find clusters which differentiated students according to their attitudes and motivation. Other examples include the work by Fernandez-Rio et al. [21] who used hierarchical clustering and K-means to identify distinct student profiles according to their perception of the class climate. With the digitalisation of educational institutions, authors also sought to identify students profiles using admission data and learning records. For instance, Cahapin et al. [22] used several algorithms such as K-means and agglomerative hierarchical clustering to identify patterns in students' admission data.

The rise of online education opened many opportunities to investigate students' behavior in learning management systems based on the trace log data that students leave behind them when working on online activities. An example is the work by Saqr et al. [23], who used K-means to cluster in-service teachers' approaches to learning in a MOOC according to their frequency of clicks on the available learning activities. Recently, research has placed a focus on the temporality of students' activities, rather than the mere count. For this purpose, clustering has been integrated within sequence analysis to find subsequences which represent meaningful patterns of students behavior. For instance, Jovanovic et al. [24] used hierarchical clustering to identify distinct learning sequential patterns based on students' online activities in a learning management system within a flipped classroom, and found a significant association between the use of certain learning sequences and learning outcomes. Using a similar approach, López-Pernas et al. [25] used hierarchical clustering to identify distinctive learning sequences in students' use of the learning management system and an automated assessment tool for programming assignments. They also found an association between students' activity patterns and their performance in the course final exam. Several other examples exist for using clustering to find patterns within sequence data [16, 26, 27].

A growing application of clustering can be seen in the study of computer-supported collaborative learning. Saqr and López-Pernas [28] used cluster analysis to discover students with similar emergent roles based on their forum interaction patterns. Using the K-means algorithm and students' centrality measures in the collaboration network, they identified three roles: influencers, mediators, and isolates. Perera et al. [29] used K-means to find distinct groups of similar teams and similar individual participating students according to their contributions in an online learning environment for software engineering education. They found that several clusters which shared some distinct contribution patterns were associated with more positive outcomes. Saqr and López-Pernas [30] analysed the temporal unfolding of students' contributions to group discussions. They used hierarchical clustering to identify patterns of distinct students' sequences of interactions that have a similar start and found a relationship between such patterns and student achievement.

An interesting application of clustering in education concerns the use of multi-modal data. For instance, Vieira et al. [31] used K-means clustering to find patterns in students' presentation styles according to their voice, position, and posture

data. Other innovative uses involve students' use of educational games [32, 33], virtual reality [34], or artificial intelligence [35]. Though this chapter illustrates traditional dissimilarity-based clustering algorithms with applications using R to data on the centrality measures of the participants of a MOOC, along with related demographic characteristics, readers are also encouraged to read Chapter 9 [8] and Chapter 10 [18] in which further tutorials are presented and additional literature is reviewed, specifically in the contexts of clustering using the model-based clustering paradigm and clustering longitudinal sequence data, respectively. We now turn to an explication of the cluster analysis methodologies used in this chapter's tutorial.

3 Clustering Methodology

In this section, we describe some of the theory underpinning the clustering methods used in the later R tutorial in Sect. 4. We focus on some of the most widely-known heuristic dissimilarity-based clustering algorithms; namely, K-means, K-medoids, and agglomerative hierarchical clustering. In Sect. 3.1, we introduce K-means clustering by describing the algorithm, outline the arguments to the relevant R function `kmeans()`, and discuss some of the main limitations and practical concerns researchers should be aware of in order to obtain the best performance when running K-means. We also discuss the related K-medoids algorithm and the associated function `pam()` in the `cluster` library [36] in R, and situate this method in the context of an extension to K-means designed to overcome its reliance on squared Euclidean distances. In Sect. 3.2, we introduce agglomerative hierarchical clustering and the related R function `hclust()`, while outlining various choices available to practitioners and their implications. Though method-specific strategies of choosing the optimal number of clusters K are provided throughout Sects. 3.1 and 3.2, we offer a more detailed treatment of this issue in Sect. 3.3, particularly with regard to criteria that can guide the choice of clustering solution among multiple competing methodologies, and not only the choice of the number of clusters K for a given method.

3.1 K-Means

K-means is a widely-used clustering algorithm in learning analytics and indeed data analysis and machine learning more broadly. It is an unsupervised technique that seeks to divide a typically multivariate data set into some pre-specified number of clusters, based on the similarities between observations. More specifically, K-means aims to partition n objects $\mathbf{x}_1, \ldots, \mathbf{x}_n$, each having measurements on $j = 1, \ldots, d$ strictly continuous covariates, into a set of K groups $\mathcal{C} = \{C_1, \ldots, C_K\}$, where C_k is the set of n_k objects in cluster k and the number of groups $K \leq n$ is pre-specified by the practitioner and remains fixed. K-means constructs these partitions in such a

way that the squared Euclidean distance between the row vector for observations in a given cluster and the centroid (i.e., mean vector) of the given cluster are smaller than the distances to the centroids of the remaining clusters. In other words, K-means aims to learn both the cluster centroids and the cluster assignments by minimising the within-cluster sums-of-squares (i.e., variances). Equivalently, this amounts to maximising the between-cluster sums-of-squares, thereby capturing heterogeneity in a data set by partitioning the observations into homogeneous groups. What follows is a brief technical description of the K-means algorithm in Sect. 3.1.1, after which we describe some limitations of K-means and discuss practical concerns to be aware of in order to optimise the performance of the method in Sect. 3.1.2. In particular, we present the widely-used K-medoids extension in Sect. 3.1.2.3.

3.1.1 K-Means Algorithm

The origins of K-means are not so straightforward to summarise. The name was initially used by James MacQueen in 1967 [4]. However, the standard algorithm was first proposed by Stuart Lloyd in a Bell Labs technical report in 1957, which was later published as a journal article in 1982 [37]. In order to understand the ideas involved, we must first define some relevant notation. Let $\mu_k^{(j)}$ denote the centroid value for the j-th variable in cluster C_k by

$$\mu_k^{(j)} = \frac{1}{n_k} \sum_{\mathbf{x}_i \in C_k} x_{ij}$$

and the complete centroid vector for cluster C_k by

$$\boldsymbol{\mu}_k = \left(\mu_k^{(1)}, \ldots, \mu_k^{(p)}\right)^\top.$$

These centroids therefore correspond to the arithmetic mean vector of the observations in cluster C_k. Finding both these centroids $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and the clustering partition $\mathcal{C}$ is computationally challenging and typically proceeds by iteratively alternating between allocating observations to clusters and then updating the centroid vectors. Formally, the objective is to minimise the total within-cluster sum-of-squares

$$\sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2, \tag{1}$$

where $\|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2 = \sum_{j=1}^{p} \left(x_{ij} - \mu_k^{(j)}\right)^2$ denotes the squared Euclidean distance to the centroid $\boldsymbol{\mu}_k$—such that $\|\cdot\|_2$ denotes the ℓ^2 norm—and $\mathbf{z}_i = (z_{i1}, \ldots, z_{iK})^\top$ is a latent variable such that z_{ik} denotes the cluster membership of observation i;

$z_{ik} = 1$ if observation i belongs to cluster C_k and $z_{ik} = 0$ otherwise. This latent variable construction implies $\sum_{i=1}^{n} z_{ik} = n_k$ and $\sum_{k=1}^{K} z_{ik} = 1$, such that each observation belongs wholly to one cluster only. As the total variation in the data, which remains fixed, can be written as a sum of the total within-cluster sum-of-squares (TWCSS) from Eq. (1) and the between-cluster sum-of-squares as follows

$$\sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})^2 = \sum_{i=1}^{n} \sum_{k=1}^{K} z_{ik} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2 + \sum_{k=1}^{K} n_k \|\boldsymbol{\mu}_k - \bar{\mathbf{x}}\|_2^2,$$

where $\bar{\mathbf{x}}$ denotes the overall sample mean vector, minimising the TWCSS endeavours to ensure that observations in the same cluster are maximally similar to observations in the same cluster and maximally dissimilar to those in other clusters.

Using the notation just introduced, a generic K-means algorithm would proceed as follows:

1. *Initialise*: Select the number of desired clusters K and define K initial centroid vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$.
2. *Allocate*: Find the optimal z_{ik} values that minimise the objective, holding the μ_k values fixed.

Calculate the squared Euclidean distance between each observation and each centroid vector and allocate each object to the cluster corresponding to the initial centroid to which it is closest in terms of squared Euclidean distance. Looking at the objective function in Eq. (1) closely and examining the contribution of observation i, we need to choose the value of $\mathbf{z}_i$ which minimises the expression $\sum_{k=1}^{K} z_{ik} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2$. This is achieved by setting $z_{ik} = 1$ for the value of k that has smallest $\|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2$ and setting $z_{ik'} = 0 \, \forall \, k' \neq k$ everywhere else.

3. *Update*: Find the optimal $\boldsymbol{\mu}_k$ values that minimise the objective, holding the z_{ik} values fixed.

If we re-write the objective function in Eq. (1) as

$$\sum_{i=1}^{n} \sum_{k=1}^{K} \sum_{j=1}^{p} z_{ik} \left(x_{ij} - \mu_k^{(j)} \right)^2,$$

we can use the fact that

$$\frac{\partial}{\partial \mu_k^{(j)}} \left(x_{ij} - \mu_k^{(j)} \right)^2 = -2 \left(x_{ij} - \mu_k^{(j)} \right)$$

to obtain

$$\frac{\partial}{\partial \mu_k^{(j)}} \sum_{i=1}^{n} \sum_{k=1}^{K} \sum_{j=1}^{p} z_{ik} \left(x_{ij} - \mu_k^{(j)} \right)^2 = -2 \sum_{i=1}^{n} z_{ik} \left(x_{ij} - \mu_k^{(j)} \right).$$

Solving this expression for $\mu_k^{(j)}$ yields

$$\mu_k^{(j)} = \frac{\sum_{i=1}^{n} z_{ik} x_{ij}}{\sum_{i=1}^{n} z_{ik}} = \frac{1}{n_k} \sum_{\mathbf{x}_i \in C_k} x_{ij}.$$

4. *Iterate*: One full iteration of the algorithm consists of an allocation step (Step 2) and an update step (Step 3). Steps 2 and 3 are repeated until no objects can be moved between clusters, at which point the algorithm has converged to at least a local minimum of Eq. (1).

Upon convergence, we obtain not only the estimated partition $\mathcal{C} = \{C_1, \ldots, C_K\}$, indicating the cluster membership of each observation, but also the estimated cluster centroids $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ which act as cluster prototypes, efficiently summarising the characteristics of each cluster. The algorithm just described is just one standard variant of K-means. there have been several algorithms proposed for the same objective which derive their names from the author who proposed them; namely MacQueen [4], Lloyd [37], Forgy [38], and Hartigan and Wong [39]. They differ in some subtle ways, particularly with regard to how initial centroids in Step 1 are chosen and whether Steps 3 and 4 are applied to all n observations simultaneously or whether allocations and centroids are updated for each observation one-by-one (i.e., some variants iterate over Step 2 and Step 3 in a loop over $i = 1, \ldots, n$). Without going into further details, we note that the default option for `kmeans()` in R uses the option `algorithm="Hartigan-Wong"` and this is what we will henceforth adopt throughout.

3.1.2 *K*-means Limitations and Practical Concerns

Though K-means is a useful tool in many application contexts due to its conceptual and computational simplicity—so ubiquitous, in fact, that the `kmeans()` function in R is available without loading any additional libraries—it suffers from numerous limitations and some care is required in order to obtain reasonable results. We now discuss some of the main limitations in turn, but note that each is addressed explicitly throughout the K-means application portion of the R tutorial in Sect. 4.2.1. The concerns relate to choosing K (Sect. 3.1.2.1), choosing initial centroids $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ (Sect. 3.1.2.2), and relaxing the reliance on squared Euclidean distances with the more general K-medoids method (Sect. 3.1.2.3).

3.1.2.1 Fixed K and the Elbow Method

The first major drawback is that the number of clusters K must be pre-specified. This is a key input parameter: if K is too low, dissimilar observations will be wrongly

grouped together; if K is too large, observations will be partitioned into many small, similar clusters which may not be meaningfully different. Choosing the optimal K necessitates running the algorithm at various fixed values of K and finding the single value of K which best balances interpretability, parsimony, and fit quality. Fit quality is measured by the TWCSS, i.e., the objective function in Eq. (1). Increasing K indefinitely will cause the TWCSS to decrease indefinitely, but this is not what we want. Instead, we seek a K value beyond which the decrease in TWCSS is minimal, in order to yield a parsimonious solution with a reasonable number of clusters to interpret, without overfitting the data or merely subdividing the actual groups. Thus, a commonly used heuristic graphical method for determining the optimal K value is to plot a range of K values against the corresponding obtained TWCSS values and look for an "elbow" or kink in the resulting curve. Such a plot will guide the choice of K in the K-means portion of R tutorial which follows in Sect. 4.2.1.

3.1.2.2 Initialisation and K-means^{++}

An especially pertinent limitation which must be highlighted is that K-means is liable to converge to sub-optimal local minima, i.e., it is not guaranteed to converge to the global minimum of the objective function in Eq. (1). Many practitioners have observed that the performance of K-means is particularly sensitive to a good choice of initial cluster centroids in Step 1. Indeed, as different initial settings can lead to different clusterings of the same data, good starting values are vital to the success of the K-means algorithm. Typically, K random vectors are used to define the initial $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ centroids. One means of mitigating (but not completely remedying) the problem is to run K-means with a suitably large number of random starting values and choose the solution associated with the set of initial centroids which minimise the TWCSS criterion.

In order to contextualise this issue, it is prudent to first describe some of the main arguments to the `kmeans()` R function. The following list is a non-exhaustive list of the available arguments to `kmeans()`:

- `x`: a numeric matrix of data or a `data.frame` with all numeric columns.
- `centers`: either the number of clusters K or a set of K initial (distinct) cluster centroids.
- `nstart`: if `centers` is specified as a number, this represents the number of random sets of K initial centroids with which to run the algorithm.
- `iter.max`: the maximum number of allocation/update cycles allowed per set of initial centroids.

The arguments `nstart` and `iter.max` have default values of `1` and `10`, respectively. Thus, a user running `kmeans()` with `centers` specified as a number will, by default, only use one random set of initial centroids, to which the results are liable to be highly sensitive, and will have the algorithm terminate after just ten iterations, regardless of whether convergence was achieved. It would seem be an improvement, therefore, to increase the values of `nstart` and `iter.max` from these defaults.

Fortunately, the function automatically returns the single optimal solution according to the random initialisation which yields the lowest TWCSS when `nstart` exceeds 1.

Though this will generally lead to better results, this approach can be computationally onerous if the number of observations n, number of features d, or size of the range of fixed K values under consideration is large. An alternative strategy, which greatly reduces the computational burden and the sensitivity of the final solution to the initial choices of $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ is to choose a suitable set of informed starting values in a data-driven fashion. To this end, the so-called K-means^{++} algorithm was proposed [40] in order to improve the performance of K-means by replacing Step 1 with an iterative distance-weighting scheme to select the initial cluster centroids. Though there is still randomness inherent in K-means^{++}, this initialisation technique ensures that the initial centroids are well spread out across the data space, which increases the likelihood of converging to the true global minimum. The K-means^{++} algorithm works as follows:

(A) Choose an initial centroid uniformly at random from the rows of the data set.
(B) For each observation not yet chosen as a centroid, compute $D^2(\mathbf{x}_i)$, which represents the squared Euclidean distance between $\mathbf{x}_i$ and the nearest centroid that has already been chosen.
(C) Randomly sample a new observation as a new centroid vector with probability proportional to $D^2(\mathbf{x}_i)$.
(D) Repeat Steps B and C until K centroids have been chosen. If any of the chosen initial centroids are not distinct, add a small amount of random jitter to distinguish the non-unique centroids.
(E) Proceed as per Steps 2–4 of the traditional K-means algorithm (or one of its variants).

Although these steps take extra time, K-means itself tends to converge very quickly thereafter and thus K-means^{++} actually reduces the computational burden. A manual implementation of the K-means^{++} is provided by the function `kmeans_pp()` below. Its output can be used as the `centers` argument when running `kmeans()`.

```
kmeans_pp <- function(X, # data
                      K  # number of centroids
                      ) {

  # sample initial centroid from distinct rows of X
  X       <- unique(as.matrix(X))
  new_center_index <- sample(nrow(X), 1)
  centers <- X[new_center_index,, drop=FALSE]

  # let x be all observations not yet chosen as a centroid
  X <- X[-new_center_index,, drop=FALSE]

  if(K >= 2) {
```

```
    # loop over remaining centroids
    for(kk in 2:K) {

      # calculate distances from all observations to all already chosen
        centroids
      distances <- apply(X, 1, function(x) min(sum((x - centers)^2)))

      # sample new centroid with probability proportional to squared
        Euclidean distance
      probabilities <- distances/sum(distances)
      new_center_index <- sample(nrow(X), 1, prob=probabilities)

      # record the new centroid and remove it for the next iteration
      centers <- rbind(centers, X[new_center_index,, drop=FALSE])
      X <- X[-new_center_index,, drop=FALSE]
    }
  }

  # add random jitter to distinguish non-unique centroids and return
  centers[duplicated(centers)] <- jitter(centers[duplicated(centers)])
  return(centers)
}
```

However, it should be noted that there is still inherent randomness in K-means^{++}—note the use of `sample()` in lines 7 and 22—and the algorithm is liable to produce different initial centroids in different runs on the same data. In effect, K-means^{++} does not remove the burden of random initialisation; it is merely a way to have more informed random initialisations. Thus, it would be prudent to run K-means *with* K-means^{++} initialisation and select the solution which minimises the TWCSS, to transfer the burden of requiring multiple runs of K-means with random starting values to fewer runs of K-means^{++} followed by K-means with more informed starting values. We adopt this strategy in the later R tutorial in Sect. 4.2.1.

3.1.2.3 K-medoids and Other Distances

The K-means objective function in Eq. (1) explicitly relies on squared Euclidean distances and requires all features in the data set to be strictly continuous. An artefact of this distance measure is that it is generally recommended to standardise all features of have a mean of zero and unit variance prior to running K-means. In general, standardisation is advisable if the values are of incomparable units (e.g., height in inches and weight in kilogram). More specifically for K-means, it is desirable to ensure all features have comparable variances to avoid having variables with higher magnitudes and variances dominate the distance calculation and have an undue prominence on the clustering partition obtained. While we employ such normalisation to the data used in our R tutorial when applying some pre-processing steps in Sect. 4.1.1, we note that this is not sufficient to overcome all shortcomings of relying on squared Euclidean distances.

For these reasons and more, K-medoids—otherwise known as partitioning around medoids (PAM)—was proposed as an extension to K-means which allows using any alternative dissimilarity measure [5]. The K-medoids objective function is given by

$$\sum_{i=1}^{n}\sum_{k=1}^{K} z_{ik} d\left(\mathbf{x}_i, \boldsymbol{\psi}_k\right),$$

where $d\left(\mathbf{x}_i, \boldsymbol{\psi}_k\right)$ can be any distance measure rather than squared Euclidean and $\boldsymbol{\psi}_k$ is used in place of the mean vector $\boldsymbol{\mu}_k$. The PAM algorithm works in much the same fashion as K-means, alternating between an allocation step which assigns each observation to the cluster with the closest $\boldsymbol{\psi}_k$ (according to the specified distance measure) and an update step which minimises the within-cluster total distance (WCTD). Notably, when minimising with respect to $\boldsymbol{\psi}_k$, the notion of a cluster centroid $\boldsymbol{\mu}_k$ is redefined as a cluster medoid $\boldsymbol{\psi}_k$, which is selected among the rows of the observed data set, i.e., the medoid is the observation $\mathbf{x}_i$ from which the distance to all other observations currently allocated to the same cluster, according to the specified distance measure, is minimised. Similar to K-means, the medoids obtained at convergence again enable straightforward characterisation of a "typical" observation from each cluster and the elbow method from Sect. 3.1.2.1 can be adapted to guide the choice of K in K-medoids by plotting a range of candidate K values against the within-cluster total distance.

This reformulation has three main advantages. Firstly, the distance $d\left(\mathbf{x}_i, \boldsymbol{\psi}_k\right)$ is not squared, which diminishes the influence of outliers. As K-means relies on squared Euclidean distances, which inflates the distances of atypical observations, and defines centroids as means, it is not robust to outliers. Secondly, by defining the medoids as observed rows of the data, rather than finding the value of $\boldsymbol{\psi}_k$ that minimises in general, which could potentially be difficult to estimate for complex data types or particularly sophisticated dissimilarity measures, the algorithm can be much more computationally efficient. It requires only a pre-calculated pairwise dissimilarity matrix as input. Finally, the flexibility afforded by being able to modify the dissimilarity measure enables data which are not strictly continuous to be clustered. In other words, K-medoids is applicable in cases where the mean is undefined. As examples, one could use the Manhattan or general Minkowski distances as alternatives for clustering continuous data, the Hamming [41], Jaccard [42], or Sørensen-Dice [43, 44] distances for clustering binary data, or the Gower distance [45] for clustering mixed-type data with both continuous and categorical features. The closely related K-modes algorithm [46] has also been proposed specifically for purely categorical data applications, as well as the K-Prototypes algorithm [47] for mixed-type variables applications; neither will considered further in this chapter). As their names suggest, they again redefine the notion of a centroid but otherwise proceed much like K-means and K-medoids.

The function `pam()` in the `cluster` library in R provides an implementation of K-medoids, with options for implementing many recent additional speed improve-

ments and improved initialisation strategies [48]. We will discuss these in the K-medoids portion of the later R tutorial in Sect. 4.2.2. Most dissimilarity measures we will use are implement in the base-R function `dist()`, with the exception of the Gower distance which is implemented in the `daisy()` function in the `cluster` library.

3.2 Agglomerative Hierarchical Clustering

Hierarchical clustering is another versatile and widely-used dissimilarity-based clustering paradigm. Though also dissimilarity-based, hierarchical clustering differs from partitional clustering methods like K-means and K-medoids in that it typically doesn't avail of the notion of computing distances to a central prototype, be that a centroid mean vector or a medoid, but instead greedily builds a hierarchy of clusters based on dissimilarities between observations themselves and sets of observations. Consequently, a hierarchical clustering solution provides a set of partitions, from a single cluster to as many clusters as observations, rather than the single partition obtained by K-means and K-medoids. The results of a hierarchical clustering are usually presented in the form of a dendrogram visualisation, which illustrates the arrangement of the set of partitions visited and can help guide the decision of the optimal single clustering partition to extract. However, hierarchical clustering shares some of the advantages K-medoids has over K-means. Firstly, any valid measure of distance can be used, so it is not restricted to squared Euclidean distances and not restricted to clustering purely continuous data. Secondly, hierarchical clustering algorithms do not require the data set itself as input; all that is required is a matrix of pairwise distances.

Broadly speaking, there are two categories of hierarchical clustering:

- *Agglomerative*: Starting from the bottom of the hierarchy, begin with each observation in a cluster of its own and successively merged pairs of clusters while moving up the hierarchy, until all observations are in one cluster. This approach is sometimes referred to as agglomerative nesting (AGNES; [6]).
- *Divisive*: Starting from the top of the hierarchy, with all observations in one cluster, recursively split clusters while moving down the hierarchy, until all observations are in a cluster of their own. This approach is sometimes referred to as divisive analysis (DIANA; [49]).

However, divisive clustering algorithms such as DIANA are much more computationally onerous for even moderately large data sets. Thus, we focus here on the agglomerative variant of hierarchical clustering, AGNES, which is implemented in both the `agnes()` function in the `cluster` library and the `hclust()` function in base R. We adopt the latter in the hierarchical clustering portion of the R tutorial in Sect. 4.2.3. That being said, even agglomerative hierarchical clustering has significant computation and memory burdens when n is large [50, 51].

There are three key decisions practitioners must make when employing agglomerative hierarchical clustering. The first of these, the distance measure, has already been discussed in the context of K-medoids. We now discuss the other two in turn; namely, the so-called linkage criterion for quantifying the distances between merged clusters as the algorithm moves up the hierarchy, and the criterion used for cutting the resulting dendrogram to produce a single partition.

3.2.1 Linkage Criteria

Agglomerative hierarchical clustering employs two different notions of dissimilarity. There is the distance measure, d, such as Euclidean, Manhattan, or Gower distance, which is used to quantify the distance between pairs of *single* observations in the data set. Different choices of distance measure can lead to markedly different clustering results and it is thus common to run the hierarchical clustering algorithm with different choices of distance measure and compare the results. However, in the agglomerative setting, individual observations are successively merged into clusters. In order to decide which clusters should be combined, it is necessary to quantify the dissimilarity of *sets* of observations as a function of the pairwise distances of observations in the sets. This gives rise to the notion of a linkage criterion. At each step, the two clusters separated by the shortest distance are combined; the linkage criteria is precisely the definition of 'shortest distance' which differentiates different agglomerative approaches. Again, the choice of linkage criterion can have a substantial impact on the result of the clustering so multiple solutions with different combinations of distance measure and linkage criterion should be evaluated.

There are a number of commonly-used linkage criteria which we now describe. A non-exhaustive list of such linkages follows—only those which we use in the later R tutorial in Sect. 4—in which we let $\mathcal{A}$ and $\mathcal{B}$ denote two sets of observations, $|\cdot|$ denote the cardinality of a set, and $d(a, b)$ denote the distance between observations in those corresponding sets according to the chosen distance measure. We note that ties for the maximum or minimum distances for complete linkage and single linkage, respectively, are broken at random.

- *Complete* linkage: Define the dissimilarity between two clusters as the distance between the two elements (one in each cluster) which are furthest away from each other according to the chosen distance measure d:

$$\max_{a \in \mathcal{A}, b \in \mathcal{B}} d(a, b).$$

- *Single* linkage: Define the dissimilarity between two clusters as the distance between the two elements (one in each cluster) which are closest to each other according to the chosen distance measure d:

$$\min_{a \in \mathcal{A}, b \in \mathcal{B}} d(a, b).$$

- *Average* linkage: Define the dissimilarity between two clusters as the average distance according to the chosen distance measure d between all pairs of elements (on in each cluster):

$$\frac{1}{|\mathcal{A}| \times |\mathcal{B}|} \sum_{a \in \mathcal{A}} \sum_{b \in \mathcal{B}} d_(a, b).$$

- *Centroid* linkage: Define the dissimilarity between two clusters $\mathcal{A}$ and $\mathcal{B}$ as the distance, according to the chosen distance measure d, between their corresponding centroid vectors $\boldsymbol{\mu}_{\mathcal{A}}$ and $\boldsymbol{\mu}_{\mathcal{B}}$:

$$d(\boldsymbol{\mu}_{\mathcal{A}}, \boldsymbol{\mu}_{\mathcal{B}}).$$

- *Ward* linkage [52]: Instead of measuring the dissimilarity between clusters directly, define the dissimilarity as the cost of merging two clusters as the increase in total within-cluster variance after merging. In other words, minimise the total within-cluster sum-of-squares by finding the pair of clusters at each step which leads to minimum increase in total within-cluster variance after merging, where $\mathcal{A} \cup \mathcal{B}$ denotes the cluster obtained after merging, with corresponding centroid $\boldsymbol{\mu}_{\mathcal{A} \cup \mathcal{B}}$:

$$\frac{|\mathcal{A}| \times |\mathcal{B}}{|\mathcal{A} \cup \mathcal{B}|} \|\boldsymbol{\mu}_{\mathcal{A}} - \boldsymbol{\mu}_{\mathcal{B}}\|_2^2 = \sum_{x \in \mathcal{A} \cup \mathcal{B}} \|x - \boldsymbol{\mu}_{\mathcal{A} \cup \mathcal{B}}\|_2^2 - \sum_{x \in \mathcal{A}} \|x - \boldsymbol{\mu}_{\mathcal{A}}\|_2^2 - \sum_{x \in \mathcal{B}} \|x - \boldsymbol{\mu}_{\mathcal{B}}\|_2^2.$$

The Ward and centroid linkage criteria differ from the other linkage criteria in that they are typically meant to be used only when the initial pairwise distances between observations are squared Euclidean distances. All of the above linkage criteria are implemented in the function `hclust()`, which is available in R without requiring any add-on libraries to be loaded and specifically performs agglomerative hierarchical clustering. Its main arguments are `d`, a pre-computed pairwise dissimilarity matrix (as can be created from the function `dist()`), and `method`, which specifies the linkage criterion (e.g., `"complete"`, `"single"`, `"average"`, and `"centroid"`). Special care must be taken when employing Ward's linkage criterion as two options are available: `"ward.D"`, which assumes that the initial pairwise distance matrix already consists of *squared* Euclidean distances, and `"ward.D2"`, which assumes the distances are merely Euclidean distances and performs the squaring internally [53].

3.2.2 Cutting the Dendrogram

One might notice that when calling `hclust()`, the number of clusters K is not specified in advance, as it is when calling `kmeans()` or `pam()`. Instead, `hclust()` returns an object which describes the hierarchy of the tree produced by the clustering process. A visualisation of such a tree is referred to as a dendrogram, which can be thought of as a representation of a set of candidate partitions. In a dendrogram representation of an agglomerative hierarchical clustering solution, each observation is initially in a singleton cluster on its own, along the x-axis, according to their similarities. Thereafter, each observation, and subsequently each set of observations, are merged along the y-axis in a nested fashion. The scale along the y-axis is proportional to the distance, according to the chosen linkage criterion, at which two clusters are combined. In the end, the groups formed towards the bottom of the graph are close together, whereas those at the top of the graph are far apart.

Obtaining a single hard partition of objects into disjoint clusters is obtained by cutting the dendrogram horizontally at the corresponding height. In other words, observations are allocated to clusters by cutting the tree at an appropriate height. Generally, the lower this height, the greater the number of clusters (theoretically, there can be as many clusters as there are observations, n), while the greater the height, the lower the number of clusters (theoretically, there can be as few as only one cluster, corresponding to no group structure in the data). Thus, an advantage of hierarchical clustering is that the user need not know K in advance; the user can manually select K after the fact by examining the constructed tree and fine-tuning the output to find clusters with a desired level of granularity. There is no universally applicable rule for determining the optimal height at which to cut the tree, but it is common to select a height in the region where there is the largest gap between merges, i.e., where there is a relatively wide range of distances over which the number of clusters in the resulting partition does not change. This is, of course, very much guided by the visualisation itself.

In R, one can visualise the dendrogram associated with a particular choice of distance measure and linkage criterion by calling `plot()` on the output from `hclust()`. Thereafter, the function `cutree()` can be used to obtain a single partition. This function takes the arguments `tree`, which is the result of a call to `hclust()`, `h` which is the height at which the tree should be cut, and `k` which more directly allows the desired number of clusters to be produced. Specifying `k` finds the corresponding height which yields `k` clusters and overrides the specification of `h`.

However, it is often the case that certain combinations of dissimilarity measure and linkage criterion produce undesirable dendrograms. In particular, complete linkage is known to perform poorly in the presence of outliers, given its reliance on maximum distances, and single linkage is known to produce a “chaining” effect on the resulting dendrogram, whereby, due to its reliance on minimum distances, observations tend to continuously join increasingly larger, existing clusters rather than being merged with other observations to form new clusters. A negative consequence of this phenomenon is lack of cohesion: observations at opposite ends of the same cluster in a dendrogram could be quite dissimilar. These limitations can

also be attributed to hierarchical clustering—regardless of the linkage employed—optimising a local criterion for each merge, unlike K-means and K-medoids which endeavour to optimise global objectives.

3.3 Choosing the Number of Clusters

Determining the number of clusters in a data set is a fraught task. Throughout Sects. 3.1 and 3.2, method-specific strategies for guiding the choice of K were presented. However, they are not without their limitations. For K-means and K-medoids, the elbow method is somewhat subjective and unreliable. Often, the presence of an elbow is not so clear at a single value of K. Likewise, for agglomerative hierarchical clustering, choosing a height at which to cut the dendrogram as the criterion for choosing K has also been criticised as an overly subjective method. Moreover, these strategies are only capable of identifying the best K value conditional on the chosen method and do not help to identify the overall best solution among multiple competing methods. Moreover, we are required to choose more than just the optimal K value when using K-medoids (for which different solutions with different dissimilarity measures and different K values can be obtained) and agglomerative hierarchical clustering (for which different solutions can be obtained using different dissimilarity measures and linkage criteria).

Overcoming these ambiguities and identifying a more general strategy for comparing the quality of clustering partitions is a difficult task for which many criteria have been proposed. Broadly speaking, cluster quality measures fall into two categories:

1. Comparing of the uncovered partition to a reference clustering (or known grouping labels).
2. Measuring of internal cluster consistency without reference to ground truth labels.

The first is typically conducted using the Rand index [54] or adjusted Rand index [55], which measure the agreement between two sets of partitions. However, as we will be exploring clustering in exploratory, unsupervised settings, using data for which there is no assumed "true" group structure, we will instead focus on a quality measure of the latter kind. As previously stated, a large number of such criteria have been proposed in the literature: several are summarised in Table 7 of Chapter 10 of this book [18], where they are used to guide the choice of K for agglomerative hierarchical clustering in the context of sequence analysis. Here, however, for the sake of brevity, we describe only one commonly used criterion which we later employ in the R tutorial in Sect. 4—which is itself a dissimilarity-based measure and is thus universally applicable to all clustering algorithms we employ—even if in most applications it would be wise to inform the choice of K with several such quantitative criteria. Moreover, the practitioner's own subject matter expertise and

assessment of the interpretability of the obtained clusters should also be used to inform the choice of K.

The quantitative criterion we employ is referred to as the average silhouette width (ASW) criterion [56], which is routinely used to assess the cohesion and separation of the clusters uncovered by dissimilarity-based methods. Cohesion refers to the tendency to group similar objects together and separation refers to the tendency to group dissimilar objects apart in non-overlapping clusters. As the name implies, the ASW is computed as the average of observation-specific silhouette widths. Under the assumption that $K > 1$, silhouette widths and the ASW criterion are calculated as follows:

(A) Let $a(i)$ be the average dissimilarity from observation i to the other members of the same cluster to which observation i is assigned.
(B) Compute the average dissimilarity from observation i to the members of all $K - 1$ other clusters: let $b(i)$ be the minimum such distance computed.
(C) The silhouette for observation i is then defined to be

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} = \begin{cases} 1 - \frac{a(i)}{b(i)} & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & \text{if } a(i) > b(i), \end{cases}$$

unless observation i is assigned to a cluster of size 1, in which case $s(i) = 0$. Notably, $a(i)$ and $b(i)$ need not be calculated using the same dissimilarity measure with which the data were clustered; it is common to adopt the Euclidean distance.
(D) Define the ASW for a given partition $\mathcal{C}$ as: $\text{ASW}(\mathcal{C}) = \frac{1}{n}\sum_{i=1}^{n} s_i$.

Given that $-1 \leq s(i) \leq 1$, the interpretation of $s(i)$ is that a silhouette close to 1 indicates that the observation has been well-clustered, a silhouette close to -1 indicates that the observation would be more appropriately assigned to another cluster, and a silhouette close to zero indicates that the observation lies on the boundary of two natural clusters. If most values of $s(i)$ are high, then the clustering solution can be deemed appropriate. This occurs when $a(i) \ll b(i)$, meaning that observation i must be well-matched its own cluster and poorly-matched to all other clusters. Conversely, if most $s(i)$ values are low or even negative, this provides evidence that K may be too low or too high.

The values of $s(i)$ can be averaged over all observations assigned to the same cluster, as a measure of how tightly grouped the observations in the given cluster are. The ASW criterion itself is simply the mean silhouette width over all observations in the entire data set. Generally, clustering solutions with higher ASW are to be preferred, however it is also prudent to dismiss solutions with many negative silhouettes or particularly low cluster-specific mean silhouettes. A silhouette plot

can help in this regard; such a plot depicts all values of $s(i)$, grouped according to the corresponding cluster and in decreasing order within a cluster. In the R tutorial which follows, ASW values and silhouette plots will guide the choice of an optimal clustering solution in a comparison of multiple methods in Sect. 4.2.4.

4 Tutorial with R

In this section, we will learn how to perform clustering using the R programming language [7], using all methods described throughout Sect. 3. We start by loading the necessary libraries. We will use `cluster` [36] chiefly for functions related to K-medoids and silhouettes. As per other chapters in this book, we use `tidyverse` [57] for data manipulation and `rio` [58] for downloading the data: see Sect. 4.1.1 for details the on the data pre-processing steps employed.

```
library(cluster)
library(rio)
library(tidyverse)
```

We note that hierarchical clustering and K-means are implemented in base R and thus no dedicated libraries need to be loaded.

4.1 The Data Set

Our case study will be to identify different groups of participants that have a similar role in the discussion forum of a massive open online course (MOOC) for teachers. For that purpose, we will rely on the centrality measures of the participants which indicate their number of contributions (`OutDegree`), replies (`InDegree`), position in the network (`Closeness_total`), worth of their connections (`Eigen`), spread of their ideas (`Diffusion_degree`), and more. For more details about the data set, please refer to the data chapter of the book (Chapter 2; [59]). To learn more about centrality measures and how to calculate them, refer to the social network analysis chapter (Chapter 15; [60]). We will henceforth refer to these data as "the MOOC centralities data set". We can download and preview (Table 1) the data with the following commands:

```
URL <-"https://github.com/lamethods/data/raw/main/6_snaMOOC/"
df <- import(paste0(URL, "Centralities.csv"))
df
```

Table 1 Preview of MOOC centralities data set

	name	InDegree	OutDegree	Closeness_total	Betweenness	Eigen	Diffusion.degree	Coreness	Cross_clique_connectivity
1	1	20	33	0.001	1258.143	0.206	1865	18	305
2	2	2	5	0.001	26.524	0.011	218	6	13
3	3	2	4	0.001	30.601	0.009	191	6	11
4	4	2	14	0.001	72.523	0.080	965	13	37
5	5	16	17	0.001	309.033	0.162	1508	18	154
6..444									
445	445	276	56	0.001	16, 690.426	0.699	3574	31	2218

Additionally, a number of categorical variables pertaining to demographic characteristics are available for the same participants. Again, please refer to the data chapter of the book (Chapter 2; [59]) for more details on these variables. With an appropriate distance measure, namely the Gower distance measure, we will incorporate some of these variables in our applications of K-medoids and agglomerative hierarchical clustering (but *not* K-means) in addition to the continuous variables contained in `df`, largely for the purpose of demonstrating clustering methods which are capable of clustering variables of mixed type. We can download and preview the auxiliary categorical data set with the commands below. Note that we select only the following variables:

- `experience` (coded as a level of experience, 1–3),
- `gender` (female or male),
- `region` (Midwest, Northeast, South, and West U.S.A., along with International),

for the sake of simplifying the demonstration of mixed-type variables clustering and reducing the computational burden. We also extract the `UID` column, a user ID which corresponds to the `name` column in `df`, which will be required for later merging these two data sets. We also ensure that all but this leading `UID` column is formatted as a `factor` (Table 2).

```
demog <- import(paste0(URL, "DLT1%20Nodes.csv"))
demog <- demog |>
  select(UID, experience, gender, region) |>
  mutate_at(vars(-UID), as.factor)
demog
```

4.1.1 Pre-processing the Data

In `df`, the first column (`name`) is the student identifier, and the remaining columns are each of the centrality measures calculated from students' interactions in the MOOC forum. We will eventually discard the `name` column from future analyses; we retain it for the time being so that `df` and `demog` can be merged appropriately. The data

Table 2 Preview of the selected additional categorical demographic variables associated with the MOOC centralities data set

	UID	Experience	Gender	Region
1	1	1	Female	South
2	2	1	Female	South
3	3	2	Female	Northeast
4	4	2	Female	South
5	5	3	Female	South
6..444				
445	445	3	Female	South

also contain a small number of observations—only three rows of `df`—with missing values for the variable `Closeness_total`, as seen by

```
df |> is.na() |> which(arr.ind=TRUE)
```

```
    row col
441 441   4
442 442   4
443 443   4
```

Given that none of the clustering methods described in this chapter are capable of accommodating missing data, we remove these three observations for future analyses too. Furthermore, one of the rows in `demog` has `NULL` recorded for the `gender` variable. We remove all invalid rows from both `df` and `demog`. The function `complete.cases()` constructs a completely observed data set by extracting the rows which contain one or more missing values and we augment the index of fully observed rows with an index of non-`NULL` `gender` values. Finally, we drop the redundant `NULL` level from the `factor` variable `gender`.

```
obs_ind <- complete.cases(df) & demog$gender != "NULL"
df$name[!obs_ind] # indices of observations with missing values
```

```
[1] 439 441 442 443
```

```
df <- df |> filter(obs_ind)
demog <- demog |> filter(obs_ind) |> droplevels()
```

Before proceeding any further, it would be prudent to explore the complete data visually, which we do via the matrix of pairwise scatter plots, excluding the `name` column, in Fig. 1:

```
pairs(df[,-1])
```

From the plots in Fig. 1, we can see that there are clearly two quite extreme outliers. Simple exploratory analyses (not shown) confirms that these are the final two rows of the complete data set. These observations are known to correspond to the two instructors who led the discussion. They have been marked using a red cross symbol in Fig. 1. Though we have argued that K-medoids is a more robust clustering method than K-means, for example, we also remove these observations in order to avoid their detrimental effects on the K-means output. The rows must be removed from both `df` and `demog` so that they can later be merged. With these observations included, K-means for instance would likely add one additional cluster containing just these two observations, about whom we already know their role differs substantially from the other observations, as they are quite far from the bulk of the data in terms of squared Euclidean distance. That is not to say, however, that there will not still be outliers in `df` after removing these two cases.

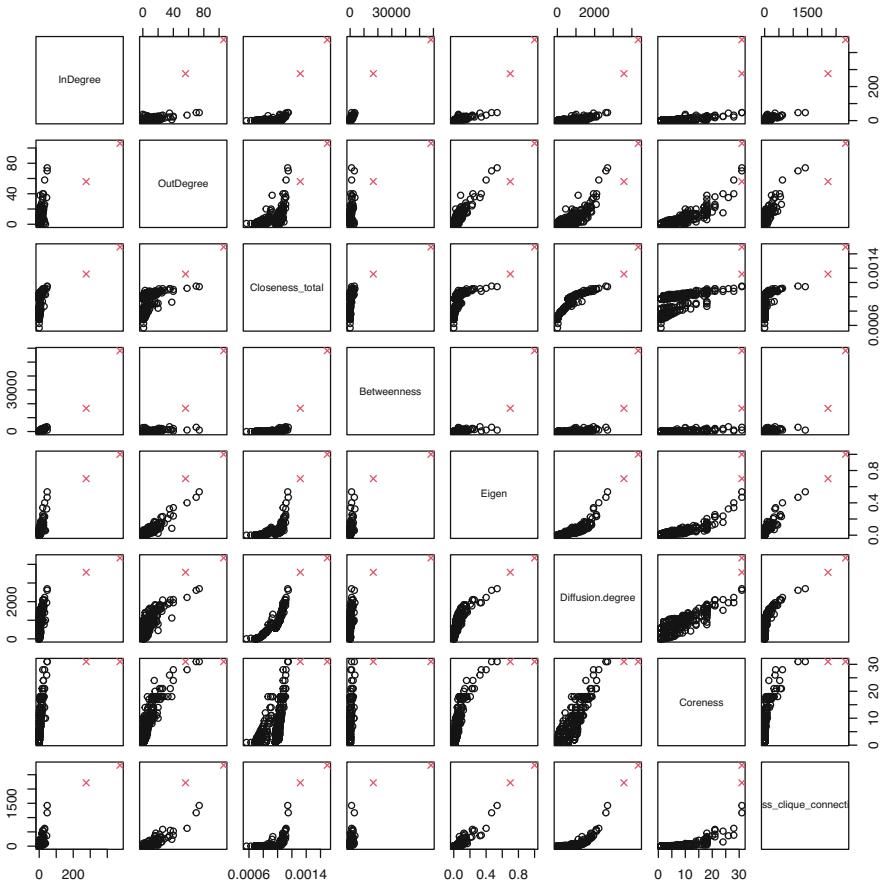


Fig. 1 Matrix of pairwise scatter plots for all variables in the MOOC centralities data set

```
keep_ind <- 1:(nrow(df) - 2)

df <- df |> slice(keep_ind)
demog <- demog |> slice(keep_ind)
```

As is good practice when using dissimilarity-based clustering algorithms, we pre-process the purely continuous data by normalising each variable to have a mean of 0 and a variance of 1, by constructing the scaled data frame `sdf` using the function `scale()`, again excluding the `name` column.

```
sdf <- scale(df[,-1], center=TRUE, scale=TRUE)
```

The object `sdf` can be used for all clustering methods described herein. To also accommodate the categorical demographic variables, we use `merge()` to

combine both the scaled continuous data and the categorical data. This requires some manipulation of the `name` column, with which the two data sets are merged, but we ultimately discard the superfluous `name` column, which we do not want to contribute to any pairwise distance matrices or clustering solutions, from both `merged_df` and `sdf`.

```
merged_df <- data.frame(name=df$name, sdf) |>
  merge(demog, by=1) |>
  select(-name)
```

Finally, before proceeding to apply various clustering methods to these data, we present summaries of the counts of each level of the categorical variables `experience`, `gender`, and `region`. These variables imply groupings of size three, two, and five, respectively, and it will be interesting to see if this is borne out in any of the mixed-type clustering applications.

```
table(merged_df$experience)
```

```
  1   2   3
118 150 171
```

```
table(merged_df$gender)
```

```
female   male
   299    140
```

```
table(merged_df$region)
```

```
International       Midwest     Northeast         South          West
           32            77           110           168            52
```

4.2 *Clustering Applications*

We now show how each of the clustering methods described above can be implemented in R, using these data throughout and taking care to address all practical concerns previously raised. For each method—K-means in Sect. 4.2.1, K-medoids in Sect. 4.2.2, and agglomerative hierarchical clustering in Sect. 4.2.3—we show clustering results following the method-specific guidelines for choosing K. However, we conclude by comparing results across different methods using the average silhouette width criterion to further guide the choice of K in Sect. 4.2.4. We refrain from providing an interpretation of the uncovered clusters until Sect. 4.2.5, after the optimal clustering solution is identified.

Before we proceed, we set the seed to ensure that results relying on random number generation are reproducible.

```
set.seed(2024)
```

4.2.1 *K*-means Application

We begin by showing a naive use of the `kmeans()` function, supplying only the scaled data `sdf` and the pre-specified number of clusters K via the `centers` argument. For now, we assume for no particular reason that there are $K = 3$ clusters, just to demonstrate the use of the `kmeans()` function and its arguments. A number of aspects of the results are printed when we examine the resulting `km1` object.

```
km1 <- kmeans(sdf, centers=3)
km1
```

```
K-means clustering with 3 clusters of sizes 48, 129, 262

Cluster means:
    InDegree  OutDegree Closeness_total Betweenness       Eigen
1  2.2941694  1.9432559       1.0923551   1.9697769  2.00747791
2 -0.4088814 -0.4311073      -1.4086380  -0.3975783 -0.55451137
3 -0.2189864 -0.1437536       0.4934399  -0.1651209 -0.09475944
  Diffusion.degree   Coreness Cross_clique_connectivity
1        1.9078646  2.1923592                 2.0066265
2       -1.1038258 -0.5622732                -0.3094092
3        0.1939543 -0.1248092                -0.2152835

Clustering vector:
  [1] 1 2 2 3 1 1 1 1 3 1 1 3 1 1 1 2 1 3 1 2 2 1 2 1 3 1 1 2 1 1 2 3 3 1 1 1 2
 [38] 3 3 2 1 3 2 1 3 3 2 3 1 1 3 3 1 1 2 3 3 1 3 1 1 1 1 1 3 3 1 1 3 3 3 3 2 3
 [75] 3 3 3 3 2 2 3 3 3 2 3 2 3 1 2 3 3 1 2 3 3 3 2 1 3 1 3 3 3 3 3 2 3 2 3 3 2
[112] 3 3 3 1 1 2 2 2 3 3 2 2 3 2 3 3 3 3 2 3 2 3 3 2 3 1 3 3 2 3 3 3 3 2 3 3 3
[149] 2 3 2 3 2 3 3 2 3 3 2 3 3 3 3 2 3 3 3 2 3 3 3 3 3 2 2 3 3 3 3 2 3 2 3 3 3
[186] 3 3 3 3 3 2 3 3 3 3 3 2 1 3 3 3 3 3 3 3 3 3 2 2 2 3 3 2 3 3 2 2 3 3 1 2 3 2
[223] 1 2 2 3 2 2 2 2 2 3 2 1 3 3 2 2 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3
[260] 2 3 2 2 2 3 2 3 2 2 2 3 2 2 2 2 3 2 3 3 3 3 2 2 3 2 2 2 2 2 2 2 3 2 2 3 2
[297] 3 3 3 3 3 3 3 3 3 2 2 3 2 1 2 2 2 3 2 2 3 3 2 3 2 3 2 3 3 3 2 2 3 3 3 2 3
[334] 2 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 3 2 2 2 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 2
[371] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[408] 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 1 2 2 2 2 2 2 3

Within cluster sum of squares by cluster:
[1] 858.90295  72.01655 414.30502
 (between_SS / total_SS =  61.6 %)
```

```
Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
    "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"
```

Among other things, this output shows the estimated centroid vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ upon convergence of the algorithm (`$centers`), an indicator vector showing the assignment of each observation to one of the $K = 3$ clusters (`$cluster`), the size of each cluster in terms of the number of allocated observations (`$size`), the within-cluster sum-of-squares *per cluster* (`$withinss`), and the ratio of the between-cluster sum-of-squares (`$betweenss`) to the total sum of squares (`$totss`). Ideally, this ratio should be large, if the total within-cluster sum-of-squares is minimised. We can access the TWCSS quantity by typing

```
km1$tot.withinss
```

```
[1] 1345.225
```

However, these results were obtained using the default values of ten maximum iterations (`iter.max=10`, by default) and only one random set of initial centroid vectors (`nstart=1`, by default). To increase the likelihood of converging to the global minimum, it is prudent to increase `iter.max` and `nstart`, to avoid having the algorithm terminate prematurely and avoid converging to a local minimum, as discussed in Sect. 3.1.2.2. We use `nstart=50`, which is reasonably high but not so high as to incur too large a computational burden.

```
km2 <- kmeans(sdf, centers=3, nstart=50, iter.max=100)
km2$tot.withinss
```

```
[1] 1339.226
```

We can see that the solution associated with the best random starting values achieves a lower TWCSS than the earlier naive attempt. Next, we see if an even lower value can be obtained using the K-means^{++} initialisation strategy, by invoking the `kmeans_pp()` function from Sect. 3.1.2.2 with `K=3` centers and supplying these centroid vectors directly to `kmeans()`, rather than indicating the number of random `centers` to use.

```
km3 <- kmeans(sdf, centers=kmeans_pp(sdf, K=3), iter.max=100)
km3$tot.withinss
```

```
[1] 1343.734
```

In this case, using the K-means^{++} initialisation strategy did not further reduce the TWCSS; in fact it is worse than the solution obtained using `nstart=50`. However, recall that K-means^{++} is itself subject to randomness and should therefore also be run several times, though the number of K-means^{++} runs need not be so high as 50. In the code below, we use `replicate()` to invoke both `kmeans_pp()` and `kmeans()` itself ten times in order to obtain a better solution.

```
KMPP <- replicate(10, list(kmeans(sdf, iter.max=100,
                                  centers=kmeans_pp(sdf, K=3))))
```

Among these ten solutions, five are identical and achieve the same minimum TWCSS value.

```
TWCSS <- sapply(KMPP, "[[", "tot.withinss")
TWCSS
```

```
[1] 1345.225 1339.226 1339.226 1339.226 1340.457 1339.226 1345.225
    1345.225
[9] 1345.225 1339.226
```

Thereafter, we can extract a solution which minimises the `tot.withinss` as follows:

```
km4 <- KMPP[[which.min(TWCSS)]]
km4$tot.withinss
```

```
[1] 1339.226
```

Finally, this approach resulted in an identical solution to `km2` being obtained—with just ten runs of K-means^{++} and K-means rather than `nstart=50` runs of the K-means algorithm alone—which is indeed superior to the solution obtained with just one uninformed random start in `km1`. We will thus henceforth adopt this initialisation strategy always.

To date, the K-means algorithm has only been ran with the fixed number of clusters $K = 3$, which may be suboptimal. The following code iterates over a range of K values, storing both the `kmeans()` output itself and the TWCSS value for each K. The range $K = 1, \ldots, 10$ notably includes $K = 1$, corresponding to no group structure in the data. The reason for storing the `kmeans()` output itself is to avoid having to run `kmeans()` again after determining the optimal K value; such a subsequent run may not converge to the same minimised TWCSS and having to run the algorithm again would be computationally wasteful.

```
K <- 10 # set upper limit on range of K values

TWCSS <- numeric(K) # allocate space for TWCSS estimates

KM <- list() # allocate space for kmeans() output
```

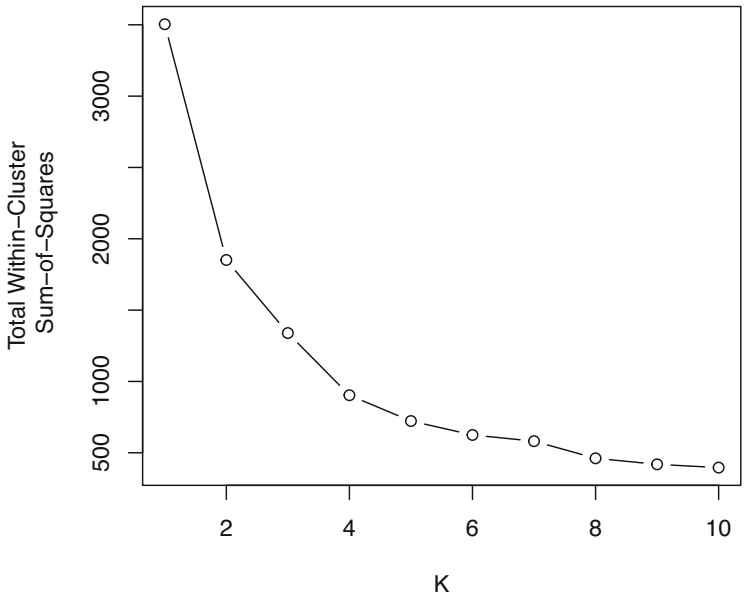


Fig. 2 Elbow plot showing a range of K values against the corresponding obtained TWCSS for K-means applied to the MOOC centralities data set using K-means with K-means^{++} initialisation

```
for(k in 1:K) { # loop over k=1,...,K

  # Run K-means using K-Means++ initialisation:
  # use the current k value and do so ten times if k > 1
  KMPP    <- replicate(ifelse(k > 1, 10, 1),
                       list(kmeans(sdf, iter.max=100,
                                   centers=kmeans_pp(sdf, K=k))))

  # Extract and store the solution which minimises the TWCSS
  KM[[k]] <- KMPP[[which.min(sapply(KMPP, "[[", "tot.withinss"))]]

  # Extract the TWCSS value for the current value of k
  TWCSS[k] <- KM[[k]]$tot.withinss
}
```

As previously stated in Sect. 3.1.2.1, the so-called "elbow method" consists of plotting the range of K values on the x-axis against the corresponding obtained TWCSS values on the y-axis and looking for an kink in the resulting curve.

```
plot(x=1:K, y=TWCSS, type="b",
     xlab="K", ylab="Total Within-Cluster\n Sum-of-Squares")
```

Figure 2 suggests that $K = 4$ would produce the best results: beyond $K = 4$, the decrease in TWCSS is minimal, which suggests that an extra cluster is not required

to model the data well; between $K = 3$ and $K = 4$, there is a more substantial decrease in TWCSS, which suggests that the fourth cluster is necessary. This method is of course highly subjective, and we will further inform our choice of K for K-means using silhouette widths and the ASW criterion in Sect. 4.2.4.

For now, we can interrogate the K-means solution with $K = 4$ by examining the sizes of the clusters and their centroid vectors, using the corresponding fourth element of the list `KM` by setting `K <- 4`.

```
K <- 4

KM[[K]]$size
```

```
[1] 127  57   8 247
```

```
KM[[K]]$centers
```

```
    InDegree  OutDegree Closeness_total Betweenness      Eigen Diffusion.degree
1 -0.4086612 -0.4381057      -1.4220978  -0.3994123 -0.5596480       -1.1138889
2  1.5706416  1.1372352       0.9247804   1.3859592  1.0781087        1.4216179
3  4.1035975  5.0512503       1.5201941   3.4673658  5.4946785        3.1631065
4 -0.2852445 -0.2007813       0.4685522  -0.2267743 -0.1390054        0.1422138
    Coreness Cross_clique_connectivity
1 -0.5672245                -0.3102236
2  1.7423739                 0.9615041
3  3.4821417                 5.5033583
4 -0.2232184                -0.2406243
```

However, these centroids correspond to the *scaled* version of the data created in Sect. 4.1.1. Interpretation can be made more straightforward by undoing the scaling transformation on these centroids, so that they are on the same scale as the data `df` rather than the scaled data `sdf` which was used as input to `kmeans()`. The column-wise means and standard deviations used when `scale()` was invoked are available as the attributes `"scaled:center"` and `"scaled:scale"`, respectively and are used in the code below. We show these centroids rounded to four decimal places in Table 3.

```
rescaled_centroids <- t(apply(KM[[K]]$centers, 1, function(r) {
                        r * attr(sdf, "scaled:scale") + attr(sdf,
                        "scaled:center")
                        } ))
round(rescaled_centroids, 4)
```

Now, we can more clearly see that the first cluster, with $n_{k=1} = 127$ observations, has the lowest mean value for all $d = 8$ centrality measures, while the last cluster, the largest with $n_{k=4} = 247$ observations, has moderately larger values for all centrality measures. The two smaller clusters, cluster two with $n_{k=2} = 57$ observations and cluster three with only $n_{k=3} = 8$ observations have the second-largest and largest values for each measure, respectively. As previously stated, we

Table 3 Centroids from the $K = 4$ K-means solution on the original data scale

InDegree	OutDegree	Closeness_ total	Betweenness	Eigen	Diffusion. degree	Coreness	Cross_clique_ connectivity
1.1024	1.7480	0.0008	20.7010	0.0071	144.1732	2.6378	4.6850
15.3684	14.8421	0.0010	849.9328	0.0996	1370.1053	16.1053	157.5789
33.6250	47.3750	0.0011	1816.6608	0.3492	2212.1250	26.2500	703.6250
1.9919	3.7206	0.0010	100.8842	0.0308	751.5061	4.6437	13.0526

defer a more-detailed interpretation of uncovered clusters to Sect. 4.2.5, after the optimal clustering solution has been identified.

4.2.2 *K*-medoids Application

The function `pam()` in the `cluster` library implements the PAM algorithm for K-medoids clustering. By default, this function requires only the arguments `x` (a pre-computed pairwise dissimilarity matrix, as can be created from the functions `dist()`, `daisy()`, and more) and `k`, the number of clusters. However, there are many options for many additional speed improvements and initialisation strategies [48]. Here, we invoke a faster variant of the PAM algorithm which necessitates specification of `nstart` as a number greater than one, to ensure the algorithm is evaluated with multiple random initial medoid vectors, in a similar fashion to `kmeans()`. Thus, we call `pam()` with `variant="faster"` and `nstart=50` throughout.

Firstly though, we need to construct the pairwise dissimilarity matrices to be used as input to the `pam()` function. Unlike K-means, we are not limited to *squared* Euclidean distances. It is prudent, therefore, to explore K-medoids solutions with several different dissimilarity measures and compare the different solutions obtained for different measures. Each distance measure will yield different results at the same K value, thus the value of K and the distance measure must be considered as a pair when determining the optimal solution.

We begin by calculating the Euclidean, Manhattan, and Minkowski (with $p = 3$) distances on the scaled continuous data in `sdf`:

```
dist_euclidean <- dist(sdf, method="euclidean")

dist_manhattan <- dist(sdf, method="manhattan")

dist_minkowski3 <- dist(sdf, method="minkowski", p=3)
```

Secondly, we avail of another principal advantage of K-medoids; namely, the ability to incorporate categorical variables in mixed-type data sets, by calculating pairwise Gower distances between each row of `merged_df`, using `daisy()`.

```
dist_gower <- daisy(merged_df, metric="gower")
```

As per the K-means tutorial in Sect. 4.2.1, we can produce an elbow plot by running the algorithm over a range of K values and extracting the minimised within-cluster total distance achieved upon convergence for each value of K. We demonstrate this for the `dist_euclidean` input below.

```
K <- 10
WCTD_euclidean <- numeric(K)
pam_euclidean <- list()
for(k in 1:K) {
  pam_euclidean[[k]] <- pam(dist_euclidean, k=k,
                            variant="faster", nstart=50)
  WCTD_euclidean[k] <- pam_euclidean[[k]]$objective[2]
}
```

Equivalent code chunks for the `dist_manhattan`, `dist_minkowski3`, and `dist_gower` inputs are almost identical, so we omit them here for the sake of brevity. Suffice to say, equivalent lists `pam_manhattan`, `pam_minkowski3`, and `pam_gower`, as well as equivalent vectors `WCTD_manhattan`, `WCTD_minkowski3`, and `WCTD_gower`, can all be obtained. Using these objects, corresponding elbow plots for all four dissimilarity measures are showcased in Fig. 3.

Some of the elbow plots in Fig. 3 are more conclusive than others. As examples, there are reasonably clear elbows at $K = 3$ for the Euclidean and Minkowski distances, arguably an elbow at $K = 4$ for the Manhattan distance, and no clear, unambiguous elbow under the Gower distance. In any case, the elbow method only helps to identify the optimal K value for a given dissimilarity measure; we defer a discussion of how to choose the overall best K-medoids clustering solution to later in this tutorial.

For now, let's interrogate the $K = 3$ solution obtained using the Euclidean distance. Recall that the results are already stored in the list `pam_euclidean`, so we merely need to access the third component of that list by setting `K <- 3`. Firstly, we can examine the size of each cluster by tabulating the cluster-membership indicator vector as follows:

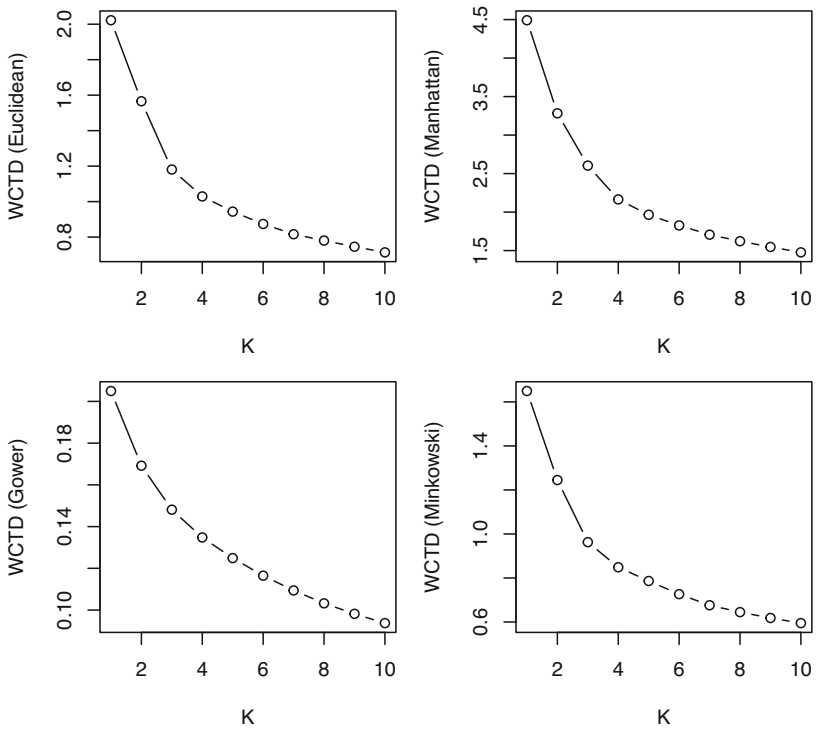


Fig. 3 Elbow plots for K-medoids clustering evaluated with different dissimilarity measures over a range of K values. In clockwise order, beginning with the top-left panel, these are the Euclidean distance, Manhattan distance, Minkowski distance and, for the merged data with additional categorical covariates, the Gower distance

```
K <- 3

table(pam_euclidean[[K]]$clustering)
```

```
  1   2   3
 67 122 250
```

Examining the medoids which serve as prototypes of each cluster is rendered difficult by virtue of the input having been a distance matrix rather than the data set itself. Though K-medoids defines the medoids to be the rows in the data set from which the distance to all other observations currently allocated to the same cluster, according to the specified distance measure, is minimised, the `medoids` component of the output instead gives the *indices* of the medoids within the data set.

```
pam_euclidean[[K]]$medoids
```

```
[1]  88 272  45
```

Table 4 Medoids for the $K = 3$ K-medoids solution obtained using the Euclidean distance on the original data scale, with the corresponding observation index in the `name` column

name	InDegree	OutDegree	Closeness_total	Betweenness	Eigen	Diffusion.degree	Coreness	Cross_clique_connectivity
88	16	14	0.0011	675.5726	0.1096	1415	18	157
272	1	1	0.0007	0.0000	0.0047	41	2	2
45	1	2	0.0010	29.4645	0.0194	684	3	5

Table 5 Cross-tabulation of the clusters obtained by K-means with $K = 4$ (along the columns) and K-medoids with $K = 3$ and the Euclidean distance (along the rows)

	1	2	3	4
1	0	57	8	2
2	122	0	0	0
3	5	0	0	245

Fortunately, these indices within `sdf` correspond to the same, unscaled observations within `df`. Allowing for the fact that observations with `name` greater than the largest index here were removed due to missingness, they are effectively the values of the `name` column corresponding to the medoids. Thus, we can easily examine the medoids on their original scale. In Table 4, they include the `name` column, which was *not* used when calculating the pairwise distance matrices, and are rounded to four decimal places.

```
df |>
  slice(as.numeric(pam_euclidean[[K]]$medoids)) |>
  round(4)
```

Thus, we can see that there is a small cluster with $n_{k=1} = 67$ observations which has the largest values for all $d = 8$ centrality measures, a slightly larger cluster with $n_{k=2} = 122$ observations and the lowest values for all variables, and the largest cluster with $n_{k=3} = 250$ and intermediate values for all variables. The cluster sizes of the K-medoids solution being more evenly balanced than the earlier K-means solution is an artefact of K-medoids being less susceptible to outlying observations by virtue of not squaring the distances. We can explore this by cross-tabulating the clusters obtained by K-means with $K = 4$ and K-medoids with $K = 3$ and the Euclidean distance in Table 5.

```
table(pam_euclidean[[3]]$clustering,
      KM[[4]]$cluster)
```

From the cross-tabulation in Table 5, we can see that the $n_k = 8$ observations in the smallest K-means cluster were absorbed into a larger cluster under the K-medoids solution, thereby demonstrating the robustness of K-medoids to outliers. Otherwise, both solutions are broadly in agreement.

4.2.3 Agglomerative Hierarchical Clustering Application

Performing agglomerative hierarchical clustering is straightforward now that the distance matrices have already been created for the purposes of running `pam()`. All that is required is to specify the distance matrix and an appropriate linkage criterion as the `method` when calling `hclust()`. We demonstrate this below for a subset of all possible distance measure and linkage criterion combinations among those described in Sect. 3.2.1. Recall that for the Ward criterion, the underlying distance measure is usually assumed to be squared Euclidean and that `"ward.D2"` is the correct `method` to use when the Euclidean distances are not already squared. For `method="centroid"`, we manually square the Euclidean distances.

```
hc_minkowski3_complete <- hclust(dist_minkowski3, method="complete")

hc_manhattan_single <- hclust(dist_manhattan, method="single")

hc_gower_average <- hclust(dist_gower, method="average")

hc_euclidean_ward <- hclust(dist_euclidean, method="ward.D2")

hc_euclidean2_centroid <- hclust(dist_euclidean^2, method="ward.D2")
```

Plotting the resulting dendrograms is also straightforward. Simply calling `plot()` on any of the items above will produce a dendrogram visualisation. We do so here for four of the hierarchical clustering solutions constructed above—the undepicted `hc_euclidean2_centroid` dendrogram is virtually indistinguishable from that of `hc_euclidean_ward`—while suppressing the observation indices along the x-axis for clarity by specifying `labels=FALSE`.

```
plot(hc_minkowski3_complete, labels=FALSE,
     main="", xlab="Minkwoski Distance (p=3) with Complete Linkage")
plot(hc_manhattan_single, labels=FALSE,
     main="", xlab="Manhattan Distance with Single Linkage")
plot(hc_gower_average, labels=FALSE,
     main="", xlab="Gower Distance with Average Linkage")
plot(hc_euclidean_ward, labels=FALSE,
     main="", xlab="Euclidean Distance with the Ward Criterion")
```

As previously alluded to in Sect. 3.2.2, some combinations of dissimilarity measure and linkage criterion are liable to produce undesirable results. The susceptibility to outliers of complete linkage clustering is visible in the top-left panel of Fig. 4, where just two observations form a cluster at a low height and are never again merged. The tendency of single linkage clustering to exhibit a "chaining" effect whereby all observations are successively merged into just one ever-larger cluster is evident in the top-right panel of Fig. 4, and similar behaviour is observed for the Gower distance with average linkage depicted in the bottom-left panel. The

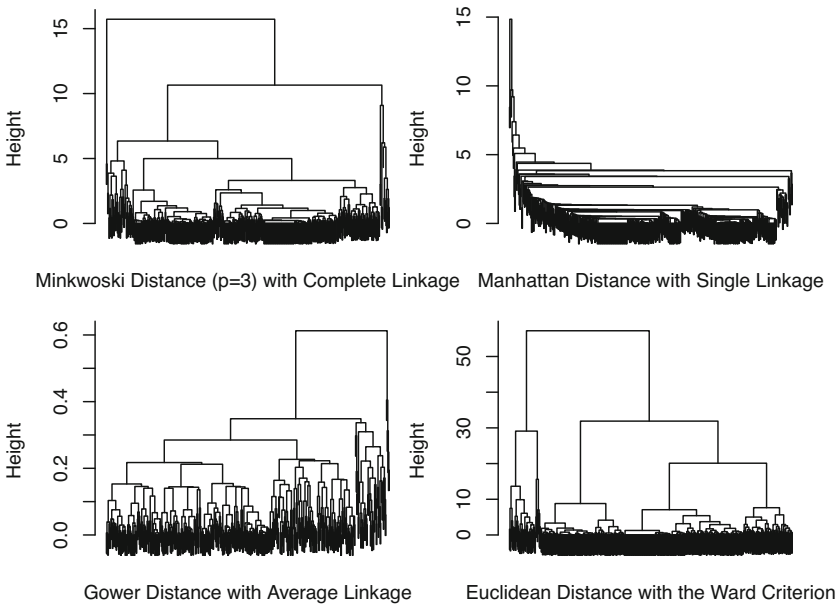


Fig. 4 Dendrograms obtained by agglomerative hierarchical clustering with a selection of dissimilarity measures and linkage criteria

most reasonable results appear to arise from using the Ward criterion in conjunction with Euclidean distances.

Taking the set of candidate partitions in `hc_euclidean_ward`, for the reasons outlined above, all that remains is to cut this dendrogram at an appropriate height. Practitioners have the freedom to explore different levels of granularity in the final partition. Figure 5 shows the dendrogram from the bottom-right panel of Fig. 4 cut horizontally at different heights, indicated by lines of different colours, as well as the corresponding implied K values.

Thereafter, one simply extracts the resulting partition by invoking `cutree()` with the appropriate height `h`. For example, to extract the clustering with $K = 3$, which we choose here because of the wide range of heights at which a $K = 3$ solution could be obtained:

```
hc_ward2 <- cutree(hc_euclidean_ward, h=45)
```

The object `hc_ward2` is now simply an vector indicating the cluster-membership of each observation in the data set. We show only the first few, for brevity, and then tabulate this vector to compute the cluster sizes. However, interpretation of these clusters is more difficult than in the case of K-means and K-medoids, as there is no centroid or medoid prototype with which to characterise each cluster.

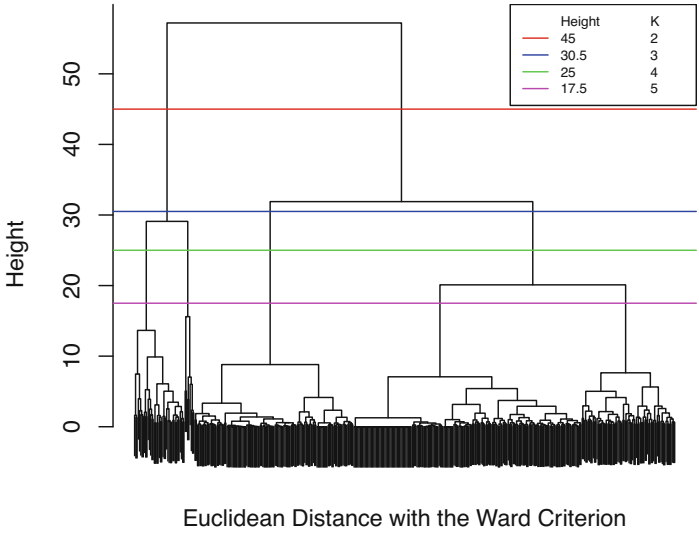


Fig. 5 Dendrogram obtained using the Euclidean distance and Ward criterion cut at different heights with the corresponding implied K indicated

```
head(hc_ward2)
```

```
[1] 1 2 2 2 1 1
```

```
table(hc_ward2)
```

```
hc_ward2
  1   2
 49 390
```

In this section, we have not presented an exhaustive evaluation of all possible pairs of dissimilarity measure and linkage criterion, but note that the code to do so is trivial. In any case, much like K-means and K-medoids, we must turn to other cluster quality indices to guide the choice of the best overall solution, be that choosing the best distance and linkage settings for agglomerative hierarchical clustering, or choosing the best clustering method in general among several competing methods. We now turn to finding the optimal clustering solution among multiple competing methods, guided by silhouette widths and plots thereof.

4.2.4 Identifying the Optimal Clustering Solution

In our application of K-means, the elbow method appeared to suggest that $K = 4$ yielded the best solution. In our applications of K-medoids, the elbow method

suggested different values of K for different distance metrics. Finally, in our applications of hierarchical clustering, we noted that visualising the resulting dendrogram could be used to guide the choice of the height at which to cut to produce a single hard partition of K clusters. Now, we must determine which method yields the overall best solution. Following Sect. 3.3, we employ silhouettes for this purpose.

For K-means and agglomerative hierarchical clustering, silhouettes can be computed using the `silhouette` function in the `cluster` library, in which case the function requires two arguments; the integer vector containing the cluster membership labels and an appropriate dissimilarity matrix. Thus, for instance, silhouettes could be obtained easily for the following two examples (using `kmeans()` and `hclust()`, respectively).

```
kmeans_sil <- silhouette(kmeans(sdf, centers=kmeans_pp(sdf, K=4),
                                iter.max=100)$cluster,
                         dist_euclidean)


hclust_sil <- silhouette(cutree(hclust(dist_euclidean, method="ward.D2"), k=2),
                         dist_euclidean)
```

For K-medoids, it suffices only to supply the output of `pam()` itself, from which the cluster membership labels and appropriate dissimilarity matrix will be extracted internally, e.g.,

```
pam_sil <- silhouette(pam(dist_euclidean, k=3, variant="faster", nstart=50))
```

Thereafter, `plot()` can be called on `kmeans_sil`, `hclust_sil`, or `pam_sil` to produce a silhouette plot. For an example based on `hclust_sil`, see Fig. 6. Note that as $K = 2$ here, `col=2:3` colours the silhouettes according to their cluster.

```
plot(hclust_sil, main="", col=2:3)
```

Figure 6 shows that most silhouette widths are positive under this solution, indicating that most observations have been reasonably well-clustered. Cluster 2 appears to be the most cohesive, with a cluster-specific average silhouette width of 0.70, while cluster 1 appears to be the least cohesive, with a corresponding average of just 0.24. The overall ASW is 0.65, as indicated at the foot of the plot.

Given that we adopted a range of $K = 1, \ldots, 10$ when using K-means and K-medoids, and four different dissimilarity measures when using K-medoids, we have 50 non-hierarchical candidate solutions to evaluate, of which some seem more plausible than others according to the respective elbow plots. For agglomerative hierarchical clustering, an exhaustive comparison over a range of K values, for each dissimilarity measure and each linkage criterion, would be far too extensive for the present tutorial. Consequently, we limit our evaluation of silhouettes to the K-means and K-medoids solutions already present

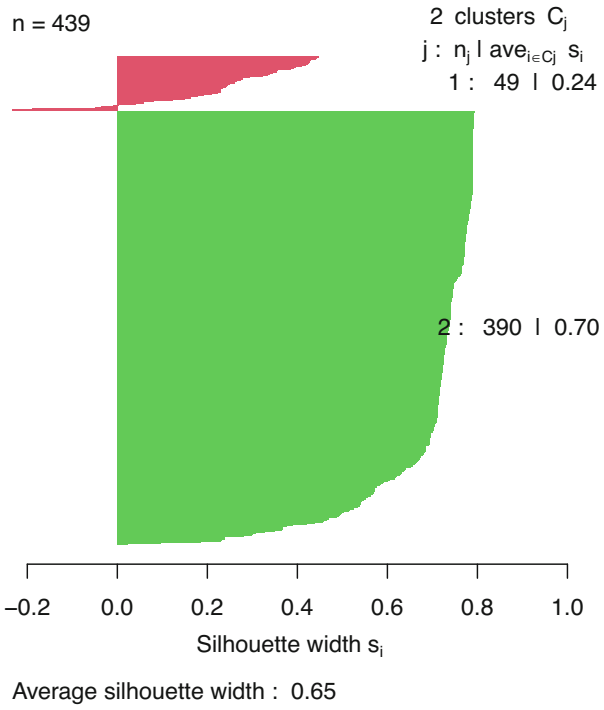


Fig. 6 Silhouette plot for the $K = 2$ hierarchical clustering solution obtained using the Ward criterion with Euclidean distances

in the objects `KM`, `pam_euclidean`, `pam_manhattan`, `pam_minkowski3`, and `pam_gower`, and the hierarchical clustering solutions which employ the Ward criterion in conjunction with Euclidean distances (of which we consider a further 10 solutions, again with $K = 1, \ldots, 10$, by considering the 10 possible associated heights at which the dendrogram can be cut). We limit the hierarchical clustering solutions to those based on the Ward criterion given that single linkage and complete linkage have been shown to be susceptible to chaining and sensitive to outliers, respectively. We use the corresponding pre-computed dissimilarity matrices `dist_euclidean`, `dist_manhattan`, `dist_minkowski3`, and `dist_gower`, where appropriate throughout.

Though the ASW associated with `hclust_sil` is given on the associated silhouette plot in Fig. 6, we can calculate ASW values for other solutions—which we must do to determine the best solution—without producing individual silhouette plots. To show how this can be done, we examine the structure of the `hclust_sil` object, showing only its first few rows.

```
head(hclust_sil)
```

```
     cluster neighbor sil_width
[1,]       1        2 0.4408007
[2,]       2        1 0.7416708
[3,]       2        1 0.7401549
[4,]       2        1 0.4696606
[5,]       1        2 0.2494557
[6,]       1        2 0.1915306
```

The columns relate to the cluster to which object i is assigned, the cluster for which the corresponding $b(i)$ was minimised, and the $s(i)$ score itself, respectively. Calculating `mean(hclust_sil[,3])` will return the ASW. Though the code is somewhat tedious, we calculate the ASW criterion values for all 60 candidate solutions—that is, six methods evaluated over $K = 1, \ldots, 10$—using a small helper function to calculate the ASW for the sake of tidying the code.

```
K <- 10
ASW <- function(x) mean(x[,3])
silhouettes <- data.frame(K=2:K,
  kmeans=sapply(2:K, function(k) ASW(silhouette(KM[[k]]$
                                    cluster, dist_euclidean))),
  kmedoids_euclidean=sapply(2:K, function(k) ASW(silhouette
                           (pam_euclidean[[k]]))),
  kmedoids_manhattan=sapply(2:K, function(k) ASW(silhouette
                           (pam_manhattan[[k]]))),
  kmedoids_minkowski3=sapply(2:K, function(k) ASW(silhouette
                            (pam_minkowski3[[k]]))),
  kmedoids_gower=sapply(2:K, function(k) ASW(silhouette
                       (pam_gower[[k]]))),
  hc_euclidean_ward=sapply(2:K, function(k) ASW(silhouette(cutree
                           (hc_euclidean_ward, k), dist_euclidean))))
```

In Fig. 7, we plot these silhouettes against K using `matplot()`, omitting the code to do so for brevity.

According to Fig. 7, there is generally little support for $K > 5$ across almost all methods considered, as most method's ASW values begin to decline after this point. The ASW values also make clear that incorporating the additional categorical demographic variables in a mixed-type clustering using the Gower distance does not lead to reasonable partitions, for any number of clusters K. Overall, the most promising solutions in terms of having the highest ASW are K-means, Ward hierarchical clustering, and K-medoids with the Manhattan distance, all with $K = 2$, and K-medoids with the Euclidean and Minkowski distances, each with $K = 3$. However, it would be wise to examine silhouette widths in more detail, rather than relying merely on the average. The silhouettes for this hierarchical clustering solution are already depicted in Fig. 6, so Fig. 8 shows individual silhouette widths for the remaining solutions.

It is notable that the silhouettes and ASW of the $K = 2$ K-means solution (top-left panel of Fig. 8) and the Ward hierarchical clustering solution (Fig. 6) appear almost identical (if one accounts for the clusters being relabelled and associated

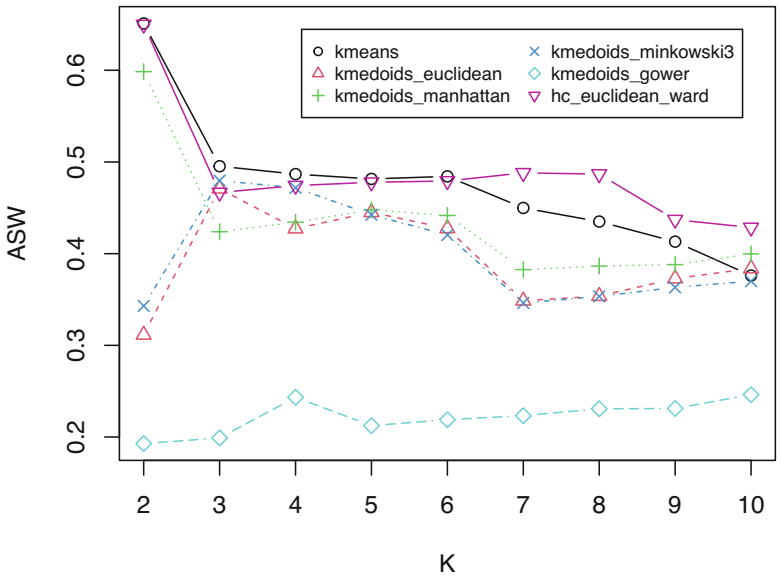


Fig. 7 ASW criterion values plotted against K for K-means, K-medoids (with the Euclidean, Manhattan, Minkowski ($p = 3$), and Gower distances), and agglomerative hierarchical clustering based on Euclidean distance and the Ward criterion

colours being swapped). Indeed, according to a cross-tabulation of their partitions (not shown), their assignments differ for just 4 out of $n = 439$ observations. Despite having the highest ASW, we can justify dismissing these solutions given that $K = 2$ was not well-supported by its corresponding elbow plot in Fig. 2. Similar logic suggests that the $K = 3$ K-means solution and the $K = 2$ K-medoids solution based on the Manhattan distance can also be disregarded. Although we stress again that an ideal analysis would more thoroughly determine an optimal solution with reference to additional cluster quality measures and note that various clustering solutions can be legitimate, for potentially different clustering aims, on the same data set [2, 3], we can judge that—among the two $K = 3$ K-medoids solutions—the one based on the Euclidean distance is arguably preferable, for two reasons. Firstly, its ASW is quite close to that of the Minkowski distance solution:

```
silhouettes |>
  filter(K == 3) |>
  select(kmedoids_euclidean, kmedoids_minkowski3)
```

```
  kmedoids_euclidean kmedoids_minkowski3
1           0.470844           0.4795972
```

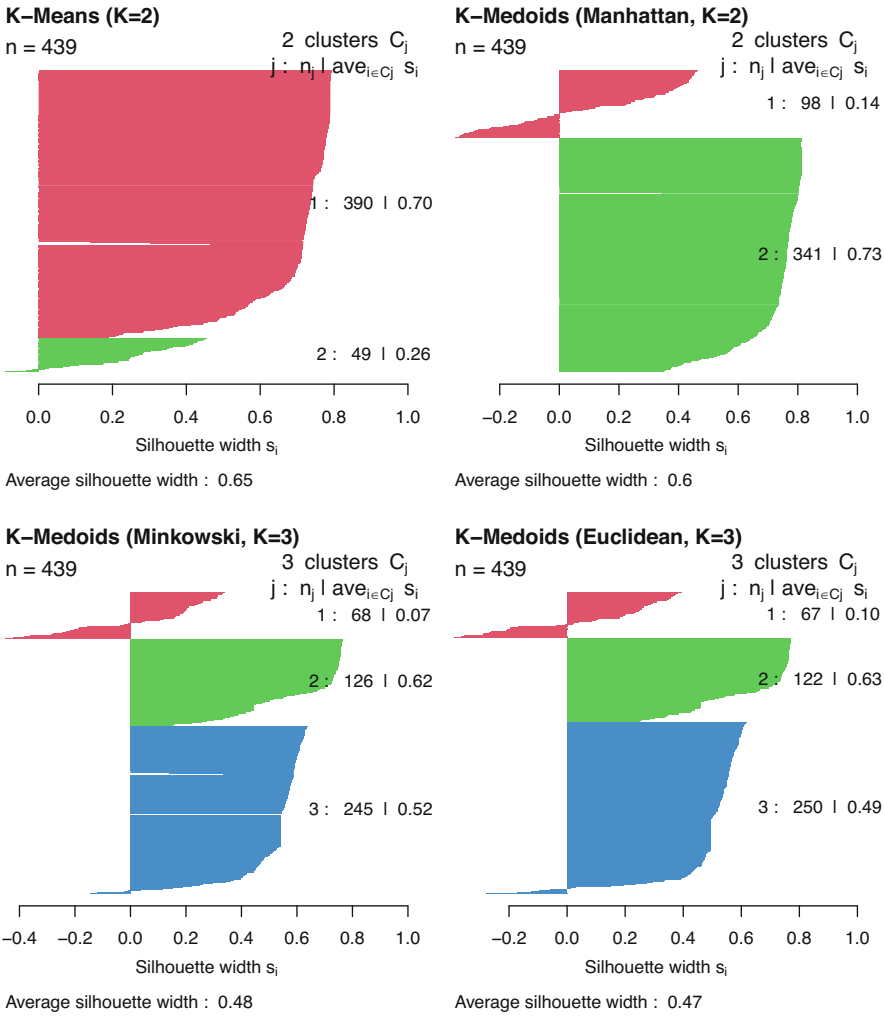


Fig. 8 Silhouette plots showing silhouette widths for a numbering of promising solutions, coloured according to cluster membership

Secondly, the first cluster has a higher cluster-specific average silhouette width under the solution based on the Euclidean distance. Indeed, this solution has fewer negative silhouette widths also:

```
sum(silhouette(pam_euclidean[[3]])[,3] < 0)
```

```
[1] 27
```

```
sum(silhouette(pam_minkowski3[[3]])[,3] < 0)
```

```
[1] 28
```

4.2.5 Interpreting the Optimal Clustering Solution

By now, we have identified that the $K = 3$ solution obtained using K-medoids and the Euclidean distance is optimal. Although aspects of this solution were already discussed in Sect. 4.2.2—in particular, Table 4 has already shown the $K = 3$ medoid vectors obtained at convergence—we now turn to examining this output in greater detail, in order to provide a fuller interpretation of each of the uncovered clusters. We extract this solution for ease of manipulation.

```
final_pam <- pam_euclidean[[3]]
```

Recall that this method yielded three clusters of sizes $n_1 = 67$, $n_2 = 122$, and $n_3 = 250$. Although the categorical variables were not used by this clustering method, additional interpretative insight can be obtained by augmenting the respective medoids in Table 4 with these cluster sizes and the `experience` values of the corresponding rows of the `merged_df` data set, which includes the additional demographic features.

```
df |>
  slice(as.numeric(final_pam$medoids)) |>
  round(4) |>
  mutate(size=table(final_pam$clustering)) |>
  left_join(slice(demog |>
                    select(UID, experience),
                  as.numeric(final_pam$medoids)),
            by=join_by(name == UID)) |>
  select(-name)
```

Interpretation and labeling of the clustering results is the step that follows, with a focus only on the medoid values of the centrality scores (had the optimal solution been obtained by `kmeans()`, we could instead examine its centroids in its `$centers` component, i.e., mean vectors). We will follow the example papers that we used as a guide for choosing the centrality measures [28] and [61]. Both papers have used traditional centrality measures (e.g., degree, closeness, and betweenness) as well as diffusion centralities (diffusion degree and coreness) to infer students' roles. Furthermore, the second paper has an extended review of the roles and how they have been inferred from centrality measures, so readers are encouraged to read this review.

Table 6 Medoids for the $K = 3$ K-medoids solution obtained using the Euclidean distance on the original data scale, augmented with the cluster sizes and the corresponding values of the `experience` variable (which was not directly used by the clustering algorithm)

InDegree	OutDegree	Closeness_total	Betweenness	Eigen	Diffusion.degree	Coreness	Cross_clique_connectivity	size	experience
16	14	0.0011	675.5726	0.1096	1415	18	157	67	1
1	1	0.0007	0.0000	0.0047	41	2	2	122	3
1	2	0.0010	29.4645	0.0194	684	3	5	250	2

As the data shows, the first cluster has the highest degree centrality measures (`InDegree` and `OutDegree`), highest betweenness centrality, as well as the highest values of the diffusion centralities (`Diffusion_degree` and `Coreness`). These values are concordant with students who were actively engaged, received multiple replies, had their contributions discussed by others, and achieved significant diffusion. All of such criteria are concordant with the role of *leaders*. It stands to reason that the *leaders* cluster would be the smallest, with $n_1 = 67$.

The third cluster has intermediate values for the degree centralities, high diffusion centrality values, as well as relatively high values of betweenness centrality. Such values are concordant with the role of an active participant who participates and coordinates the discussion. Therefore, we will use the label of *coordinators*.

Finally, the second cluster has the lowest values for all centrality measures (though its diffusion values are still fairly reasonable). Thus, this cluster could feasibly be labelled as an *isolates* cluster, gathering participants whose role in the discussions is peripheral at best. Overall, the interpretations of this $K = 3$ solution are consistent with other findings in the existing literature, e.g., [28].

We can now label the clusters accordingly to facilitate more informative cluster-specific summaries. Here, we also recall the size of each cluster with the new labels invoked, to demonstrate their usefulness.

```
final_pam$clustering <- factor(final_pam$clustering,
                               labels=c("leaders", "coordinators",
                                        "isolates"))
table(final_pam$clustering)
```

```
   leaders coordinators      isolates
        67          122           250
```

As an example, we can use these labels to guide a study of the mean vectors of each cluster (bearing in mind that these are not centroid centroid vectors obtained by K-means, but rather mean vectors obtained calculated for each group defined by the K-medoids solution), for which the interpretation of the leader, coordinator, and isolate labels are still consistent with the conclusions drawn from the medoids in Table 4. Note that, for the sake of readability, the group-wise summary below is transposed.

```
df |>
  group_by(clusters=final_pam$clustering) |>
  select(-name) |>
  summarise_all(mean) |>
  mutate_if(is.numeric, round, 2) |>
  pivot_longer(cols=-clusters,
               names_to="centrality") |>
```

```
    pivot_wider(names_from=clusters,
                values_from=value)
```

```
# A tibble: 8 x 4
  centrality                leaders coordinators isolates
  <chr>                       <dbl>        <dbl>    <dbl>
1 InDegree                    17.1          1.1      1.98
2 OutDegree                   18.8          1.68     3.62
3 Closeness_total              0            0        0
4 Betweenness                943.          21.2     99.1
5 Eigen                        0.13         0.01     0.03
6 Diffusion.degree          1466.         132.     742.
7 Coreness                    17.2          2.56     4.58
8 Cross_clique_connectivity  220.           4.53    12.6
```

From Table 6, we can also see that each observation which corresponds to a cluster medoid contains low, high, and medium levels of experience, respectively. However, one should be cautious not to therefore conclude that the clusters neatly map to experience levels, as the following cross-tabulation indicates little-to-no agreement between the groupings of experience levels in the data and the uncovered clusters.

```
table(final_pam$clustering,
      merged_df$experience,
      dnn=c("Clusters", "Experience"))
```

```
             Experience
Clusters       1  2  3
  leaders     17 23 27
  coordinators 40 34 48
  isolates    61 93 96
```

Finally, we can produce a visualisation of the uncovered clusters in order to better understand the solution. Visualising multivariate data with $d > 2$ is challenging and consequently such visualisations must resort to either plotting the first two principal components or mapping the pairwise dissimilarity matrix to a configuration of points in Cartesian space using multidimensional scaling. The latter is referred to as a "CLUSPLOT" [62] and is implemented in the `clusplot()` function in the same `cluster` library as `pam()` itself. This function uses classical (metric) multidimensional scaling [63] to create a bivariate plot displaying the partition of the data. Observations are represented by points in the scatter plot an ellipse spanning the smallest area containing all points in the given cluster is drawn around each cluster. In the code below, only `clusplot(final_pam)` is strictly necessary to produce such a plot for the optimal $K = 3$ Euclidean distance K-medoids solution; all other

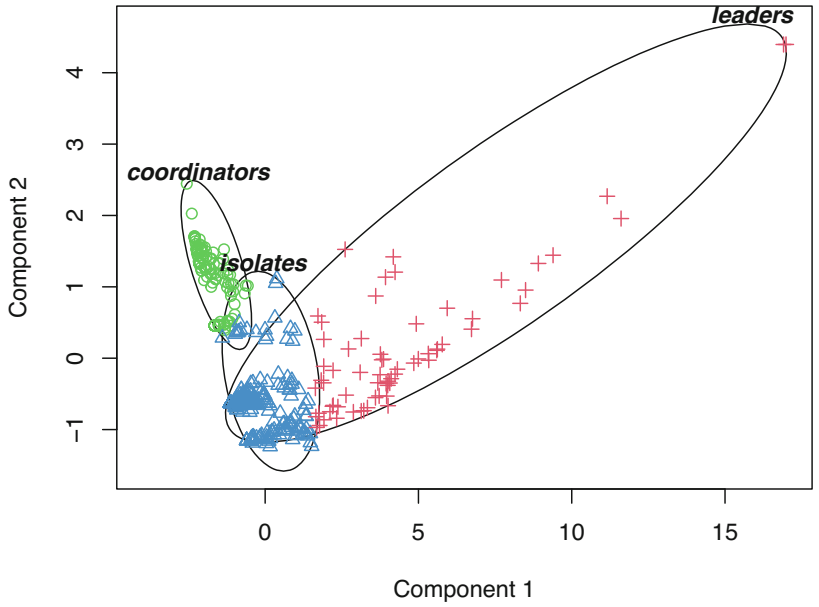


Fig. 9 Two-dimensional clustering plot for the final $K = 3$ Euclidean distance K-medoids solution obtained via classical multidimensional scaling. The ellipses around each cluster are given the associated labels of *leaders*, *coordinators*, and *isolates* and the points are coloured according to cluster membership

arguments are purely for cosmetic purposes for the sake of the resulting Fig. 9 and are described in `?clusplot`.

Figure 9 shows that the *leaders* cluster—the smallest cluster with the highest value for all centrality measures—is quite diffuse. Conversely, the larger *coordinators* cluster, with the smallest values for all centrality measures, and the *isolates* cluster, the largest of all, with intermediate values for all centrality measures, are more compact. This is consistent with the cluster-specific average silhouette widths shown in the bottom-right panel of Fig. 8. That being said, the large span of the *leaders* cluster again affirms the relative robustness of K-medoids to outliers, of which some (all of which are leaders) still remain despite the earlier pre-processing steps.

```
clusplot(final_pam,                                  # the pam() output
         main="", sub=NA,              # remove the main title and subtitle
         lines=0,                       # do not draw any lines on the plot
         labels=4,                                  # only label the ellipses
         col.clus="black",                 # colour for ellipses and labels
         col.p=as.numeric(final_pam$clustering) + 1, # colours for points
         cex.txt=0.75,                          # control size of text labels
         xlim=c(-4, 17)           # expand x-axis to avoid trimming labels
         )
```