

Diplomarbeit

Neue computerorientierte Lehr- konzepte in der Statistik

zur Erlangung des akademischen Grades eines

Diplom-Ingenieur

im Rahmen des Studiums

Statistik und Wirtschaftsmathematik

eingereicht von

Gregor de Cillia, B.Sc.

Matrikelnummer 0926164

ausgeführt am Institut für Stochastik und Wirtschaftsmathematik
an der Fakultät für Mathematik und Geoinformation
der technischen Universität Wien

unter Anleitung von

Priv.-Doz. Dipl-Ing. Dr.techn. **Matthias Templ**

Wien, 11.05.2018

Gregor de Cillia

Matthias Templ

Diploma Thesis

New Computer-Oriented Teaching Concepts in Statistics

submitted in partial fulfillment of the requirements
for the degree of

Diplom-Ingenieur

in

Statistics and Mathematics in Economics

by

Gregor de Cillia, B.Sc.

Registration Number 0926164

carried out on the Institute of Statistics and Mathematical Methods
in Economics
on the Faculty of Mathematics and Geoinformation
of the Vienna University of Technology

under the supervision of
Priv.-Doz. Dipl-Ing. Dr.techn. **Matthias Templ**

Vienna, 11.05.2018

Gregor de Cillia

Matthias Templ

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Wien, 11.05.2018

Gregor de Cillia

Kurzfassung

In der Vergangenheit wurden viele Behmühungen unternommen um ein System von feedbackbasierten *blended learning* Systemen zu schaffen. Neue Technologien und neue Ideen machen es möglich diese Systeme zu verbessern und diese auch für den Unterricht von Statistik auf Universitätsniveau zu verwenden.

Das Ziel ist sowohl eine systematische Implementierung von Lehrsoftware mittels moderner interaktiver Frameworks als auch die Entwicklung von neuen Verfahren zum Lehren in den Bereichen statistische Modellierung und statistische Unsicherheit.

Diese Arbeit präsentiert das neu entwicelte *R* Paket `tguishiny` (de Cillia et al., 2018), welches Lehrenden eine Möglichkeit gibt, feedback-basierte Kurse zu den Themen R-Programmierung und statistischer Modellierung zu halten.

Fragen, die mit diesem Paket generiert werden können interaktive Visualisierungen aus dem `shiny` Framework enthalten. Antworten, die von Studenten eingereicht wurden können über eine “teacher view” live beobachtet werden.

Das UI-Design der Fragen wurde weitreichend von dem `tGUIOnline` (Dinges and Templ, 2009) Projekt beeinflusst, welches in der Bundesanstalt Statistik Österreich (*Statistik Austria*) entwickelt und eingesetzt wurde.

Abstract

A lot of efforts has been made to implement a system of feedback-based blended learning systems in the past. New possibilities and new ideas make it possible to improve these systems in order to maximize the effectiveness of teaching statistics to students at university level.

The aim is both, a systematic implementation of teaching concepts in software using modern interactive tools and to invent new ways to teach statistical concepts in the area of statistical uncertainty and statistical modeling.

This thesis presents the newly developed *R* package `tguishiny` (de Cillia et al., 2018), which gives teachers a way to create feedback-based courses for teaching *R* syntax and statistical concepts.

Questions created with this package can contain interactive visualizations thanks to the powerful `shiny` framework. Answers submitted by students are summarized live in a “teacher view”.

The ui design of the questions has been largely influenced by the `tGUIOnline` (Dinges and Templ, 2009) project, which is a teaching tool developed in and used by the national statistical office of Austria .

Contents

1	Introduction	1
1.1	Digital- and blended learning	1
1.2	History of blended learning	1
1.3	The tGUIOnline project	3
1.4	Tguishiny overview	4
1.5	Digital learning projects in R	6
1.6	Outlook	9
2	Tools	11
2.1	RStudio	11
2.2	shiny	11
2.3	R6	11
2.4	Other useful packages	12
3	Tguishiny in detail	15
3.1	The tguiApp function	15
3.2	Question classes	15
3.3	Other classes	25
3.4	Teacher guide	26
4	Theoretical background on examples	29
4.1	Compositional principal component analysis	29
4.2	Robust estimators and the ilr transformation	30
4.3	Compositional biplots	32
5	Demonstration examples in tguishiny	35
5.1	Data	35
5.2	DfQuestions	35
5.3	PlotQuestions	37
5.4	An interactive box plot	39
5.5	Compositional principal component analysis	39
6	Summary	41
6.1	Archievements	41
6.2	Expandability of the project	42

Introduction

This thesis presents a newly created *R* package called `tguishiny` (de Cillia et al., 2018), which was developed for the Zurich University of Applied Sciences (ZHAW) and *Statistics Austria*.

This chapter covers historic milestones in *blended learning* followed by a quick overview of the package and finally presents some software projects similar to `tguishiny`. The subsequent chapters will cover the implementation and features of the package in detail and will showcase courses that have been developed with it.

1.1 DIGITAL- AND BLENDED LEARNING

With the development of technology in the past years, a lot of new teaching tools emerged. This chapter will briefly define the terms *digital-* and *blended learning* which are both related to this trend.

Digital learning is a broad term describing different types of technologically assisted learning models. These learning models are usually web based and give students the possibility to both learn and practice skills at the same time.

Blended learning is a special form of *digital learning* which “combines face-to-face with distance delivery systems [...] using the web for what it does best, and using class time for what it does best” (Osguthorpe and Graham, 2003).

This creates a clear distinction between the two terms since *blended learning* includes a face-to-face component. Therefore, a *blended learning* environment usually requires a course instructor.

1.2 HISTORY OF BLENDED LEARNING

This section is mainly based on Güzera and Caner (2014), where articles about *blended learning* between 1999 and 2012 have been studied. Since these reviews stop at 2012, the most recent developments in *blended learning* were summarized by the author of this thesis.

Table 1.1 has been taken from Güzera and Caner (2014) and shows a rapid increase in the number of publications about *blended learning* between the first two time periods. Notice that the “year-range” is equal for each time period (three years). This chapter is organized according to the classifications in this Table.

Classification	Sub-Classification	Year Range	Number of articles
Past	First attempts	1999-2002	125
	Definition period	2003-2006	1200
	Popularity period	2007-2009	1460
“Present”		2010-2012	1660

Table 1.1: Time periods for the publication of “blended learning” literature according to Güzera and Caner (2014).

First attempts

This period consists of publications from a very broad variety of research fields including prekindergarden-schooling, military courses and inter-cultural trainings. Articles from this period explored the idea of supporting online-learning with traditional learning but there were no exact definitions on *blended learning*.

Definition period

The most cited articles published in this period are on defining *blended learning* and exploring its possibilities and benefits. The definition given in Section 1.1 was given in this period.

One article from Garrison and Kanuka discusses *blended learning* in higher education considering resource management, scheduling and planning of courses.

Popularity period

The reviewed articles in Güzera and Caner (2014) from this period were centered around two topics. The first topic is the perception of participants on *blended learning* and *online learning* concepts. All in all, *blended learning* received good feedback in those articles, although instructions in a traditional setting were perceived to be clearer.

“ ... the general conclusion in all studies indicated that students favored web-based online learning environment as effective but they did not want to give up from face to face component of the course. (Osguthorpe and Graham, 2003) ”

The second group of articles reviewed in the “popularity period” are centered about the effectiveness of *blended learning* systems. Effectiveness has been measured in a different manner for different articles.

“ [...] there is no significant difference on achievements of students between blended learning and traditional learning but on the other variables like satisfaction, motivation, drop-out rate for at-risk students, attitude and knowledge retention blended learning is observed as superior. (Osguthorpe and Graham, 2003) ”

“Present”

Seven articles from this time period (2010 - 2012) have been reviewed in Güzera and Caner (2014) which cover a broad variety of subjects. Two of those articles were focused around learning English as a foreign language and showed that *blended learning* is not only well received in this area but also increased the scores of participating students compared to traditional methods.

After 2012

Recently, several publications on *blended learning* in health education appeared such as McCutcheon et al. (2015) and de Jong et al. (2014). Most of these publications covered an retrospective evaluation of *blended learning* programs that were held for med students.

A meta-analysis about the `blended learning` in medical education is available at Liu et al. (2016). The conclusion looks as follows.

“ Blended learning appears to have a consistent positive effect in comparison with no intervention, and to be more effective than or at least as effective as nonblended instruction for knowledge acquisition in health professions. Due to the large heterogeneity, the conclusion should be treated with caution. (Liu et al., 2016) ”

Another interesting group of publications from this time period deal with *blended learning* for K-12 education. See Dichev et al. (2013) and Horn et al. (2013).

1.3 THE TGUIONLINE PROJECT

This section covers the most important aspects of the `tGUIOnline` project, which was a major influence in the development of the `tguishiny` package.

`tGUIOnline` was firstly introduced in Dinges and Templ (2009) as “being an effective instrument to train and teach staff on mathematical and statistical topics” (Dinges et al., 2011b). It uses the `rApache` (Horner, 2013) framework to create an interactive learning experience for statistical concepts. An example of a multiple choice question generated with this framework can be seen in Figure 1.1.

Students can use several control elements to alter the plot on the right, which will give insights about the dataset `mtcars` and possible dependencies between its variables. The graphics are then interpreted by the students via a multiple choice interface. Teachers will get live updates about how often each answer possibility has been chosen among all students with a separate “teacher GUI”.

The `tGUIOnline` project uses a visual approach for its teaching purposes. More examples and an in-depth description of the project can be found in Dinges et al. (2011b).

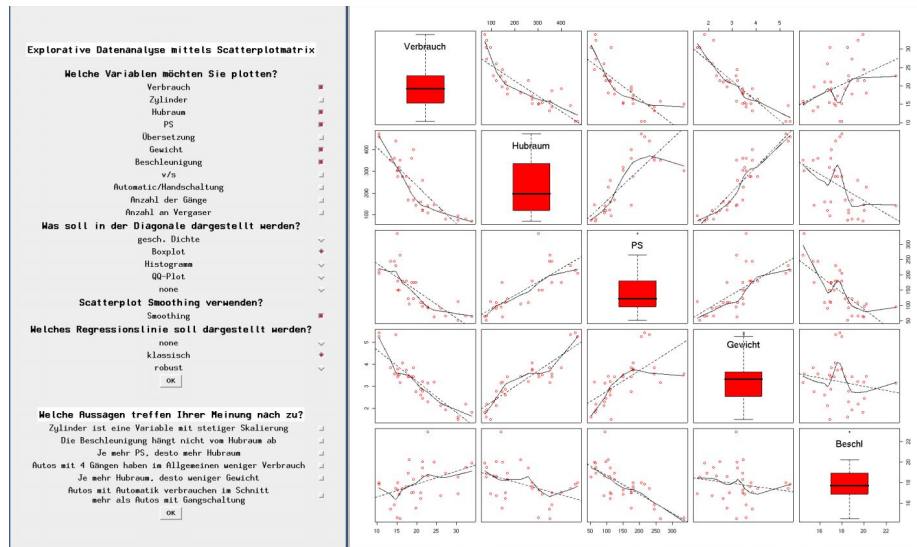


Figure 1.1: Exercise from tGUIOnline. The students can select variables to be plotted in an interactive manner and they should answer the multiple choice question related to this exercise.

1.4 TGUI SHINY OVERVIEW

The *R* package `tguishiny` tries to implement similar concepts as the `tGUIOnline` project using the `shiny` (Chang et al., 2017) package instead of `RApache` for the *GUI* development. `shiny` is a web framework for the *R* language, which severely increased in popularity over the past two years. See the download statistics in Figure 1.2.

Why switch from rApache to shiny?

The `shiny` package provides good documentation and easy syntax to create web applications. The fact that shiny is based on *HTML* gives lots of flexibility for front-end development by using custom *CSS*- or *JavaScript* libraries.

Developers that are not familiar with web programming can even use a lot of features, e.g. packages like `plotly`, `ggvis` and `DT` can be used to add new input and output elements to shiny applications.

The `shiny` framework also follows the “reactive” programming paradigm, so sending submissions from one machine to another can be done easily. To archive the same result in `rApache`, it is necessary to read the same database in a periodic manner.

Main concept

`tguishiny` aims at teaching statistical concepts and corresponding usage of

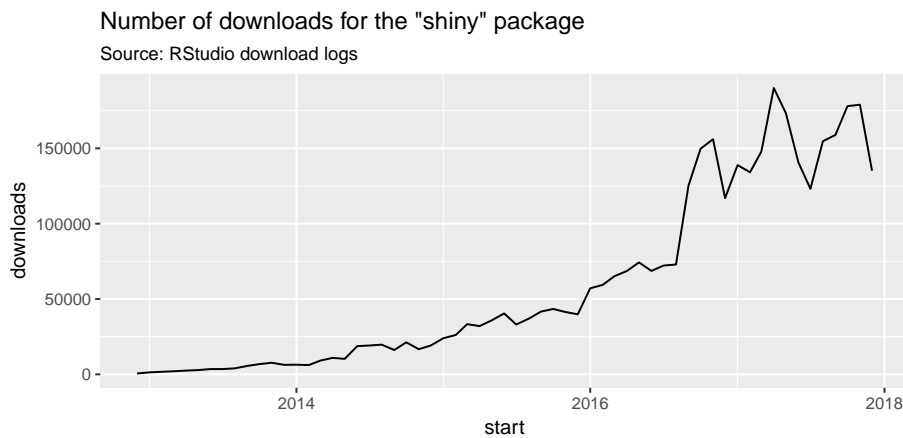


Figure 1.2: Monthly number of downloads for the package `shiny` from <http://cran.rstudio.com/> within the past years. The data for this plot has been acquired with the package `dlstats` (Yu, 2017).

the programming language *R* with *interactive* courses, which may or may not be assisted by a teacher. Similar to `tGUIOnline`, visualizing those concepts is a key issue and teachers will get *immediate feedback* about any user submissions.

The library uses an *object oriented interface* for the questions to increase flexibility and ease the creation of new questions. Knowledge about the `shiny` package is not required for the authors of questions, although it is recommended for creation of exercises that contain interactive graphics.

Unlike most other products, `tguishiny` allows the creation of questions that *take R code as an answer*. The user-code will be evaluated upon submission and the evaluation of all submitted answers is visible to the teacher. This makes it possible to teach practical *R*-skills to students on many different levels of complexity.

Differences between tguishiny and tGUIOnline

The majority of the courses created in the `tGUIOnline` framework have already been translated to `tguishiny`. It is therefore mostly backwards compatible with the features provided in the old framework. Additionally, new courses have been developed by students of the Zurich University of Applied Sciences (ZHAW).

New features include questions that expect *R* input from students, which opens quite some new possibilities for course design. Teachers can focus on the practical use of *R* for data science rather than limiting their questions to statistical concepts and interpretations of statistical tests.

The fact that `shiny` is used as a backend for the server-ui communication means that teachers familiar with `shiny` can easily create questions contain-

ing interactive contents to make questions more interesting for the students. Existing `shiny` applications can also be converted into questions quite easily.

1.5 DIGITAL LEARNING PROJECTS IN R

This section only addresses rather recent teaching tools interacting with the *R* language. All of the tools presented are open-source and can be considered *digital learning* tools.

1.5.1 *swirl*

Swirl is a very powerful way for students to learn *R* on their own. According to the documentation, the package

“ teaches you *R* programming and data science interactively, at your own pace, and right in the *R* console! (Kross et al., 2017) ”

Students are asked questions in the console in a prompt-like fashion. Answers can be typed directly into the console. Code will be submitted with the return key and submitted code will be evaluated. The next part of a course is only unlocked once the current part is answered correctly.

```
|=====| 36%
| Recall from the previous lesson that sapply(flags, class) will return a character vector containing
| the class of each column in the dataset. Try that again now to see the result.

> sapply(flags, class)
      name landmass      zone      area population language religion      bars stripes
"factor" "integer" "integer" "integer" "integer" "integer" "integer" "integer" "integer"
colours   red      green   blue      gold      white   black   orange   mainhue
"integer" "integer" "integer" "integer" "integer" "integer" "integer" "integer" "factor"
circles   crosses saltires quarters sunstars crescent triangle icon   animate
"integer" "integer" "integer" "integer" "integer" "integer" "integer" "integer" "integer"
text      topleft botright
"integer" "factor" "factor"

| You are really on a roll!

|=====| 40%
| If we wish to be explicit about the format of the result we expect, we can use vapply(flags, class,
| character(1)). The 'character(1)' argument tells R that we expect the class function to return a
| character vector of length 1 when applied to EACH column of the flags dataset. Try it now.

> vapply(flags, class, character(1))
      name landmass      zone      area population language religion      bars stripes
"factor" "integer" "integer" "integer" "integer" "integer" "integer" "integer" "integer"
colours   red      green   blue      gold      white   black   orange   mainhue
"integer" "integer" "integer" "integer" "integer" "integer" "integer" "integer" "factor"
circles   crosses saltires quarters sunstars crescent triangle icon   animate
"integer" "integer" "integer" "integer" "integer" "integer" "integer" "integer" "integer"
text      topleft botright
"integer" "factor" "factor"
```

Figure 1.3: A Screenshot of a *swirl* course teaching students the use of the `lapply` function. *R* code submitted by the users is evaluated and new questions are loaded as soon as a right answer is given.

It is possible for teachers to create custom courses with the function `swirlify::new_lesson`. Below is a screenshot showing a *swirl* course. This course can be accessed with the `swirl` function.

The main disadvantage when it comes to the usage of `swirl` in *blended learning*

is that teachers are not able to see what students are answering in swirl courses. On the other hand, the package is lightweight and uses a simple instructive interface for their courses.

1.5.2 Jupyter

Jupyter is meant to be used as a way to create notebooks containing live code, \LaTeX equations and plots (Kluyver et al., 2016). More than 40 programming languages are supported, including R. No installation of R is needed in order to run a Jupyter notebook, since free online servers like <https://try.jupyter.org/> are available.

This makes Jupyter a very handy tool to create reproducible research across operating systems. Teachers can use the project to create questions in the form of notebooks and ask students to fill in some missing pieces.

Jupyter R demo

The following markup is produced *with* **markdown**

In [1]:

```
2+2
```

4

In [2]:

```
head(mtcars, 5)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2

Figure 1.4: A Jupyter notebook showcasing dynamic evaluation of R code and table markup.

Again, there is no way for the teacher to receive the submissions of students unless the notebook is sent to the teachers machine. Jupyter is therefore only suited for online-courses but not very well for *blended learning*.

1.5.3 learnr

A package developed by the *RStudio* team based on *R Markdown*.

“ The `learnr` package makes it easy to turn any *R Markdown* document into an interactive tutorial. Tutorials consist of content along with interactive components for checking and reinforcing understanding. (Allaire, 2017b)

”

One really handy advantage of this package is, that it is well documented, and allows questions that ask for *R* code as an answer. Embedding `shiny` components is also possible just like it is with other `R Markdown` documents.

The resulting tutorials can be published via <http://shinyapps.io> using the “publish” button in *RStudio*. Feedback for teachers is not supported.

Modify the following code to limit the number of rows printed to 5:

Code
Start Over
Hint
Run Code

```
1 head(mtcars, 5)
2
3
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0

5 rows | 1-10 of 12 columns

Figure 1.5: A sample course created with `learnr` that allows *R* input by users. Clicking “run code” will evaluate the *R* input and update the table accordingly.

1.5.4 exams

The `exams` (Grün and Zeileis, 2009) package is described as an

“all for one approach for to automatic exams generation.”

The package provides interfaces to moodle. Teachers can either host exams online or print them. Printed exams can be graded automatically based on scans of the answered sheets. This project is used in a number of universities in Austria and can therefore be considered well-tested and stable. New questions can be created with either \LaTeX or *Markdown* and the following question types are available.

- **schoice** Single choice questions that allow random reshuffling of the answer order.
- **mchoice** Multiple choice questions that allow random reshuffling of the answer order.
- **num** Ask users to calculate a single numeric value. Tolerance intervals can be used.
- **string** Ask users to enter a `string` that will be compared to the answer `string`.
- **cloze** Longer exercise containing several sub-exercises.

Figure 1.6 shows a question created with the `exams` package.

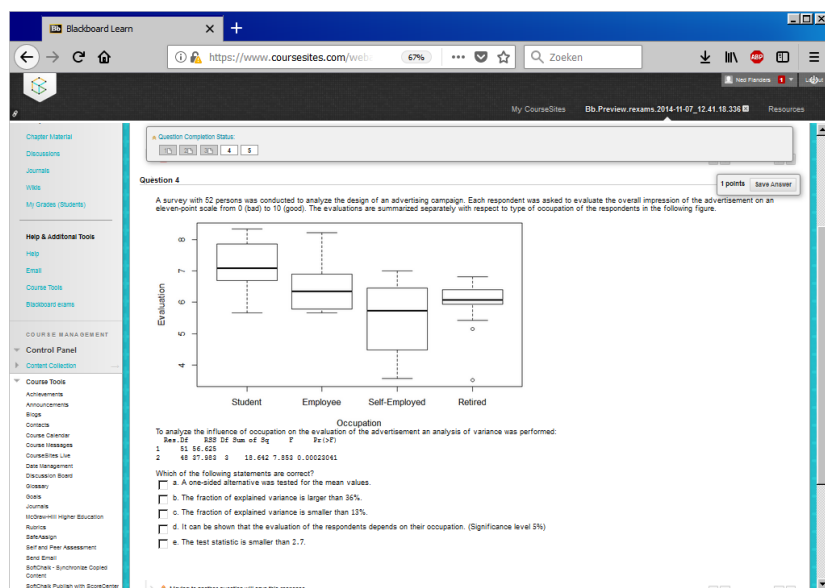


Figure 1.6: The package `exams` used in a moodle course. Students are asked to interpret a statistical graph via a multiple choice question.

1.5.5 Summary

The Table 1.2 summarizes the most important features for all the projects presented in this section as well as `tGUIOnline` and `tgushiny`.

	Automated grading	Accepts <i>R</i> code from students	Feedback for teachers	Can be hosted on the web
Swirl	✓	✓		
Jupyter		✓		✓
learnr	✓	✓		✓
exams	✓		✓	✓
tguiOnline	✓		✓	✓
tgushiny	✓	✓	✓	✓

Table 1.2: Summary of features for the discussed projects.

We can see that most frameworks support automatic grading of answers and can also be hosted in web pages. However, accepting *R* code as answers and giving feedback for teachers are rather rare features.

1.6 OUTLOOK

The following chapters will present the `tgushiny` package in detail and showcase a rather convoluted example created with the package.

- Chapter 2 lists the most important R packages that are used as dependencies for `tgushiny` and discusses why those packages have been chosen.

- Chapter 3 presents details about the design of the package and gives insight about the different kinds of questions that can be created with it. The Chapter also includes a teacher guide which describes how the package can be used to include custom questions.
- Chapter 4 describes different methods for principal component analysis (*PCA*) of compositional data.
- Chapter 5 showcases a more complex example developed with `tgushiny`. This example includes questions about manipulating tables, creating statistical graphs and interpreting outputs of compositional biplots.
- Chapter 6 summarizes the features developed in the package and gives an overview of possible improvements.

In this chapter, we will talk about the most important tools used when creating the `tguishiny` project.

2.1 RSTUDIO

RStudio (RStudio Team, 2016) is an integrated development environment (*IDE*) for *R* and widely used by the *R* community. A lot of packages used in this project are developed by the *RStudio* team and the author highly encourages the use of *RStudio* together with the `tguishiny` package.

2.2 SHINY

As the name suggests, the package `shiny` (Chang et al., 2017) is the single most important dependency for the `tguishiny` project. A short description about the functionality provided by this package can be found in Chang et al. (2017).

“ Shiny is an open source *R* package that provides an elegant and powerful web framework for building web applications using *R*. Shiny helps you turn your analyses into interactive web applications without requiring *HTML*, *CSS*, or *JavaScript* knowledge. ”

Most of the *JavaScript* code necessary to run the `tguiApp` is part of the `shiny` package or packages depending on `shiny`.

2.3 R6

This package serves as a tool for *object oriented programming*. It will become clear in Chapter 3 why this is important for this project. For now, it is only relevant to note, that the inheritance feature will play a big role for the question templates.

“ The R6 package allows the creation of classes with reference semantics, [...] These classes allow public and private members, and they support inheritance [...]. (Chang, 2017) ”

There are a lot of other solutions in *R* for *object oriented programming* like the `setRefClass` function, which could be used in place of `R6`. However, the lightweight nature of the package ultimately was the reason this one was picked.

2.4 OTHER USEFUL PACKAGES

Packages are a very important part of the *R* language, giving users the ability to load functions and other objects that help them with their specific tasks. The `tguishiny` package contains dependencies to a various number of packages which can be displayed via the following code. Notice that some of the dependencies are just necessary to build certain questions while others are used for the question classes and the `tguiApp` function itself.

The dependencies can be seen when typing the following in *R*.

```
pack <- installed.packages()
pack["tguishiny", "Depends"]

## [1] "shiny"

cat( pack["tguishiny", "Imports"] )

## evaluate,
## shinyAce,
## png,
## R6 (>= 2.2.2),
## reshape2,
## dplyr,
## plotly (>= 4.7.1),
## RSQLite,
## ggplot2 (>= 2.2.1.9000),
## DT,
## stringr,
## rhandsontable,
## shinyjs,
## shinyBS
```

This section will present the most important of those dependencies and shortly describes their main use cases.

2.4.1 *ggplot*

`ggplot2` is a plotting system for *R*, based on the grammar of graphics (Wickham, 2016b). Graphics produced with this package *can be saved as objects* which has big advantages for the `tguishiny` project. The class `PlotQuestion` can be used to ask students to create `ggplot` objects which are then compared with the `ggplot` object specified by the question author. Most *R* users are familiar with the package since it is one of the most popular *R* packages.

The following code snippet uses the `ggplot2` package to produce Figure 2.1.


```
library(ggplot2)
mtcars$cyl = as.factor(mtcars$cyl)
gg <- ggplot(mtcars, aes(wt, mpg, color = cyl, size = hp)) +
  geom_point()
gg
```

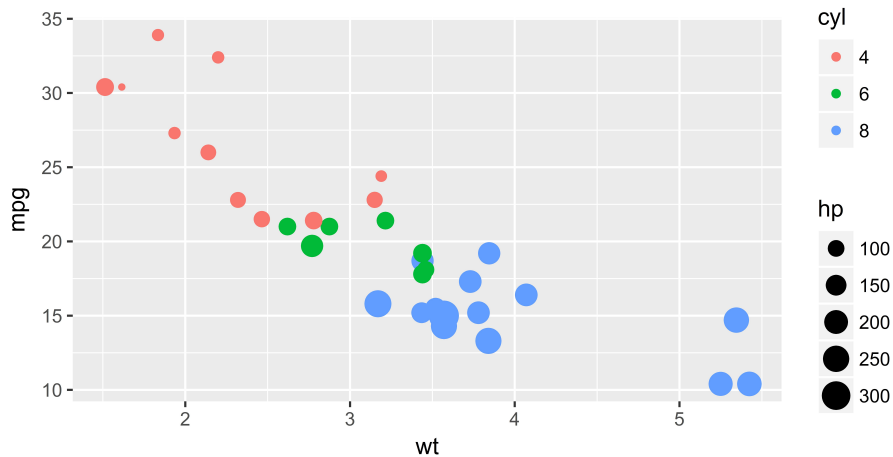


Figure 2.1: A plot created with the `ggplot2` package. Colors and sizes are used to display the variables `cyl` (number of cylinders) and `hp` (horse-power).

2.4.2 Plotly

“ A *JavaScript* library, that lets users easily create interactive charts and dashboards to share online with their audience (Sievert et al., 2017).

”

The `plotly` team also released a *R* package to create `plotly` graphs from the *R* console. This package also includes a function called `ggplotly` which converts `ggplot` objects into interactive charts. The `plotly` package is used in the `PlotQuestion` class.

2.4.3 DT

This *R* package is an *R* interface for the *JavaScript* library *DataTables*. *R* data objects (matrices or data frames) can be displayed as tables on *HTML* pages, and `DT` provides filtering, pagination, sorting, and many other features in the tables. (Xie, 2016)

2.4.4 shinyjs

`shinyjs` lets you perform common useful *JavaScript* operations in Shiny apps that will greatly improve your apps without having to know any *JavaScript* (Attali, 2016). Currently, the main use for this package is enabling and disabling input elements like check boxes and text fields.

2.4.5 evaluate

According to the documentation, `evaluate` (Wickham, 2016a) provides the following features.

“ Parsing and evaluation tools that make it easy to recreate the command line behavior of *R*. (Wickham, 2016a) ”

The package is used by the `RQuestion` class to mimic a console output. Without this package, users couldn't know if their code contains syntax errors before submitting it.

Below is a short code snippet that demonstrates how `evaluate` operates. The output of the `replay` function is a collection of strings that look just like the output from the *RStudio* console.

```
library(evaluate)
replay(evaluate("x <- 2\nx <- x + 1\nx\ny"))

## > x <- 2
## > x <- x + 1
## > x
## [1] 3
## > y

## Error in eval(expr, envir, enclos): object 'y' not found
```

Notice that even error messages are displayed correctly. In this case the variable `y` was not defined. Errors, warnings and plots can be captured by providing *callback functions* to the `evaluate` call.

2.4.6 rmarkdown

One of the latest additions to `tguishiny` is the `MarkdownQuestion` class, which relies on the `knitr` (Xie, 2014) package to include dynamic code into a question. One advantage of this approach is that new questions can more easily be created from existing *Markdown*, or *R Markdown* (`.Rmd`), documents. Another advantage is, that *Markdown* is intended to be as easy-to-read and easy-to-write as is feasible, which can save a quite some time when revisiting documents that haven't been altered for a while.

Tguishiny in detail

In this Chapter we will take a closer look at the package details and present its most important features. First, the function `tguiApp` is described. Then, the *question classes* provided in the package and their common usage are discussed. Subsequently, a guide on how to use the package as a teacher is provided.

3.1 THE TGUIAPP FUNCTION

This function will start a shiny app containing several questions. The questions can be accessed via a dropdownmenu on the top. The functions parameters are shown in Table 3.1.

Argument	Description
<code>admin</code>	Should the app be logged in as admin at startup? Alternative is user. Ignored if <code>logged_in</code> is <code>FALSE</code> .
<code>logged_in</code>	Should the app be logged in at all at startup? Default is <code>TRUE</code> .
<code>reset_database</code>	Should the database be reset prior to actually invoking the app? If <code>TRUE</code> , all previous answers submitted in the app will be lost.
<code>db_path</code>	Path to your database file. If the file does not exist, it will be created.
<code>questions_path</code>	Path to the question-pool you want to include in the app.
...	Further arguments passed down to <code>runApp</code> .

Table 3.1: Arguments of the `tguiApp` function.

The arguments `admin` and `logged_in` specify the initial state of the application. In most cases `logged_in = TRUE` makes sense for testing purposes while `logged_in = FALSE` is advisable when deploying the app on a web server.

The `db_path` argument links to a database file inside the package folder by default. Again, this is very handy for testing of the app but not recommended for deploying unless the package is used by no other users on the server.

`questions_path` can be used to replace the default question pool with a custom one. More details about this option can be found in the `README.md` file inside the default questions folder.

3.2 QUESTION CLASSES

The `tguishiny` package contains 10 *question classes* which serve as *templates* for creating new questions. These classes were built with the `R6` pack-

age (Chang, 2017). Each class uses a shiny module (Cheng, 2017) internally to avoid *ID* conflicts in the resulting *HTML* code. The inheritance between the different classes is summarized in Figure 3.1.

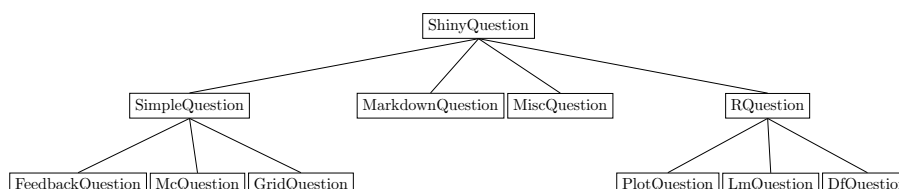


Figure 3.1: Inheritance diagram: A summary of all question classes currently included in the `tguishiny` package.

`shinyQuestion` serves as a *base class* and all other classes are (directly or indirectly) *derived* from this class. This means that all features available for `ShinyQuestion` are also accessible in other classes.

Similarly, the `RQuestion` class, which provides students a way to submit *R* code as an answer, inherits its features to the classes `PlotQuestion`, `LmQuestion` and `DfQuestion`. For a better understanding, we will discuss some of those classes briefly.

3.2.1 ShinyQuestion

This class serves as a *base class* for all other *question classes*. It can be viewed as an abstract class in the sense that it is not supposed to be used directly but rather defines an interface to be used by the `tguiApp` function.

A minimal object of the class `ShinyQuestion` can be created as follows.

```
questionObject <- ShinyQuestion$new()
```

The `show_interactive` method defined in this class gives authors of a question a way to run questions as standalone `shiny` applications. These standalone apps already provide an *SQL* connection to a database stored in the random access memory.

This way, questions can be tested comprehensively before they are included in the `tguiApp`. In the example above, the object `questionObject` can generate an app via.

```
questionObject$show_interactive()
```

The methods `bind_outputs` and `ui` are useful for *embedding* questions into `shiny` applications. The implementation of those two methods relies on shiny modules (Cheng, 2017) and can be used in the following manner.

```
shinyApp(
  ui = fluidPage(
    questionObject$ui()
  ),
  server = function(input, output, session){
    questionObject$bind_outputs()
  }
)
```

It is possible to include pictures and videos from static resources by passing the resource paths to the parameters `defaultPic`, `evalPic`, `defaultVideo` and `evalVideo`. `defaultPic` and `evalPic` can also be used with expressions as a convenient way to include *R* plots into questions.

```
library(tguishiny)
set.seed(1234)
questionObject <- ShinyQuestion$new(
  default_pic = expression({hist(rnorm(1000))}),
  eval_pic = expression({
    plot(cumsum(rnorm(1000)), type = "l")
  }),
  language = "english"
)
# display the question
questionObject$show_interactive()
```

The code shown above will start a `shiny` application containing two plots. Those plots will be visible in the default tab and eval tab, see Figure 3.2.

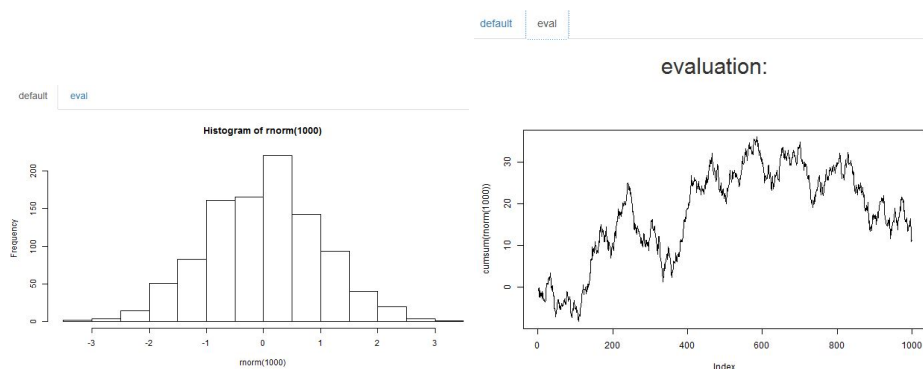


Figure 3.2: A `ShinyQuestion` including two pictures: one in the “default view” and one in the “evaluation view”.

Since all other question classes derive from the `ShinyQuestion` class, those classes also have access to the `default_pic` and `eval_pic` parameters as well as the member-functions discussed above.

Furthermore, shiny modules (Cheng, 2017) can be embedded in questions to allow even more flexibility when it comes to question authoring. For example, the following *R* code will create a question that embeds a `sliderInput`. Notice that it is necessary to use the namespace-function `NS` in `text`.

```
mq <- ShinyQuestion$new(
  server = function(input, output, session){
    output$ui <- renderUI({
      sliderInput('slider_id', 'slider', 0, 1, 0)
    })
  },
  text = function(id){
    ns <- NS(id)
    fluidPage(uiOutput(ns('ui')))
  }
)
```

3.2.2 McQuestion

This class makes it possible to create multiple choice questions as well as single-choice questions in `tguishiny`. At construction time, the parameters `choices` and `correct_answers` must be supplied.

```
myQ <- McQuestion$new(
  choices = c("A wrong answer", "A correct answer",
             "Another correct answer"),
  correct_answers = 2:3
)

myQ$show_interactive()
```

Figure 3.3 shows a typical object of this class. An answer is considered correct, if all marked choices are true statements and all unmarked choices are false statements.

The constructor arguments for the `McQuestion` class can be seen in Table 3.2. Notice that setting `multiple` to `FALSE` will create a single choice question.

Which of the following distributions is discrete?

☐ Binomial distribution
☐ Normal distribution
☐ Geometric distribution
☐ Logarithmic distribution
☐ Chi-squared distribution

Figure 3.3: A multiple choice question produced with the `tguishiny` package.

Argument	Description
<code>choices</code>	Possible answer choices. Should be a character vector or a list of ui objects.
<code>correct_answers</code>	Integer vector containing the indices of the correct answers.
<code>help_text</code>	Help text to be displayed at the bottom of the question.
<code>multiple</code>	Are multiple answers allowed? If <code>FALSE</code> , this question will display a single choice question (exactly one answer must be chosen). Default is <code>TRUE</code> .
<code>inline</code>	If <code>TRUE</code> , render the choices inline (i.e. horizontally). Default is <code>FALSE</code> .
<code>...</code>	Arguments getting passed down to <code>ShinyQuestion\$new</code>

Table 3.2: Arguments for the `McQuestion` constructor function.

One advantage of this class is that it is very easy to use and question authors might already have multiple choice questions prepared with other frameworks that can be ported easily into `tguishiny`.

3.2.3 RQuestion

`RQuestion` objects display a text-input field where users can enter *R* code. This code can be evaluated and the results will be displayed in another text field. If answers are submitted, the code in the text-input field will be sent to a *SQL* database for persistent storage.

The initialization-parameters contain a so-called `answer_script` which represents the correct answer in form of an expression. Derived classes might also allow other *R* objects as valid `answer_script` parameters¹. The evaluation routine will check whether all objects defined in the `answer_script` are also created by the code submitted by the student. For example, the following `RQuestion` object defines the variables `x` and `a` in the `answer_script`.

¹For example, `PlotQuestion` allows `ggplot` objects. See the subsection below.

```
my_q = RQuestion$new(
  answer_script = expression({ x = 0; y = "a" }),
  text = p("Assign the value", code(0), "for", code("x"),
           "and", code("'a'"), "for", code("y"), "."),
  title = "R Question")
```

The corresponding question (Figure 3.4) will expect the student to define variables with the same names and values to be considered correct. The values of the variables are compared with `isTRUE(all.equal(...))`.

Figure 3.4: A `RQuestion` that asks users to define two variables. The `shinyAce` (Trestle Technology, LLC, 2016) package is used for the input and output fields.

Students can *evaluate* their code and see whether syntax errors or unreasonable outputs are produced. Once they are confident with their answer, they can *submit* it and the submission will be immediately visible to the teacher.

The input- and output fields in the UI have been implemented with the `shinyAce` (Trestle Technology, LLC, 2016) package, which serves as an *R* interface for the *JavaScript* library *Ace Editor*. The output text is created with the `evaluate` (Wickham, 2016a) package.

If users type plotting code into the input field, a plot will be displayed under the output field. Those plots are generated with the function `renderPlot`.

3.2.4 *PlotQuestion*

This class asks users to create a plot which will be compared to the plot produced by the `answer_script` when users submit their answers. Currently, it is suggested to only use `ggplot` graphics with this class, since older *R* versions make it difficult to save ordinary plots as *R* objects².

If the answer plot is a `ggplot`, it will be compared to the user submitted plot on a pixel by pixel basis. Internally, the function `ggsave` is used to perform this. The evaluation routine will consider the answer as correct, if all pixels of the saved image files match and as wrong if at least one pixel is incorrect. Partial scores are not supported.

In the teacher interface, a list of all submitted answers is visible. When selecting one of those answers, the corresponding plot of the student will be generated and displayed to the teacher. Figure 3.5 illustrates this behavior.

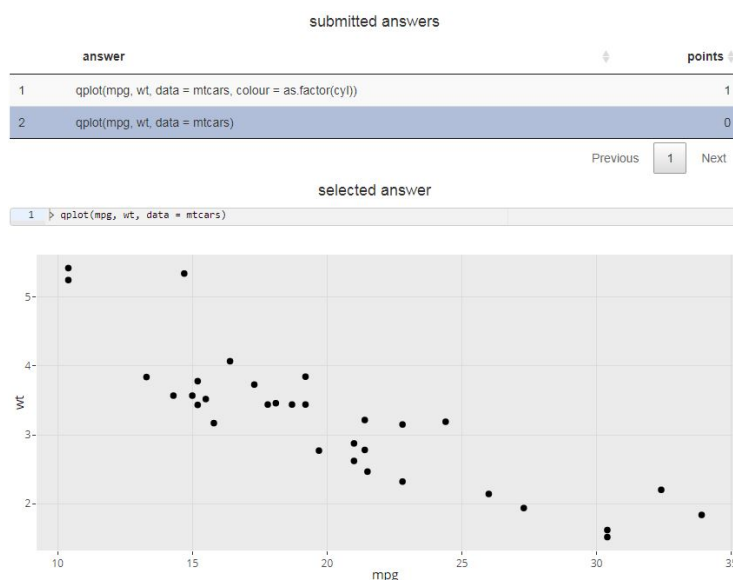


Figure 3.5: The teacher interface for the `PlotQuestion` class. Teachers can choose any of the submitted answers and view the resulting graph.

If the answer plot of the question is a `ggplot` object, the function `ggplotly` from the package `plotly` (Sievert et al., 2017) is used to automatically add tooltips and other interactions to the plot. This feature can be disabled with the `use_plotly` argument in `PlotQuestion$new`.

²For details, please refer to the documentation of `recordPlot` from the `grDevices` library. Apparently, *R* version 3.3.0 is necessary to guarantee a reliable storage of non-`ggplot` graphics as objects.

3.2.5 DfQuestion

A question of this kind asks users to create a `data.frame` or another R object deriving from `data.frame` like a `tibble`. The table produced by the student's R code will be compared to the table generated from the `answer_script` on a column-by-column basis.

This question class is useful to teach data manipulation techniques like changing variable types, discretization of continuous data and casting tables from long-to wide format. Some examples are available in Chapter 5.

3.2.6 MarkdownQuestion

Questions of this type are constructed with a *R Markdown* (Allaire et al., 2017) file which will be embedded in the question. The main advantage of this class is that it can contain several other `ShinyQuestion` objects. It is therefore suitable for creating longer exercises.

The included questions can be displayed in the *Markdown* file with the `include_question` function as demonstrated in the following code listing. Currently only *R Markdown* v1³ syntax is supported since the `.Rmd` file is rendered with the `knit2html` command.

```
md = "
# Hello World!
This text is *formatted* `with` **markdown**

```{r}
include questions
include_question(1)
... other R output
rnorm(10)
... and outputs defined in the server
uiOutput(ns('fromServer'))
```
"

mq <- MarkdownQuestion$new(
  markdownFile = textConnection(md),
  questions = list(McQuestion$new(letters[1:3], correct = 2:3)),
  server = function(input, output, session){
    output$fromServer = renderUI({
      sliderInput("a", "slider", 1, 2, 3)
    })
  }
)
```

³*R Markdown* v1 (version 1) is an old version of the *R Markdown* standard that depends on the **markdown** package rather than **pandoc**. Both versions (standards) are somewhat similar, but there are certainly cases, where documents following the v2 standard can not be rendered by the **markdown** package and vice versa.

The exercise presented in Chapter 4 is constructed with the `MarkdownQuestion` class. A screenshot of the app defined above can be seen in Figure 3.6.

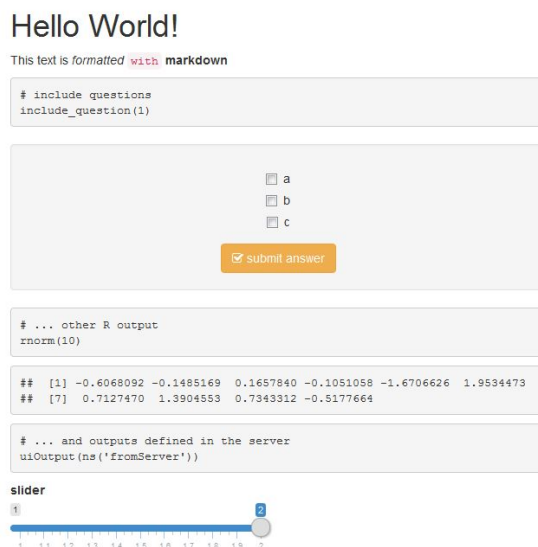


Figure 3.6: A `MarkdownQuestion` including dynamic code and a `McQuestion`. The end of the question shows a slider created with `shiny`.

As we can see, the expression `rmnorm(10)` is evaluated just like it would be in a usual *R Markdown* file. The call to `include_question` will display a `ShinyQuestion` object.

3.2.7 Experimental classes

There are three more question classes implemented in `tguishiny`. Those classes are still under development and the interfaces might change significantly. Therefore, these classes will only be discussed briefly.

GridQuestion is a class that asks users to assign several answers to several questions. The student interface is similar to a likert questionnaire.

FeedbackQuestion has been inspired by the `tGUIOnline` project and captures feedback about the courses in form of a text input box. Similar to the `RQuestion` interface, an `aceEditor` is used to capture the user inputs.

MiscQuestion is a very generic class that gives a lot of flexibility in designing questions. Arbitrary shiny components can be used to define the teacher- and student interface. This class is currently used in the “datacollection” shown in Figure 3.7.

Data collection

For the first few exercises, we will need a simple dataset. Feel free to use an anonymous name.

Name

Sex

☒ male ☐ female

Weight

45 kg 65 kg 140 kg

45 55 65 75 85 95 105 115 125 135 140

Age

15 Jahre 35 Jahre 65 Jahre

15 25 35 45 55 65

Height

140 cm 170 cm 220 cm

140 148 156 164 172 180 188 196 204 212 220

Shoe size

30 40 50

30 32 34 36 38 40 42 44 46 48 50

Average travel time to work

0 min 40 min 120 min

0 12 24 36 48 60 72 84 96 108 120

According to your personal estimation, how likely is an economic upturn in 2018?

0% 50% 100%

0 10 20 30 40 50 60 70 80 90 100

☒ submit answer

Figure 3.7: The “datacollection” as an example for a `MiscQuestion` object.

3.2.8 Summary

As we have seen, the question classes serve as templates for creating questions. Some of those classes (like `DfQuestion`) just need single *R* objects to define their appearance while others (for example `MarkdownQuestion` and `MiscQuestion`) are highly customizable. Table 3.3 summarizes the most important features of the 10 classes.

| question class | inherits | description | evaluation |
|-------------------------------|----------------------------|---|--|
| <code>ShinyQuestion</code> | None | Base class | |
| <code>McQuestion</code> | <code>ShinyQuestion</code> | Multiple choice | Distribution of given answers |
| <code>GridQuestion</code> | <code>ShinyQuestion</code> | Assign several answers to several questions | Distribution of given answers |
| <code>FeedbackQuestion</code> | <code>ShinyQuestion</code> | Give feedback to courses | Not graded |
| <code>MiscQuestion</code> | <code>ShinyQuestion</code> | Flexible interface | Depends on the contents |
| <code>MarkdownQuestion</code> | <code>ShinyQuestion</code> | Combine several questions | Depends on the sub-questions |
| <code>RQuestion</code> | <code>ShinyQuestion</code> | <i>R</i> code | Evaluation of resulting <i>R</i> object |
| <code>DfQuestion</code> | <code>RQuestion</code> | Data manipulation tasks | Compare with answer table |
| <code>PlotQuestion</code> | <code>RQuestion</code> | Plotting task | Compare with answer plot |
| <code>LmQuestion</code> | <code>RQuestion</code> | Exercises for linear models | evaluation of the <code>lm</code> object |

Table 3.3: Summary for the question classes in `tguishiny`.

3.3 OTHER CLASSES

Beside the question classes, there are two more `R6` classes defined in the `tgui shiny` package. This section will explain the purpose of those classes.

3.3.1 *SqlHandler*

The `tguiApp` uses *SQL* for persistent data storage. To ensure a clean connection between the app and an *SQL* database, the `SqlHandler` class has been developed. It wraps for the functionalities provided by the `RSQLite` (Müller et al., 2017) package. Member functions for this class include `getTable(tablename)`, `writeTable(tablename)`, `listTables()` and wrappers to other usual database operations.

This class internally keeps track of which tables in the *SQL* database have been changed which allows importing tables as observable objects. To be more clear: whenever a table changes, a counter will be increased. This counter then orders the shiny-framework to update views that depend on the tables.

By default, `SqlHandler` will store tables in the random access memory (*RAM*) but when invoked via `tguiApp`, the tables will be saved as a *SQL* database file for persistent storage. It is important to note that shiny does not provide persistent storage facilities by default (Attali, 2017).

The `tguiApp` function uses a single `SqlHandler` object for all questions and all sessions. The corresponding path of its database file can be specified with the `db_path` argument.

```
tguiApp(db_path = "some/local/path.db")
```

3.3.2 *QuestionList*

This class wraps the data-structure `list(q1, q2, ..., qN)` where all elements of the list are `ShinyQuestion` objects. It includes vectorized getter and setter functions for question properties.

The class also provides a functionality to source several questions from a directory simultaneously, which is used by `tguiApp` to process the `questions_path` argument.

The member-function `attributes_df` is used to build the ui in `tguiApp` depending on the question-pool provided.

The following code listing shows the most important functionalities of the `QuestionList` class. Questions can be added with the `add_question` method and the contents can be shown with `attributes_df`.

```

l = QuestionList$new()

q1 = McQuestion$new("choice", 1, name = "name1")
q2 = ShinyQuestion$new(name = "name2", category = "category2")

l$add_question(q1)
l$add_question(q2)
l$attributes_df()

##      category              id name      class
## 1          bfb2bb17630f92d18a70 name1    McQuestion
## 2 category2 16f2f6e1a12a1e15b698 name2 ShinyQuestion

```

3.4 TEACHER GUIDE

This section will cover some instructions on how to use and customize `tguishiny` as a teacher.

3.4.1 Developing new questions

The `show_interactive` method should be all you need to create and test new questions. Documentation and examples concerning the question classes can be found in the help pages.

```

?ShinyQuestion
?McQuestion
?RQuestion

```

A help page is available for each class introduced in Section 3.2.

3.4.2 Adding new questions to the `tguiApp`

The cleanest way is to create a custom question pool by mimicking the structure of a folder called `questions` in the `tguishiny` installation folder. The path of this folder can be obtained with `find.package`.

```

file.path(find.package("tguishiny"), "app", "questions")
## "C:/Program Files/R/R-3.4.1/library/tguishiny/app/questions"

```

Then you can provide the path of your personal questions folder as an argument for `tguiApp`.

```

tguiApp(questions_path = "path/to/questions")

```

This should replace the default question-pool with the questions in your folder. Make sure that all question files *return* objects derived from the `ShinyQuestion` class. For example:

```
# file: questions/1_myCategory/1_myQuestion.R
McQuestion$new(
  id = "some_unique_id",
  choices = c("first", "second", "third"),
  correct = 1
)
```

Explicitly providing an `id` is currently necessary for a persistent storage of answers.⁴ The `id` will be used by *HTML*, therefore spaces and special characters should not be part of it. Also, each `id` must be unique inside a `questions` folder.

3.4.3 Deploying *tguishiny* via *shiny-server*

Once you have `shiny-server` installed, create a new folder that is served by `shiny-server` and include a *R* script with the name `app.R` in that folder. This script should look as follows.

```
# file: app.R
library(tguishiny)

tguiApp(questions_path = 'path/to/questions',
        db_path = 'path/to/database.db')
```

Both arguments `questions_path` and `db_path` are optional but `db_path` should almost always be set. The database file will be created if it does not exist already.

Using `shiny-server` with `tguishiny` is highly recommended since this will give several users access to the app. In order to make the app portable, relative paths inside the app folder should be used. For example `questions = 'questions'` and `db_path = 'database.db'`.

⁴If the app becomes idle because no user is connected and then becomes active again, the `id` is used to match previously submitted answers to questions.

Theoretical background on examples

This chapter will present some statistical background to be used in Chapter 5, where a demonstration of selected `tguishiny` examples is conducted. This also simulates a class situation, where typically theoretical concepts are explained in detail before `tguishiny` is used for exercises.

4.1 COMPOSITIONAL PRINCIPAL COMPONENT ANALYSIS

This section is mainly based on Jolliffe (2002) where results of Aitchison (1983) are summarized in Chapter 13. We will define the “ordinary” *PCA* method and discuss why that method not well suited for compositional data (like the election data in Chapter 5).

4.1.1 The ordinary principal component analysis

A standard *PCA* uses an *orthogonal transformation* $\mathbf{\Gamma}$ of a data matrix $\mathbf{X} \in \mathbb{R}^{n \times D}$, such that the transformed variables $\mathbf{L} = (\mathbf{X} - \mathbf{1M}(\mathbf{X})')\mathbf{\Gamma}$ are uncorrelated and satisfy the following condition.

$$\mathbb{V}(\mathbf{L}) = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n), \lambda_1 > \lambda_2 > \dots > \lambda_n$$

Here, $\lambda_1, \dots, \lambda_n$ denote the eigenvalues of the covariance matrix $\mathbb{V}(\mathbf{X})$. $\mathbf{M}(\mathbf{X})$ denotes the vector of column-wise means of \mathbf{X} and $\mathbf{1}$ denotes the vector of n ones.

The underlying idea behind this method is to “reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much possible of the variation present in the data set.” (Jolliffe, 2002).

The so-called *loadings* $\mathbf{\Gamma}$ are calculated as a matrix containing all eigenvectors of the *covariance matrix* of \mathbf{X} .

$$\mathbb{V}(\mathbf{X}) = \mathbf{\Gamma D \Gamma}^T, \mathbf{\Gamma}, \mathbf{D} \in \mathbb{R}^{D \times D} \quad (4.1)$$

The scores then follow directly from the transformation equation.

$$\mathbf{L} = (\mathbf{X} - \mathbf{1M}(\mathbf{X})')\mathbf{\Gamma} \quad (4.2)$$

Like many other statistical methods, the *PCA* model assumes that \mathbf{X} are measurements of a multivariate normal distribution.

4.1.2 Compositional data

Compositional data represent multivariate observations where the relevant information is contained in the ratios between the variables (Kynčlová et al., 2016). We will therefore consider a data matrix \mathbf{X} satisfying the following constraint.

$$x_{i,1} + x_{i,2} + \dots + x_{i,p} = k_i, \quad i = 1, \dots, n \quad (4.3)$$

Equation 4.3 introduces “a bias towards negative values among the correlations” (Jolliffe, 2002). It is therefore necessary to use an adapted version of the “classical” *PCA* procedure when dealing with compositional data.

One adaption is presented in Aitchison (1983). It uses a the centered logratio (*clr*) transformation on the columns of \mathbf{X} and applies the “classical” *PCA* on the transformed matrix. A *clr* transformation uses logarithms and re-centers the data such that the sum of all \mathbf{y}_j is zero.

$$y_{i,j} = \ln(x_{i,j}) - \frac{1}{p} \sum_{k=1}^p \ln(x_{i,k}), \quad j = 1, \dots, D, \quad i = 1, \dots, n \quad (4.4)$$

Note that frequencies of zero are not allowed because of the logarithm. Equation 4.4 can also be written in terms of the geometric mean as

$$y_{i,j} = \ln \left(\frac{x_{i,j}}{\sqrt[p]{\prod_{k=1}^p x_{i,k}}} \right).$$

This transformation preserves the distances between objects and therefore the *PCA* can be applied to the transformed data (Kynčlová et al., 2016).

4.2 ROBUST ESTIMATORS AND THE ILR TRANSFORMATION

This section will give insights about *robust* estimation of means and covariance matrices as well as applications in *PCA*. Also, the *ilr* transformation will be presented which is a necessary alternative to the *clr* transformation in order to use the robust estimators presented here.

After that, the *PCA* algorithm in the *R* package `robCompositions` (Templ et al., 2011) which implements both *robust* estimators and the *ilr* transformation is outlined.

4.2.1 The Minimum Covariance Determinant (MCD) method

The minimum covariance determinant (MCD) method of Rousseeuw is a highly robust estimator of multivariate location and scatter. Its objective is to find h observations (out of n) whose covariance matrix has the lowest determinant. (Rousseeuw and Driessen, 1999)

By using subsets, this method removes outliers from a data matrix since outliers usually increase the determinant of the covariance matrix. h is a fixed number and usually chosen as a percentage of n . Note that a subset minimizing the variance always exists since the number of potential subsets is finite.

Consider a data matrix $X \in \mathbb{R}^{n \times D}$. Given the minimum determinant subset

$S \subseteq \{1, \dots, n\}$, the robust location and scatter estimators are

$$\mathbf{M}_{\text{rob}}(\mathbf{X}) = \frac{1}{h} \sum_{i \in S} \mathbf{X}_{i,\cdot}, \quad (4.5)$$

$$\mathbf{V}_{\text{rob}}(\mathbf{X}) = \frac{C_{D,n,h}}{h-1} \sum_{i \in S} (\mathbf{X}_{i,\cdot} - \mathbf{M}_{\text{rob}}(X))(\mathbf{X}_{i,\cdot} - \mathbf{M}_{\text{rob}}(X))^T \quad (4.6)$$

where $\mathbf{X}_{i,\cdot}$ is the i -th row of the data matrix and $C_{D,n,h}$ is a factor for consistency.

The *MCD* method relies on full rank data matrices (Kynčlová et al., 2016). If X is not of full rank and if n is bigger than the number of variables D , then the columns of the data matrix are linearly dependent. This will then also hold true for all the subsets S . Therefore the covariance determinant is zero for all subsets and there is no reasonable way to determine an “optimal” subset.

4.2.2 The isometric logratio (*ilr*) transformation

clr transformed data does not provide full rank data matrices. Let $x \in \mathbb{R}^D$ be an observation of a compositional variable and $y \in \mathbb{R}^D$ be the *ilr* transformation of x , then we have

$$\sum_{i=1}^D y_i = \sum_{i=1}^D \ln \left(\frac{x_i}{\sqrt[D]{\prod_{k=1}^D x_k}} \right) = \ln \left(\prod_{i=1}^D \frac{x_i}{\sqrt[D]{\prod_{k=1}^D x_k}} \right) = \ln(1) = 0.$$

Therefore the *MCD* method is not applicable. According to Filzmoser et al. (2009), several other robust estimators are also not suited for *clr* transformed data.

“ The main disadvantage of this [the *clr*] transformation is that the resulting data are collinear because $\sum_{i=1}^D y_i = 0$. Methods that rely on full rank data matrices, like standard robust covariance estimators [...] will thus not be applicable. (Filzmoser et al., 2009)

”

The *isometric logratio* (*ilr*) transformation however maps x into a $D - 1$ dimensional space and will create full rank matrices. For a single observation, the transformation is defined as follows.

$$y_j^{\text{ilr}} = \frac{j}{j+1} \ln \left(\frac{\sqrt[j]{\prod_{k=1}^j x_k}}{x_{j+1}} \right), \quad j = 1, \dots, D-1 \quad (4.7)$$

The *ilr* transformation is in fact a linear and orthonormal transformation of the *clr* transformed data. This means, that the isometry of the *clr* transformation is preserved (Filzmoser et al., 2009).

One disadvantage of this method is that results calculated in the *ilr* space have to be back-transformed into the *clr* space for interpretation.

4.2.3 Outline of the PCA algorithm in robCompositions

Given a data matrix $X \in \mathbb{R}^{n \times D}$, the *ilr* transformation Y according to (4.7) is calculated. Then the robust estimators $\mathbf{M}_{\text{rob}}(\mathbf{Y})$ and $\mathbf{V}_{\text{rob}}(\mathbf{Y})$ of the *ilr* transformed data are determined using the *MCD* method in (4.5) and (4.6). For the calculation of robust loadings, an eigenvalue decomposition of the robust scatter estimator according to (4.1) and (4.2) is performed.

$$\mathbf{V}_{\text{rob}}(\mathbf{Y}) = \mathbf{\Gamma}_{\text{ilr}} \mathbf{D}_{\text{ilr}} \mathbf{\Gamma}_{\text{ilr}}^T, \quad \mathbf{L}_{\text{ilr}} = (\mathbf{X} - \mathbf{M}_{\text{rob}}(\mathbf{X})) \mathbf{\Gamma}$$

The results $\mathbf{\Gamma}_{\text{ilr}}$ and \mathbf{L}_{ilr} are then transformed back into the *clr* space.

4.3 COMPOSITIONAL BILOTS

Biplots visualize loadings and scores of a principal component analysis (Kynčlová et al., 2016). They use data from the first few rows of the scores (\mathbf{L}) and the first few columns of loadings ($\mathbf{\Gamma}$). The contents of the matrices are represented as dots for scores and arrows for loadings.

In most cases, a two-dimensional plot is used showing the first and second component (the first and second row of \mathbf{L} as well as the first and second column $\mathbf{\Gamma}$). These components correspond to the biggest eigenvalues of the covariance matrix and will therefore give insights about the most potent latent factors.

Compositional biplots display the same information as ordinary biplots, but $\mathbf{\Gamma}$ and \mathbf{L} are calculated using a compositional *PCA*.

Compositional biplots for *clr* transformed data have been introduced in Aitchison and Greenacre (2002). Kynčlová et al. (2016) adapts this idea for *ilr* transformed data and gives guidelines on how to interpret both *ilr biplots* and *clr biplots*.

Figure 4.1 shows an example of an *ilr biplot* created with the `robCompositions` package. The data for this plot contains results of an election for all federal states of Germany. The entries $x_{i,j}$ of the underlying dataset denote the absolute number of votes for party j in a state i .

| | CDU_CSU | SDP | GRUENE | FDP | DIE_LINKE | other_parties |
|----|---------|---------|--------|--------|-----------|---------------|
| SH | 638756 | 513725 | 153137 | 91714 | 84177 | 146781 |
| HH | 285927 | 288902 | 112826 | 42869 | 78296 | 82009 |
| NI | 1825592 | 1470005 | 391901 | 185647 | 223935 | 348180 |
| HB | 96459 | 117204 | 40014 | 11204 | 33284 | 31247 |
| NW | 3776563 | 3028282 | 760642 | 498027 | 582925 | 851718 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Table 4.1: Number of votes for an election in Germany 2013.

Interpretation

In the following, we will denote the columns of $\mathbf{\Gamma}$ as γ_j and the rows of \mathbf{L} and \mathbf{X} as \mathbf{l}_i and $\mathbf{x}_{i,\cdot}$. The function g denotes the geometric mean and \mathbb{V}_i the variance

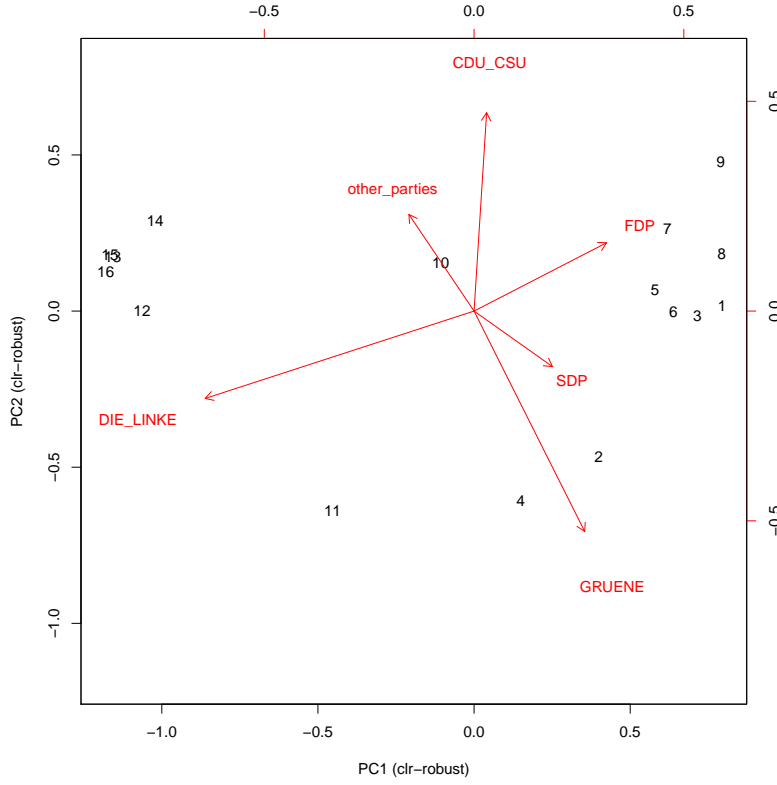


Figure 4.1: A *biplot* about an election in Germany from 2013. Produced with `robCompositions`

with respect to i (the observations). The plot can be interpreted in the following manner (see Kynčlová et al. (2016) for details).

- (i) The length of the arrows represent $\|\gamma_j\|_2^2 \approx \frac{D}{D-1} \mathbb{V}_i(\ln(x_{i,j}/g(x_{i,\cdot})))$.
- (ii) The distances between the vertices are $\|\gamma_j - \gamma_k\|_2^2 \approx \frac{D}{D-1} \mathbb{V}_i(\ln(x_{i,j}/x_{i,k}))$.
- (iii) The inner product of the arrows and points gives

$$\langle \mathbf{l}_i, \gamma_j \rangle \approx \sqrt{\frac{D}{D-1}} \ln \left(\frac{x_{i,j}}{g(x_{i,\cdot})} \right).$$

These results can now be applied to the *biplot* shown in Figure 4.1. In this case, i denotes the federal states of Germany and j denotes the different parties.

- (i) The party *DIE LINKE* has the longest arrow. Thus the variance of

$$\frac{x_{\text{state}, \text{DIE_LINKE}}}{g(x_{\text{state}, \cdot})}$$

is bigger for this party than for all other parties. In other words, “the relative variability of the obtained votes differs a lot among all observed states” (Kynčlová et al., 2016). The shortest arrow belongs to the party *SDP* where the relative number of votes was relatively stable with respect to the federal states.

- (ii) The arrows for the parties *DIE LINKE* and *FDP* have vertices which are very far apart. This means, that the relative number of votes

$$\frac{x_{\text{state}, \text{DIE_LINKE}}}{x_{\text{state}, \text{FDP}}}$$

show a relatively high proportional variability with respect to the federal states of Germany. On the other hand, the pair *GRÜNE* and *SDP* have a relatively stable ratio of votes.

- (iii) The arrow of the party *FDP* almost points to state 7 (Rheinland-Palatinate). Consequently, the relative number of votes

$$\frac{x_{7, \text{FDP}}}{g(x_7, \cdot)}$$

is high compared to most other states. It can be verified, that the party *FDP* in fact had the strongest results in the states 8, 1, 6 and 7.

Demonstration examples in tguishiny

In this chapter, we will talk about one exercise regarding the general election results of autumn 2017 in Austria. The exercise begins with questions about exploitative data analysis using the classes `PlotQuestion` and `DfQuestion`.

At the end of the exercise, the theory derived in Chapter 4 will be used to interpret the results from a *compositional principal component analysis*. The exercise has been implemented with the `MarkdownQuestion` class.

5.1 DATA

For the most part, the data used in this exercise is a single `dataframe` which was imported from <http://data.gv.at>. It includes counts of the votes for each of the participating parties in 2286 electoral districts. 164 of those districts represent postal votes.

The data is summarized on a state level in Table 5.1. Only the six biggest parties have been included since the remaining fractions were not eligible in all federal states.

| state | eligible | submitted | SPÖ | ÖVP | FPÖ | GRÜNE | NEOS | PILZ |
|---------------|----------|-----------|--------|--------|--------|-------|-------|-------|
| Burgenland | 232740 | 196577 | 64070 | 63858 | 49127 | 3932 | 5603 | 5529 |
| Carinthia | 440207 | 345354 | 99923 | 91458 | 108215 | 8249 | 14692 | 12298 |
| Lower Austria | 1288802 | 1092339 | 267348 | 384279 | 280011 | 29619 | 51815 | 44767 |
| Upper Austria | 1103664 | 902801 | 246201 | 280595 | 239444 | 32792 | 42556 | 32772 |
| Salzburg | 395723 | 319159 | 70191 | 119072 | 77120 | 12714 | 17985 | 11149 |
| Styria | 969653 | 774037 | 192738 | 241917 | 225990 | 21430 | 38341 | 29980 |
| Tyrol | 543116 | 414913 | 85650 | 158092 | 102610 | 18367 | 23537 | 15746 |
| Vorarlberg | 272909 | 197125 | 34961 | 67982 | 47837 | 14137 | 17666 | 5805 |
| Vienna | 1154184 | 878574 | 300664 | 188273 | 186088 | 51398 | 56323 | 65498 |

Table 5.1: Summary of the election dataset on a state level including postal votes.

The principal component analysis also uses a second `dataframe` shown in Table 5.2, which gives information about unemployment, income and the gross regional product for each state. This data has been imported from <http://de.statista.com> and <http://www.statistik.at>.

Note, that Table 5.2 is not summarized. This data is only available on a state level.

5.2 DFQUESTIONS

As we can see, Table 5.1 uses a *wide format* regarding the parties. Since `ggplot` works best with data in *long format*, some of the `DfQuestion`s require the

| state | unemp. | income | GRP |
|---------------|--------|--------|-------|
| Burgenland | 6.8% | 22200 | 27500 |
| Carinthia | 8.4% | 21500 | 33300 |
| Lower Austria | 7.5% | 23100 | 32500 |
| Upper Austria | 5.0% | 22300 | 40300 |
| Salzburg | 4.7% | 22800 | 46100 |
| Styria | 6.1% | 21900 | 35400 |
| Tyrol | 4.8% | 22000 | 42000 |
| Vorarlberg | 5.5% | 23500 | 42300 |
| Vienna | 12.3% | 21500 | 47700 |

Table 5.2: Additional data for each state.

students to transform the data into a *long format*. Figure 5.1 shows one exercise of these questions.

Create the following dataframe and save it as `bigparties_long`

| | state | party | votes |
|---|---------------|-------|--------|
| 1 | Burgenland | SPÖ | 64070 |
| 2 | Carinthia | SPÖ | 99923 |
| 3 | Lower Austria | SPÖ | 267348 |
| 4 | Upper Austria | SPÖ | 246201 |
| 5 | Salzburg | SPÖ | 70191 |

Previous 1 2 3 4 5 6 Next

```

1 bigparties_long <- bigparties %>%
2   subset(select =) %>%
3   gather()

```

[</> evaluate](#) [submit answer](#)

console output

There is no output available yet. Please click 'submit'

Figure 5.1: A `DfQuestion` asking students to cast the election dataset from a *wide format* to a *long format*.

The corresponding input dataset is called `bigparties` and has the same structure as Table 5.1.

Knowing how to cast tables from *long format* to *wide format* is a commonly required skill that students learn in this exercise. Teachers can see all the code submitted by students and discuss common syntax errors afterwards.

The default code in the question suggests using function `gather` from the

`tidyr` package, although other alternatives like `reshape2::melt` are also possible to answer the question.

The following code will be considered a correct answer for the question shown in Figure 5.1.

```
bigparties_long <- bigparties %>%
  subset(select = c(state, SPÖ:FPÖ)) %>%
  gather(party, votes, -state)
```

Another question of the class `DfQuestion` asks the users to classify the data on a regional level (2286 regions) based on the region *ID* (not included in Table 5.1). This question is useful to teach students how to use the function `mutate` from the `dplyr` package.

5.3 PLOTQUESTIONS

This section will give some insight about the `PlotQuestion`s included in the exercise. First, there are two questions regarding bar charts of the election results. Figure 5.2 shows one of those questions. The question requires users to apply the functions `ggplot` and `geom_col` appropriately. The code for using a custom color palette (corresponding to the party-colors) is hinted before the question.

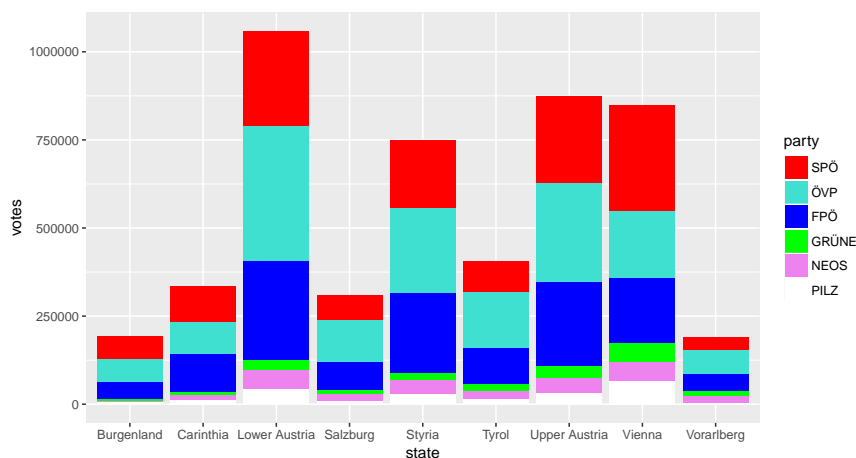
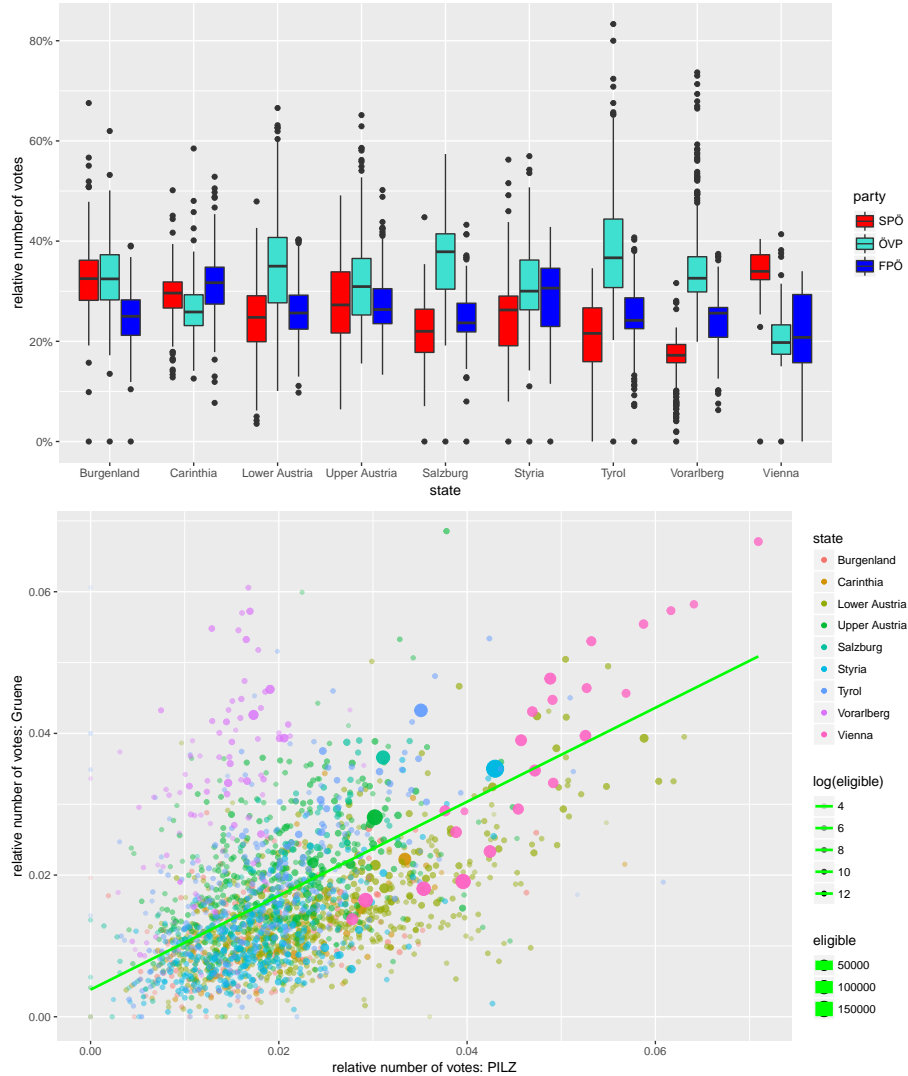


Figure 5.2: A stacked bar chart showing the absolute number of votes for the six biggest parties by state.

Students will learn to both create and interpret stacked bar charts using the `ggplot2` package. Teachers can then address common misconceptions by investigating the *R* code submitted by the students.

Further examples of `PlotQuestion`s included in the exercise are box plots and several scatter plots with regression lines. Some of those can be seen in

Figure 5.3.

**Figure 5.3:** Multiple plots related to the election results.

For the second graph in Figure 5.3, students are required to customize the colors, point sizes and alpha values (transparency) in the graph, which is a useful skill to master. The question will include a lot of hints for this fine tuning process. Hints are given both in textual form and in the default user script for the question.

It is important to note, that all graphs in this chapter are constructed with the `ggplot2` package, for the reasons discussed in Subsection 3.2.4.

5.4 AN INTERACTIVE BOX PLOT

One part from this exercise features an interactive box plot followed by a multiple choice question which asks the students to interpret the plots appropriately. Exercises like this one are useful to showcase which informations can and cannot be obtained from statistical graphs.

The interactive graph shows the relative number of votes for a party depending on different categorical variables. Both the party and the categorical variables can be chosen by the student via a dropdown menu.



Figure 5.4: An interactive boxplot followed by a multiple choice question.

5.5 COMPOSITIONAL PRINCIPAL COMPONENT ANALYSIS

The very end of the exercise shows an interactive *PCA*. The analysis has been conducted with the package `robCompositions` (Templ et al., 2011) and can be seen in Figure 5.5.

The *PCA* widget is followed by a `McQuestion` that asks students to interpret



Figure 5.5: A graphical output of the *PCA* for the election exercise. Several control elements can be used to alter the graph. The other tabs (`print` , `plot` , etc.) will display other output based on the `pcaCoDa` object.

the results. In order to answer all the questions, students will have to navigate inside the widget and have some fundamental insights about the statistical methods used.

☐ The biplot suggests that the party SPÖ had a big variance in the relative amount of votes due to its long arrow.
 ☐ The big distance between the vertices of SPÖ and ÖVP suggest that both parties got a big amount of votes.
 ☐ The fact that the ÖVP arrow points away from most areas in Vienna suggests that the party had less success there than in other federal states.
 ☐ Since the arrows 'postal' and FPÖ show in opposite directions, areas with a high number of postal votes showed less support for the

Figure 5.6: The multiple choice question belonging to the *biplot/PCA* exercise.

Note that students will hear a talk about the *PCA* method used prior to answering those questions. Teachers will be able to see, which mistakes are most common in answering the multiple choice question. Subsequently, answer choices that have been marked incorrectly by most students can be discussed further by the teacher.

Summary

This chapter will present the most important features that have been implemented in the `tguishiny` project and finish off with a list of possible improvements.

6.1 ARCHIEVEMENTS

The aim of the `tguishiny` project was to create a well documented, easy-to-use package for creating teaching resources in shiny. This goal has been reached to a certain extent.

Documentation

The documentation of the `tguishiny` package follows the standard-approach in *R* packaging, which is to use *R* documentation (`.Rd`) files. So far, the package has successfully been used by students of the *ZHAW* university using the documentation as a resource.

R inputs

As already mentioned in Section 1.5, there are only a limited number of teaching products that allow *R* inputs as part of student's answers. The `tguishiny` project not only provides this feature, but also gives students a way to view outputs generated by their answer before submitting it.

Immediate feedback

Teachers will immediately be notified whenever a student submits a new answer, which is again a feature missing in a lot of common teaching tools provided by *R*. The feedback system uses the observer design pattern provided by `shiny`.

Persistent storage

Answers of students are automatically stored in a *SQL* database and will therefore be accessible for later use.

Login logic

A deployed version of the `tguiApp` requires users and teachers to fill a login form in order to have access to the questions. Teachers can add new users (teacher and student accounts) inside the *GUI*.

Grading answers

Each question class has a separate logic to assign scores for answers. The scoring system reaches from zero points (wrong) to one point (right) and might take values in between these two to indicate a partially right answer.

6.2 EXPANDABILITY OF THE PROJECT

There are certain directions this project could take depending on the preferred features that will be desired in the future. Since the project will be used by the *ZHAW* university as well as the *Statistics Austria* institute in the future, further developments will be undertaken.

This section will present some possible enhancements and give a short description how they could be implemented.

UI improvements

Libraries like `shinydashboard` (Chang and Borges Ribeiro, 2017), `flexdashboard` (Allaire, 2017a) and `shinywidgets` (Perrier and Meyer, 2018) can be used to enhance the UI of the project and also make the ui code more instructive.

So far, the focus for the project was certainly more weighted towards back end stability and server-side development. Looking at the examples in `shinyWidgetsGallery` inspires for some quick and powerful additions that could be added in the future.

More question-containers

Currently, the class `MarkdownQuestion` is the only one that supports sub-questions. Having more alternatives when it comes to this could certainly be a welcome feature for question authors. A prompt-like container that automatically navigates to the next sub-question when the current one is answered could be a feasible and useful addition.

UI for creating new questions

A graphical user interface that allows creation of new questions could encourage teachers to get in touch with the project more easily. Such a *GUI* could be implemented with `shiny` as part of the `tguiApp`.

However, the current logic requires the source code of every question to be present whenever the app gets invoked. Persistent storage is only implemented via the *SQL* interface `RSQLite` which can't save *R* objects. Implementing this feature would therefore require a more “reactive” version of the `QuestionList` class.

New question classes

So far, quite a view classes are available to create new `ShinyQuestion` objects. Some of them have been inspired by certain *JavaScript* libraries that were already ported into *R*. Following this approach could lead to some interesting new question classes.

For example, the `dygraphs` (Vanderkam et al., 2017) package might give access to a `TimeSeriesQuestion` class which could be used in courses that are focused on time series analysis and prediction models. Similar concepts could be used with `leaflet` (Cheng et al., 2017) and `d3Network` (Gandrud, 2015).

Exams

Since questions from the `tguishiny` package are graded automatically upon submission, the project can be used for the creation of exams. Providing some additional features like a *countdown* for the exam time and an automatically created *report* about student's scorings in the exams could be a handy addition to the project.

Security

Since user submitted *R* code is run on the server hosting the `tguishiny` project, improvements in security are an important feature to consider in the future. As for now, it is possible for students to manipulate files in the app's directory which can lead to serious issues.

A rather simple improvement in that regard would be to create a *blacklist* of commands that are not allowed in the scripts submitted by the students.

Bibliography

- J. Aitchison. Principal Component Analysis of Compositional Data. *Biometrika*, 70(1):57–65, 1983. doi:10.2307/2335943.
- J. Aitchison and M. Greenacre. Biplots of Compositional Data. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 51(4):375–392, 2002. doi:10.1111/1467-9876.00275.
- J. Allaire. *flexdashboard: R Markdown Format for Flexible Dashboards*, 2017a. URL <https://cran.r-project.org/package=flexdashboard>. R package version 0.5.
- J. Allaire. *learnr: Interactive Tutorials for R*, 2017b. URL <https://cran.r-project.org/package=learnr>. R package version 0.9.
- J. Allaire, J. Cheng, Y. Xie, J. McPherson, W. Chang, J. Allen, H. Wickham, A. Atkins, R. Hyndman, and R. Arslan. *rmarkdown: Dynamic Documents for R*, 2017. URL <https://cran.r-project.org/package=rmarkdown>. R package version 1.6.
- D. Attali. *shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds*, 2016. URL <https://cran.r-project.org/package=shinyjs>. R package version 0.9.
- D. Attali. Persistent Data Storage in Shiny Apps. *shiny.rstudio.com*, 2017. <http://shiny.rstudio.com/articles/persistent-data-storage.html>.
- W. Chang. *R6: Classes with Reference Semantics*, 2017. URL <https://cran.r-project.org/package=R6>. R package version 2.2.2.
- W. Chang and B. Borges Ribeiro. *shinydashboard: Create Dashboards with 'Shiny'*, 2017. URL <https://cran.r-project.org/package=shinydashboard>. R package version 0.6.1.
- W. Chang, J. Cheng, J. Allaire, Y. Xie, and J. McPherson. *shiny: Web Application Framework for R*, 2017. URL <https://cran.r-project.org/package=shiny>. R package version 1.0.3.
- J. Cheng. Modularizing Shiny App Code. *shiny.rstudio.com*, 2017. <http://shiny.rstudio.com/articles/modules.html>.
- J. Cheng, B. Karambelkar, and Y. Xie. *leaflet: Create Interactive Web Maps with the JavaScript 'Leaflet' Library*, 2017. URL <https://cran.r-project.org/package=leaflet>. R package version 1.1.0.
- G. de Cillia, M. Templ, and B. Meindl. *tguishiny: Blended Learning Tools for Statistics*, 2018. URL <https://bitbucket.org/matthias-da/tguishiny/>. R package version 1.0.0.

- N. de Jong, M. Savin-Baden, A. M. Cunningham, and D. M. L. Verstegen. Blended Learning in Health Education: Three Case Studies. *Perspectives on Medical Education*, 3(4):278–288, 2014. doi:10.1007/s40037-014-0108-1.
- C. Dichev, D. Dicheva, G. Agre, and G. Angelova. Current Practices, Trends and Challenges in K-12 Online Learning. *Cybernetics and Information Technologies*, 12(3):91–110, 2013. doi:10.2478/cait-2013-0028.
- G. Dinges and M. Templ. Motivation zur Statistik - Computergestützt Lernen in der Statistik Austria. *Austrian Journal of Statistics*, 38(1):3–16, 2009. doi:10.17713/ajs.v38i1.256.
- G. Dinges, A. Kowarik, B. Meindl, and M. Templ. Moderne Wege der Wissensvermittlung; Statistikunterricht interaktiv. *Statistische Nachrichten*, pages 919–923, September 2011a. doi:10.17713/ajs.v38i1.256.
- G. Dinges, A. Kowarik, B. Meindl, and M. Templ. An Open Source Approach for Modern Teaching Methods: The Interactive TGUI System. *Journal of Statistical Software*, 39(7), 2011b. doi:10.18637/jss.v039.i07.
- P. Filzmoser, K. Hron, and C. Reimann. Principal Component Analysis for Compositional Data with Outliers. *Environmetrics*, 20(6):621–632, 2009. doi:10.1002/env.966.
- C. Gandrud. *d3Network: Tools for creating D3 JavaScript network, tree, dendrogram, and Sankey graphs from R*, 2015. URL <https://cran.r-project.org/package=d3Network>. R package version 0.5.2.1.
- B. Grün and A. Zeileis. Automatic Generation of Exams in R. *Journal of Statistical Software*, 29(10):1–14, 2009. doi:10.18637/jss.v029.i10.
- B. Güzera and H. Caner. The Past, Present and Future of Blended Learning: an in Depth Analysis of Literature. *Procedia - Social and Behavioral Sciences*, 116(1):4596–4603, 2014. doi:10.1016/j.sbspro.2014.01.992.
- M. B. Horn, H. Staker, and C. Christensen. *Is K-12 Blended Learning Disruptive? An Introduction of the Theory of Hybrids*. Clayton Christensen Institute, 2013. URL <https://www.christenseninstitute.org/publications/hybrids/>.
- J. Horner. *rApache: Web Application Development with R and Apache.*, 2013. URL <http://www.rapache.net/>.
- I. Jolliffe. *Principal Component Analysis*. Springer Verlag, New York, March 2002. doi:10.1007/b98835.
- T. Kluyver, B. Ragan-Kelley, et al. Jupyter Notebooks – a Publishing Format for Reproducible Computational Workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016. doi:10.3233/978-1-61499-649-1-87.
- S. Kross, N. Carchedi, B. Bauer, and G. Grdina. *swirl: Learn R, in R*, 2017. URL <https://cran.r-project.org/package=swirl>. R package version 2.4.3.

- P. Kynčlová, P. Filzmoser, and K. Hron. Compositional Biplots Including External Non-Compositional Variables. *Statistics*, 50(5):1132–1148, 2016. doi:10.1080/02331888.2015.1135155.
- Q. Liu, W. Peng, F. Zhang, R. Hu, Y. Li, and W. Yan. The Effectiveness of Blended Learning in Health Professions: Systematic Review and Meta-Analysis. *Journal of Medical Internet Research*, 18(1):e2, 2016. doi:10.2196/jmir.4807.
- K. McCutcheon, M. Lohan, M. Traynor, and D. Martin. A Systematic Review Evaluating the Impact of Online or Blended Learning vs. Face-to-Face Learning of Clinical Skills in Undergraduate Nurse Education. *Journal of Advanced Nursing*, 71(2):255–270, 2015. doi:10.1111/jan.12509.
- K. Müller, H. Wickham, D. A. James, and S. Falcon. *RSQLite: 'SQLite' Interface for R*, 2017. URL <https://cran.r-project.org/package=RSQLite>. R package version 2.0.
- R. T. Osguthorpe and C. R. Graham. Blended Learning Environments: Definitions and Directions. *Quarterly Review of Distance Education*, 4(3):227–233, 2003. ISSN 1528-3518. URL <https://www.learntechlib.org/p/97576>.
- V. Perrier and F. Meyer. *shinyWidgets: Custom Inputs Widgets for Shiny*, 2018. URL <https://github.com/dreamRs/shinyWidgets>. R package version 0.3.6.900.
- P. J. Rousseeuw and K. V. Driessen. A Fast Algorithm for the Minimum Covariance Determinant Estimator. *Technometrics*, 41(3):212–223, 1999. doi:10.1080/00401706.1999.10485670.
- RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc., Boston, MA, 2016. URL <http://www.rstudio.com/>.
- C. Sievert, C. Parmer, T. Hocking, S. Chamberlain, K. Ram, M. Corvellec, and P. Despouy. *plotly: Create Interactive Web Graphics via 'plotly.js'*, 2017. URL <https://cran.r-project.org/package=plotly>. R package version 4.6.0.
- M. Templ, K. Hron, and P. Filzmoser. *robCompositions: an R-package for Robust Statistical Analysis of Compositional Data*. John Wiley and Sons, 2011. doi:9781119976462.ch25. URL <https://cran.r-project.org/package=robCompositions>.
- Trestle Technology, LLC. *shinyAce: Ace Editor Bindings for Shiny*, 2016. URL <https://cran.r-project.org/package=shinyAce>. R package version 0.2.1.
- D. Vanderkam, J. Allaire, J. Owen, D. Gromer, P. Shevtsov, and B. Thieurmél. *dygraphs: Interface to 'Dygraphs' Interactive Time Series Charting Library*, 2017. URL <https://cran.r-project.org/package=dygraphs>. R package version 1.1.1.4.
- H. Wickham. *evaluate: Parsing and Evaluation Tools that Provide More Details than the Default*, 2016a. URL <https://cran.r-project.org/package=evaluate>. R package version 0.10.
- H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016b. doi:10.1007/978-0-387-98141-3. URL <http://ggplot2.org>.

- Y. Xie. knitr: A Comprehensive Tool for Reproducible Research in R. In V. Stodden, F. Leisch, and R. D. Peng, editors, *Implementing Reproducible Computational Research*. Chapman and Hall/CRC, 2014. ISBN 978-1466561595.
- Y. Xie. *DT: A Wrapper of the JavaScript Library 'DataTables'*, 2016. URL <https://cran.r-project.org/package=DT>. R package version 0.2.
- G. Yu. *dlstats: Download Stats of R Packages*, 2017. URL <https://CRAN.R-project.org/package=dlstats>. R package version 0.1.0.