Keynote lecture about R/exams at eRum 2018 (European R Users Meeting) in Budapest: Slides, video, e-learning, replication materials.

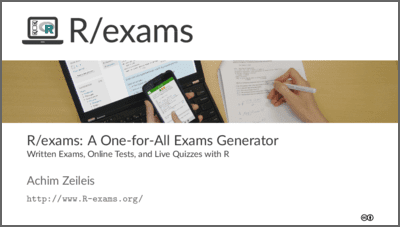


**Keynote lecture at eRum 2018**

[R/exams](http://www.r-exams.org/) was presented in a keynote lecture by Achim Zeileis at [eRum 2018](https://2018.erum.io/), the European R Users Meeting, this time organized by a team around [Gergely Daróczi](https://twitter.com/daroczig) in Budapest. It was a great event with many exciting presentations, reflecting the vibrant R community in Europe (and beyond).

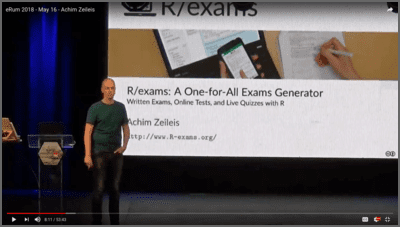
This blog post provides various resources accompanying the presentation which may be of interest to those who did not attend the meeting as well as those who did and who want to explore the materials in more detail.

Most importantly the presentation slides are available in PDF format (under CC-BY):

[](http://www.r-exams.org/assets/posts/2018-05-20-erum2018/slides.pdf)

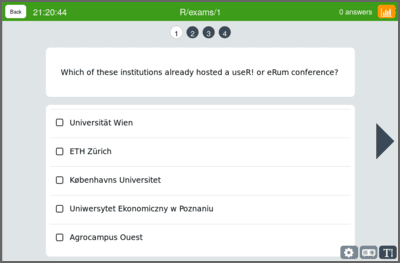
**Video**

The eRum organizers did a great job in making the meeting accessible to those useRs who could not make it to Budapest. All presentations were available in a livestream on YouTube where also videos of all lectures were made available after the meeting (Standard YouTube License):

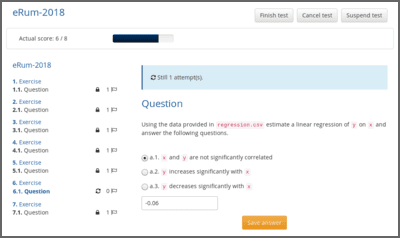
[](https://www.youtube.com/watch?v=NMIzbrklBEE)

**E-Learning**

To illustrate the e-learning capabilities supported by R/exams, the presentation started with a live quiz using the audience response system [ARSnova](https://arsnova.eu/). The original version of the quiz was hosted on the ARSnova installation at Universität Innsbruck. To encourage readers to try out ARSnova for their own purposes, a copy of the quiz was also posted on the official ARSnova server at Technische Hochschule Mittelhessen (where ARSnova is developed under the General Public License, GPL):

[](https://arsnova.eu/mobile/#id/39755874)

The presentation briefly also showed an online test generated by R/exams and imported into [OpenOLAT](https://www.openolat.com/), an open-source learning management system (available under the Apache License). The online test is made available again here for anonymous guest access. *(Note however, that the system only has one guest user so that when you start the test there may already be some test results from a previous guest session. In that case you can finish the test and also start it again.)*

[](https://lms.uibk.ac.at/url/RepositoryEntry/2823520256/CourseNode/97603810083315?guest=true)

**Replication code**

The presentation slides show how to set up an exam using the R package and then rendering it into different output formats. In order to allow the same exam to be rendered into a wide range of different output formats, only single-choice and multiple-choice exercises were employed (see the choice list below). However, in the e-learning test shown in OpenOLAT all exercises types are supported (see the elearn list below). All these exercises are readily provided in the package and also introduced online: deriv/deriv2, fruit/fruit2, ttest, boxplots, cholesky, lm, function. The code below uses the R/LaTeX (.Rnw) version but the R/Markdown version (.Rmd) could also be used instead.

## package

library("exams")

## single-choice and multiple-choice only

choice <- list("deriv2.Rnw", "fruit2.Rnw", c("ttest.Rnw", "boxplots.Rnw"))

## e-learning test (all exercise types)

elearn <- c("deriv.Rnw", "fruit.Rnw", "ttest.Rnw", "boxplots.Rnw",

"cholesky.Rnw", "lm.Rnw", "function.Rnw")

First, the exam with the choice-based questions can be easily turned into a PDF exam in NOPS format using exams2nops, here using Hungarian language for illustration. Exams in this format can be easily scanned and evaluated within R.

# Written Multiple-Choice Exams with R/exams

Step-by-step guide to generating, conducting, scanning, and automatically evaluating large-scale written exams with exams2nops() in R/exams.

## Create exam

The first step in conducting a **written exam** with multiple-choice (and/or single-choice) exercises in R/exams' NOPS format is to create the exam in PDF format. First, we load the R exams package and then simply create a list of exercise file names.

library("exams")

myexam <- list(

"tstat2.Rnw",

"ttest.Rnw",

"relfreq.Rnw",

"anova.Rnw",

c("boxplots.Rnw", "scatterplot.Rnw"),

"cholesky.Rnw"

)

Here, we use a number of schoice and mchoice questions that are directly shipped within the package. In practice, you would use files that you have authored and stored somewhere locally. Above, exercises in .Rnw format are used but all of the examples are also available in .Rmd format, leading to virtually identical output.

Then, we create a small exam with only n = 2 randomly-drawn versions, storing the resulting PDF files (plus metainformation) on the disk in a new output directory nops\_pdf. To customize the exam we assign a different number of points to the different exercises and also show the respective number of points at the beginning of each question.

set.seed(403)

ex1 <- exams2nops(myexam, n = 2,

dir = "nops\_pdf", name = "demo", date = "2015-07-29",

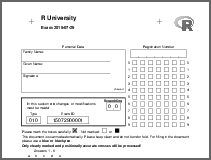
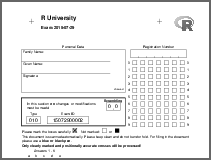
points = c(1, 1, 1, 2, 2, 3), showpoints = TRUE)

A random seed is set to make the generated exams exactly reproducible for you (or ourselves at some point in the future). The output directory now contains three files that were generated.

dir("nops\_pdf")

## [1] "demo.rds" "demo1.pdf" "demo2.pdf"

The two PDF files are the two exams we requested above.

**[](http://www.r-exams.org/assets/posts/2017-09-18-exams2nops/demo1.pdf)** **[](http://www.r-exams.org/assets/posts/2017-09-18-exams2nops/demo2.pdf)**

Furthermore, the metainfromation about the exam (exam IDs, questions, correct and wrong answer alternatives) is stored in a demo.rds file (serialized R data). This crucial for being able to evaluate the exam later on.

## Print PDF files

* A small number of exams can easily be printed on a standard printer. Otherwise simply use a print shop.
* It is recommended not to scale the printout (i.e., without "Fit to printable area") and to staple the exams in the top-left corner.
* By default the PDFs from exams2nops() have a blank second page for duplex printing (without content on the back of the exam sheet). For non-duplex printing simply set duplex = FALSE when creating the PDFs with exams2nops().

## Conduct exam

The exam is conducted as usual. But if you used the possibilities of **dynamic exercises** in R/exams, the risk of cheating is greatly reduced. At the end of the exam you just need to collect the completed exam sheet (first page). Of course, you can also collect the rest of the exam papers (e.g., to keep future students from seeing the exercises). However, an advantage of letting the students keep their exercises reduces the need of having post-exam reviews etc.

## Scan results

Each completed exam sheet has information on the exam ID, student ID, and the checked answers. This needs to be scanned into images (PDF or PNG) and can then be processed by nops\_scan(). Typically, it's easy to use the photocopiers provided by your university to scan all sheets into PDF or PNG files. For example, our university provides us with Canon ImageRunners and the sheet feeder can easily take about 40-50 sheets and render them into a single PDF file.

Practical recommendations:

* The scanned images become smaller in size if the images are read in just black/white (or grayscale). This may sometimes even facilitate extracting the information (see below).
* If the exams were stapled in the top-left corner (see above) the sheet feeder often works better if the sheets are rotated by 180 degrees (so that the damaged corner is not fed first into the machine). This often improves the scanning results considerably and can be accomodated by setting rotate = TRUE in nops\_scan() below.

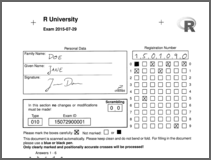
For demonstration, we use two completed demo sheets that are provided along with the exams package and copy them to a dedicated directory nops\_scan.

img <- dir(system.file("nops", package = "exams"), pattern = "nops\_scan",

full.names = TRUE)

dir.create("nops\_scan")

file.copy(img, to = "nops\_scan")

**[](http://www.r-exams.org/assets/posts/2017-09-18-exams2nops/nops_scan1.png)** **[](http://www.r-exams.org/assets/posts/2017-09-18-exams2nops/nops_scan2.png)**

(Note that the first scanned image corresponds to one of the PDFs above while the other one was generated with custom title/logo/language/etc.)

Using the function nops\_scan() we can now read all scanned images (i.e., all locally available PNG and/or PDF files) and collect everything in a ZIP file. (Note that if there were PDF files that need to be scanned, then the PDF toolkit pdftk and the function convert from ImageMagick need to be available outside of R on the command line.)

nops\_scan(dir = "nops\_scan")

dir("nops\_scan")

## [1] "nops\_scan\_20170810004404.zip" "nops\_scan1.png"

## [3] "nops\_scan2.png"

The resulting file **nops\_scan\_20170810004404.zip** contains copies of the PNG files along with a file called Daten.txt (for historical reasons) that contains the scanned information in machine-readable from.

See ?nops\_scan for more details, e.g., multicore support or options for rotating PDF files prior to scanning.

## Evaluate results

In the previous scanning step the exam sheets have just been read but not yet evaluated, i.e., it has not yet been assessed which questions were answered (partially) correctly and which were wrong, and no points have been assigned. Therefore, we use nops\_eval() to carry out these computations and to make the results available - both in a format easy to read for machines (a CSV file) and a format for humans (one HTML page for each student).

To do so, three files are required:

* An RDS file with the exam meta-information, generated by exams2nops() above.
* A ZIP file with the scanned sheets, generated by nops\_scan() above.
* A CSV file (semicolon-separated values) with the student infomation (registration number, name, and some ID or username). In practice, this CSV file will typically be processed from some registration service or learning management system etc. However, here we simply create a suitable CSV file on the fly.

write.table(data.frame(

registration = c("1501090", "9901071"),

name = c("Jane Doe", "Ambi Dexter"),

id = c("jane\_doe", "ambi\_dexter")

), file = "Exam-2015-07-29.csv", sep = ";", quote = FALSE, row.names = FALSE)

The resulting file is **Exam-2015-07-29.csv**.

Now the exam can be evaluated creating an output data frame (also stored as a CSV file) and individual HTML reports (stored in a ZIP file). Here, we employ an evaluation scheme without partial points in the multiple-choice questions and differing points across questions.

ev1 <- nops\_eval(

register = "Exam-2015-07-29.csv",

solutions = "nops\_pdf/demo.rds",

scans = Sys.glob("nops\_scan/nops\_scan\_\*.zip"),

eval = exams\_eval(partial = FALSE, negative = FALSE),

interactive = FALSE

)

dir()

## [1] "Exam-2015-07-29.csv" "nops\_eval.csv" "nops\_eval.zip"

## [4] "nops\_pdf" "nops\_scan"

The evaluated data can be inspected by opening **nops\_eval.csv** in some spreadsheet software or we can directly look at the data in R. Based on this information, the marks could be entered into the university’s information system.

ev1

## registration name id exam scrambling

## 1501090 1501090 Jane Doe jane\_doe 15072900001 00

## 9901071 9901071 Ambi Dexter ambi\_dexter 15072900002 00

## scan points mark answer.1 solution.1 check.1 points.1

## 1501090 nops\_scan1.png 4 5 00100 00100 1 1

## 9901071 nops\_scan2.png 0 5 10100 10000 0 0

## answer.2 solution.2 check.2 points.2 answer.3 solution.3 check.3

## 1501090 11101 11100 0 0 00000 00000 1

## 9901071 10111 11001 0 0 01000 01010 0

## points.3 answer.4 solution.4 check.4 points.4 answer.5 solution.5

## 1501090 1 00100 00110 0 0 00010 00010

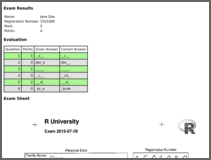
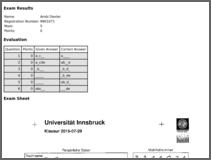
## 9901071 0 00000 01011 0 0 00000 11010

## check.5 points.5 answer.6 solution.6 check.6 points.6

## 1501090 1 2 01101 01111 0 0

## 9901071 0 0 11100 00011 0 0

And nops\_eval.zip contains subdirectories with HTML reports for each of the two participants.

**[](http://www.r-exams.org/assets/posts/2017-09-18-exams2nops/Exam-2015-07-29-jane_doe.html)** **[](http://www.r-exams.org/assets/posts/2017-09-18-exams2nops/Exam-2015-07-29-ambi_dexter.html)**

set.seed(2018-05-16)

exams2nops(choice, institution = "eRum 2018", language = "hu")

Second, the choice-based exam version can be exported into the JSON format for ARSnova: [Rexams-1.json](http://www.r-exams.org/assets/posts/2018-05-20-erum2018/Rexams-1.json). This contains an entire ARSnova session that can be directly imported into the ARSnova system as shown above. It employs a custom exercise set up just for eRum as well as a slightly tweaked exercise that displays better in ARSnova.

set.seed(2018-05-16)

exams2arsnova(list("conferences.Rmd", choice[[1]], "fruit3.Rmd", choice[[3]]),

name = "R/exams", abstention = FALSE, fix\_choice = TRUE)

Third, the e-learning exam can be generated in QTI 1.2 format for OpenOLAT, as shown above. The exams2openolat command below is provided starting from the current R/exams version 2.3-1. It essentially just calls exams2qti12 but slightly tweaks the MathJax output from pandoc so that it is displayed properly by OpenOLAT.

set.seed(2018-05-16)

exams2openolat(elearn, name = "eRum-2018", n = 10, qti = "1.2")

**What else?**

In the last part of the presentation a couple of new and ongoing efforts within the R/exams project are highlighted. First, the natural language support in NOPS exams is mentioned which was recently described in more detail in this blog. Second, the relatively new “stress tester” was illustrated with the following example. (A more detailed blog post will follow soon.)

Natural Language support in NOPS exams

## Example

To illustrate how the language support works, once it has been fully incorporated into the exams package, we set up a short exam with three exercises: **deriv2**, **tstat2**, **swisscapital**. All of these are readily available in the package (and are actually in English).

library("exams")

myexam <- c("deriv2.Rnw", "tstat2.Rnw", "swisscapital.Rnw")

Then we set up PDF output in English (en), German (de), and Spanish (es). By setting language most text on the title page is modified, only the name of the institution and the title of the exam have to be set separately. For the English example we produce n = 1 PDF output file in the output directory nops\_pdf (created automatically).

set.seed(403)

exams2nops(myexam, n = 1, language = "en",

institution = "R University", title = "Exam",

dir = "nops\_pdf", name = "en", date = "2018-01-08")

Then we do the same for the other two languages.

set.seed(403)

exams2nops(myexam, n = 1, language = "de",

institution = "R Universit\\\"at", title = "Klausur",

dir = "nops\_pdf", name = "de", date = "2018-01-08")

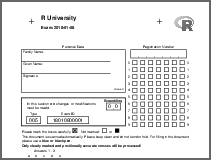
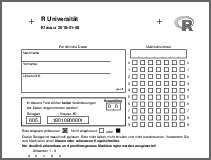
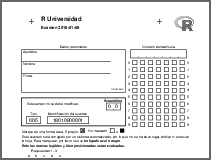
set.seed(403)

exams2nops(myexam, n = 1, language = "es",

institution = "R Universidad", title = "Examen",

dir = "nops\_pdf", name = "es", date = "2018-01-08")

The title pages of the resulting PDF files then have the desired languages.

**[](http://www.r-exams.org/assets/posts/2018-01-08-nops_language/en1.pdf)** **[](http://www.r-exams.org/assets/posts/2018-01-08-nops_language/de1.pdf)** **[](http://www.r-exams.org/assets/posts/2018-01-08-nops_language/es1.pdf)**

## Language specification

To add a new language, essentially just a single text file (say lang.dcf) is needed containing suitable translations of all the phrases on the title page as well as a few additional phrases, e.g., occuring in the HTML evaluation reports etc. As an example, the first few phrases in English (en.dcf) are:

PersonalData: Personal Data

Name: Name

FamilyName: Family Name

GivenName: Given Name

Signature: Signature

RegistrationNumber: Registration Number

Checked: checked

NoChanges: In this section \textbf{no} changes or modifications must be made!

...

And the corresponding translations to German (de.dcf) are:

PersonalData: Pers{\"o}nliche Daten

Name: Name

FamilyName: Nachname

GivenName: Vorname

Signature: Unterschrift

RegistrationNumber: Matrikelnummer

Checked: gepr{\"u}ft

NoChanges: In diesem Feld d{\"u}rfen \textbf{keine} Ver{\"a}nderungen der Daten vorgenommen werden!

...

Note that here LaTeX markup is used for the German umlaute and for bold highlighting. Alternatively, special characters can be added in a suitable encoding (typically UTF-8) but then the encoding has to be declared when calling exams2nops() (e.g., encoding = "UTF-8").

Most of the phrases required in the .dcf are very straightforward and only some are a bit technical. There are also a couple of coordinates (MarkExample\*) necessary for aligning some text lines. If you have set up your own lang.dcf you can easily pass it to exams2nops() by setting language = "/path/to/lang.dcf". The same has to be done for nops\_eval() when evaluating the exam.

## Currently available languages

Due to the kind support from friends and various dedicated R/exams users, there is already support for many important Western languages as well as a few other languages/countries. All of these are directly available in the R package. But for convenience and manual inspection the .dcf files are also linked here.

| **File** | **Language** | **Contributor** |
| --- | --- | --- |
| **cz.dcf** | Czech | **Jindřich Marek** |
| **da.dcf** | Danish | **Tue Vissing Jensen** & **Jakob Messner** |
| **de.dcf** | German | **Achim Zeileis** |
| **en.dcf** | English | **Achim Zeileis** |
| **es.dcf** | Spanish | **Maria Kogelnik** |
| **fi.dcf** | Finnish | **Klaus Nordhausen** |
| **fr.dcf** | French | **Arthur Allignol** |
| **gl.dcf** | Galician | **Marta Sestelo** & **Nora M. Villanueva** |
| **gsw.dcf** | Swiss German | **Reto Stauffer** |
| **hr.dcf** | Croatian | **Krunoslav Juraić** & **Tatjana Kecojevic** |
| **hu.dcf** | Hungarian | **Gergely Daróczi** & Dénes Tóth |
| **it.dcf** | Italian | **Domenico Zambella** |
| **ja.dcf** | Japanese | **Kohei Watanabe** |
| **ko.dcf** | Korean | 정세원 |
| **nl.dcf** | Dutch | **Niels Smits** |
| **no.dcf** | Norwegian (Bokmål) | **Tormod Bøe** |
| **pt.dcf** (**pt-BR.dcf**, **pt-PT.dcf**) | Portugese | **Mauricio Calvão** & Fabian Petutschnig & **Thomas Dellinger** |
| **ro.dcf** | Romanian | **Cristian Gatu** |
| **ru.dcf** | Russian | **Boris Demeshev** |
| **sk.dcf** | Slovak | Peter Fabsic |
| **sl.dcf** | Slovenian | Matjaž Jeran |
| **sr.dcf** | Serbian | **Tatjana Kecojevic** |
| **tr.dcf** | Turkish | **Emrah Er** |
| **vi.dcf** | Vietnamese | Trần Thị Hoàng Hà |

## Contributing new languages

If you want to contribute a new language, simply set up a .dcf file starting out from one of the examples above and send the file or a link to <info at R-exams.org>. Do not worry if not everything is 100% perfect, yet, we can still sort this out together! For Western languages (e.g., sv is still missing) it is probably the most robust solution to code special characters in LaTeX. For languages requiring other alphabets (e.g., gr) it is probably easiest to use UTF-8 encoding.

s <- stresstest\_exercise("deriv2.Rnw")

plot(s)

Finally, a psychometric analysis illustrated how to examine exams regarding: Exercise difficulty, student performance, unidimensionality, fairness. The replication code for the results from the slides is included below (omitting some graphical details for simplicity, e.g., labeling or color).

## load data and exclude extreme scorers

library("psychotools")

data("MathExam14W", package = "psychotools")

mex <- subset(MathExam14W, nsolved > 0 & nsolved < 13)

## raw data

plot(mex$solved)

## Rasch model parameters

mr <- raschmodel(mex$solved)

plot(mr, type = "profile")

## points per student

MathExam14W <- transform(MathExam14W,

points = 2 \* nsolved - 0.5 \* rowSums(credits == 1)

)

hist(MathExam14W$points, breaks = -4:13 \* 2 + 0.5, col = "lightgray")

abline(v = 12.5, lwd = 2, col = 2)

## person-item map

plot(mr, type = "piplot")

## principal component analysis

pr <- prcomp(mex$solved, scale = TRUE)

plot(pr)

biplot(pr, col = c("transparent", "black"),

xlim = c(-0.065, 0.005), ylim = c(-0.04, 0.065))

## differential item functioning

mr1 <- raschmodel(subset(mex, group == 1)$solved)

mr2 <- raschmodel(subset(mex, group == 2)$solved)

ma <- anchortest(mr1, mr2, adjust = "single-step")

## anchored item difficulties

plot(mr1, parg = list(ref = ma$anchor\_items), ref = FALSE, ylim = c(-2, 3), pch = 19)

plot(mr2, parg = list(ref = ma$anchor\_items), ref = FALSE, add = TRUE, pch = 19, border = 4)

legend("topleft", paste("Group", 1:2), pch = 19, col = c(1, 4), bty = "n")

## simultaneous Wald test for pairwise differences

plot(ma$final\_tests)