R packages

Packages in R are bundles of code, documentation, data and tests (Wickham, 2015). They make up a fundamental part of the R language since they allow to share code with other R users. In other programming languages these units are often referred to as libraries. The constantly growing amount of R packages is one characteristic of R as an open source software (Chambers, 2008). Packages can be downloaded from CRAN (<https://cran.r-project.org/)> which provides the official repository for R packages as well as from GitHub via the devtools package (Wickham & Winston, 2016). Currently (January 26th, 2017), there are 9988 packages available on the CRAN website.

There are a few noticeable aspects why packages form such a fundamental basis of R. First, they allow to share code with others which enables a consistently growing range of functionalities in R. Moreover, when loading a package, all functions of that package are automatically loaded into the namespace. This is very time-saving especially when a package is frequently used as opposed to sourcing scripts with the required functions. Dependencies, other packages that the loaded package requires, are automatically loaded or even installed if necessary. And finally, the documentation of functions in a package allows the creator of the package as well as other users to understand the way a function works.

A personal motivation to create a package for repeated measures ANOVA was there were functionalities, like the adjusted confidence intervals, that we needed for a research project which had not been implemented in R, or at least not conveniently bundled in a single package. The MAGA package is an attempt to fill this void and provide the functionalities for other R users and researchers that use repeated measures ANOVA.

The Quantlet structure of the code made it easy to choose the functionalities that were turned into a function in the package. For creating the package, devtools (Wickham & Winston, 2016) and roxygen2 (Wickham, Danenberg & Eugster, 2015) were used. While devtools automates different tasks of package development, roxygen2 can be used to write package documentation directly into the R scripts of the functions which is then compiled to Rd format (Wickham, 2015). Since the MAGA package has not been published on CRAN yet (January, 26th 2017), devtools::install.github() may be used to install the package from its GitHub repository.

When writing the package, we carefully chose function names that indicate what the function does for a convenient use of the package. Another topic which is especially relevant when writing a package is error handling. This is crucial to ensure that the functions are robust against violation of the requirements of the function arguments. In each of the functions of the package, a sequence of if-statements was implemented to check if the requirements of the function input were met. When considering error handling in package creation it is important to differentiate between the functions stop() and warning(). Both return a message to the user but only stop() interrupts the code. So whenever we checked for requirements that were essential for the function to work we chose stop(). Additionally, the function documentation specifies the data type of each argument (see section about function documentation).

The following code displays the if-statements that were used to check whether the input data meets the requirements:

# id must either be an integer specifying the column position of the independent variable

if (id %in% 1:ncol(rma\_data) == FALSE || length(id) != 1) {

stop("id must be an integer specifying the column position of the independent variable")

}

This part of the code ensures that the id variable is specified via a single integer which specifies the column position of the ID variable. This variable is an index variable which numerates the subjects. If the variable is not an integer between 1 and the number of columns of the input data an error message is returned.

# all variables must be numeric

if (all(sapply(rma\_data, is.numeric)) == FALSE | any(sapply(rma\_data, is.factor))) {

stop("All variables in rma\_data must be numeric")

}

This if-statement controls for the right data type. All variables of the data must be numeric. Since R considers factor variables numeric, the code after the logical OR ensures that the function is also interrupted when one or more variables are factors. Again, an error message is returned in that case to inform the user about the requirements.

# n > k (i.e. more entities than factor levels)

if (n <= k) {

stop("Number of entities must exceed number of factor levels")

}

Moreover, for ANOVA methods to work, the number of subjects needs to exceed the number of factor levels. This means that the number of rows (automatically defined as n by the function) has to be strictly larger than k (the number of columns minus one for the index variable) .

# k >= 2 (i.e. at least two or more factor levels)

if (k < 2) {

stop("At least two factor factor levels required")

}

Finally, the data needs to have at least two factor levels, since otherwise no ANOVA model can be estimated. This is controlled by the last of the if-statements.

A particular case where the warning() function was used is described in section XXX.

Function documentation can be demonstrated on one particular function since the necessary steps apply to all other functions as well. For demonstration purposes, the function ow\_rma\_opc was chosen. The following code is compiled by roxygen2 to an Rd file which contains the information that are displayed when the help page in R is called, for example via the command ?ow\_rma\_opc.

Each line in the documentation needs to start with #'.

The first line is the heading of the help page and summarizes briefly what the function does.

#' Estimate and plot orthogonal polynomial trends

#'

The next line is displayed by roxygen2 via the heading *Description* and features short yet precise information about the function.

#' Compute orthogonal polynomial contrasts and plot orthogonal polynomial regression curves for a repeated measures ANOVA.

#'

The next section *Usage* is automatically created by roxygen2. It demonstrates how to call the function.

Via @param the function parameters are listed under the heading *Arguments*. Here, it is important to inform the user about the required data types, e.g. numeric vector or data.frame. Moreover, default values, if existing, are documented.

#' @param ow\_rma\_data An object of type data.frame. Each row should represent one subject and each column one variable.

#' @param id An integer specifying the column position of the subject ID. Default is 1. Set to "none" if the data does not contain an ID variable.

#' @param maxpoly Integer. Specifies the highest order polynomial for the contrast analysis. Truncated to number of factor levels (k) -1 if larger. Default is NA which will be set to k-1 within the function.

#'

In the next step, the object(s) that the function returns need to be specified. This is achieved via @return and summarized under the section label *Value*. Here, the function returns a list. In the next lines \item is used to describe the objects in particular. The list stores a data frame containing the results from the polynomial contrast analysis and the ggplot object that displays the orthogonal polynomial regression curves.

#' @return Returns an object of type list.

#' \item{contrast\_table}{An object of type data.frame containing the contribution of the polynomial trends to the total factor effect and their respective significance levels}

#' \item{poly\_plot}{A ggplot object. Displays the orthogonal polynomial regression curves.}

The last sections contain the authors’ names, notes, literature and examples if provided.

#' @author Joachim Munch, Frederik Schreck, Quang Nguyen Duc, Constantin Meyer-Grant, Nikolas Hoeft

#' @note LITERATURE

#' @examples

#'

Finally @rdname specifies where the documentation is stored, here for the function ow\_rma\_opc, and @export is especially important to ensure that the function is added to the namespace file.

#' @rdname ow\_rma\_opc

#' @export

A final word needs to be said about the description file. A template for this is automatically created when building a new package in RStudio. Basically, this file contains some meta data about the package like a description, which does not require further elaboration. One important aspect though is the use of dependencies. These are other packages that are required to run the functions in the new package. In the section *Imports* the names of the required packages are specified. When the MAGA package is installed, R automatically ensures that the dependencies (here: dplyr, ggplot2 and tidyverse) are installed as well. Moreover, they are also loaded whenever MAGA is loaded via the library() command.

Package: MAGA

Type: Package

Title: A package to make ANOVA great again!

Version: 0.1.0

Author: Joachim Munch, Frederik Schreck, Quang Nguyen Duc, Constantin Meyer-Grant, Nikolas Hoeft

Maintainer: Nikolas Hoeft <nikolas.hoeft@hu-berlin.de>

Description: A package for repeated measures ANOVA models.

Allows to simulate data, compute ANOVA and RM ANOVA

models, investigate error term reduction due to model selection,

compute and display adjusted confidence intervals.

Furthermore it enables to conduct Mauchly's test for

sphericity and it provides various correction factors to

adjust p-values in the presence of sphericity. The package

also supplies a function to fit orthogonal polynomial

contrasts and to plot orthogonal polynomial regression curves.

Imports:

dplyr,

ggplot2,

tidyverse

License:

Encoding: UTF-8

LazyData: true

RoxygenNote: 5.0.1

Of course there are many other possible steps for package development. The ones described above however are the most important and basic ones and were therefore chosen for this report.

**LITERATURE**

Chambers, J. (2008). *Software for Data Analysis*. New York, NY: Springer New York. Retrieved from http://link.springer.com/10.1007/978-0-387-75936-4

Wickham, H. (2015). *R Packages* (1st ed.). Sebastopol, CA: O’Reilly and Associates.

Wickham, H. & Chang, W. (2016). devtools: Tools to Make Developing R Packages Easier. R package version 1.12.0. <https://CRAN.R-project.org/package=devtools>

Wickham, H., Danenberg, P. & Eugster, M. (2015). roxygen2: In-Source Documentation for R. R package version 5.0.1. https://CRAN.R-project.org/package=roxygen2