

732A94 Advanced R Programming

Computer lab 5

Krzysztof Bartoszek
(designed by Leif Jonsson and Måns Magnusson)

18 September 2017

Seminar date: **3 October 08:15** (P18)
Lab deadline: **6 October 23:59**

Instructions

- This lab should be conducted by students **two by two**.
 - The lab consists of writing a package that is version controlled on github.com.
 - Both student should **contribute equally much** to the package.
 - Other significant collaborations/discussions should be acknowledged in the solution.
 - Commit continuously your addition and changes.
 - Collaborations should be done using github (ie you should commit using your own github account).
 - In the lab some functions can be marked with an *. These parts are only mandatory for students taking the advanced course or students working together in groups of three.
 - The deadline for the lab is on the lab's title page.
 - The lab should be turned in as a url to the repository containing the package on github using **LISAM**. This should also include name, github user names and liuid of the students behind the project. In case of problems the url may be emailed to **krzysztof.bartoszek@liu.se**.
-

Contents

1	A package connecting to a web API	3
1.1	Create a package	3
1.1.1	Write the R code	3
1.1.2	Write unit tests	3
1.1.3	Documentation and vignettes	3
1.2	* Create a Shiny application using your package	4
1.3	Seminar and examination	4
1.3.1	Examination	4

Chapter 1

A package connecting to a web API

In this lab we will create a package to connect to a web API using R. The purpose of this lab is to start the work with a API package you will use and develop further during the course. You can choose a connection to any API you find to have interesting data to analyze as well (you may use this data in a machine learning project in the last lab). If the project is good enough you will be able to submit this package to CRAN during the last week.

The API needs to be accepted by the teacher before you start. In the document `732A94_AdvancedRHT2017_Lab05_API_projects.pdf` you can find API:s suggestions to implement as an R package.

Master students will also implement a simple Shiny application using the package to connect to the API.

1.1 Create a package

Start out by creating a new package on github see lab 3 for details on how to setup a package. Think about the naming of your package.

Read about the package and discuss which functions you should have to access the data in the API. There are often limits and configurations for a given API you need to access.

In genreal it is good to have as few functions as possible theat are exported in the package. But behind these functions you may need to have more functionality not accessed by the users of your package.

1.1.1 Write the R code

Read about the package and discuss which functions you should have to access the data in the API. There are often limits and configurations for a given API you need to access.

In general it is good to have as few functions as possible that are exported in the package. But behind these functions you may need to have more functionality not accessed by the users of your package.

1.1.2 Write unit tests

Write a unit test suite for the functions in the package. Examples of test can be:

1. Downloading a big query
2. Testing the limits of the API configuration
3. Testing the inputs and outputs of your functions
4. Testing that the function always return a given value for a specific set of inputs.

See chapter “Testing” in [3] or [2] for details.

1.1.3 Documentation and vignettes

Write a vignette that describes your package and how to use it with examples. For more information on how to write a vignette and include them in your package see “Vignettes” in [3].

Document all public functions using `roxygen2`. See chapter “Object documentation” in [3] for details.

1.2 * Create a Shiny application using your package

Create a new repository on github for a Shiny application. Create a simple Shiny application that makes it possible to interactively analyze data using your API package. At least one interactive widget should be used. See “Lesson 6” in [1] for information on how to handle the situation of reactive programming in Shiny when “slow” data is used.

It should be possible to run your shiny app out using the `runGitHub()` function.

1.3 Seminar and examination

During the seminar you will bring your own computer and demonstrate your package and what you found difficult in the project.

We will present as many packages as possible during the seminar and you should

1. Show that the package can be built using R Studio and that all unit tests is passing.
2. Discuss why you implemented the functions as you did.
3. Give a short presentation of the testsuite.
4. Show your vignette/run the examples live.
5. * Present your Shiny application.

1.3.1 Examination

Turn in a the adress to your github repo with the package (and Shiny application) using LISAM. To pass the lab you need to:

1. Have the R package up on GitHub with a Travis CI pass/fail badge.
2. The test suites for the implemented function(s) should be included in the package. You should have written these tests yourselves and should include at least 10 different unit test in one test suite.
3. The package should build without warnings (pass) on Travis CI.

Bibliography

- [1] R Studio. Shiny tutorial, 2016.
- [2] Hadley Wickham. testthat: Get started with testing. *The R Journal*, 3(1):5–10, 2011.
- [3] Hadley Wickham. *R packages*. ” O’Reilly Media, Inc.”, 2015.