Stat 243: Lab, Monday Oct-24

Gaston Sanchez

Creating a simple R package for rolling a die

The purpose of this lab is to practice creating a simple R package using the code that you wrote last week with the _die-rolling-programming‘ example.

If you didn't follow the material presented in lecture, you may need to install the following packages:

```
install.packages(c("devtools", "roxygen2", "testthat", "knitr"))
```

Creating a package as an R Project

Here's the list of suggested steps to create an R package:

- Open RStudio
- Go to `File` in the menu bar
- Select `New Project`
- Choose `New Directory`
- Choose `R Package`
- Specify a name for the package (e.g. `rolldie`), and choose directory
- Click on button `Create Project`

What's in the package structure?

The series of previous steps should initialize an "R project" with the following contents:

```
.Rbuildignore
rolldie.Rproj
DESCRIPTION
NAMESPACE
R/
man/
.Rproj.user/
```

- Open the `DESCRIPTION` file and customize its fields with your information
- Don't touch `NAMESPACE` (this will be updated via `devtools`)
- Don't modify the contents in `man/`

Adding code and Roxygen comments for `die`

- Go to the `R/` directory and add an R script file that will contain your the code of the `"die"` object: `die()`, checking functions, `print.die()`, etc
- Don't include the code of `"roll"` yet

- Document your code using Roxygen comments
- Add examples to the main functions
- If you need some help, check the file `die-roll-s3class.R` to see examples of roxygen comments.
- Walk through the packaging stages with `"devtools"`:

```
library(devtools)

# creating documentation (i.e. the Rd files in man/)
devtools::document()

# checking documentation
devtools::check_man()

# building tarball (e.g. rolldie_0.1.tar.gz)
devtools::build()

# checking install
devtools::install()
```

If you managed to create the package with no problems, then continue adding another R script file in the folder `R/` that will contain the functions and methods of the `roll` object.

Including tests for your functions

There are various packages in R that allow you to include unit tests. One of them is the package `"testthat"`.

- In the directory of the package, create a folder `"tests"`
- Inside the folder `tests/` create another folder `"testthat"`; this is where you include R scripts containing the unit tests.
- All the script files inside `testthat/` should start with tha name **test** e.g. `test-die.R`, `test-roll.R`, etc.
- Inside the folder `testthat/`, create an R script `test-check-die.R`

The idea is to write tests for `check_sides()`, `check_prob()`, and `die()`.

- Go to the `tests/testthat/` folder and open the `test-check-die.R` file
- use `context()` to describe what the test are about
- to run the tests from the R console, use the function `test_file()`
- Give a `context()` to to describe what the test are about.
- You typically group various related tests in one context; e.g. a context for *die arguments*
- In a given context, you form groups of expectations inside calls to `test_that()`
- For example, you can expect that `check_sides()` returns `TRUE` if the value of `sides` is of length 6.
- Likewise, you can expect an error from `check_sides()` if the values of `sides` is of length different than 6:

```
context("Die arguments")

test_that("check_sides with ok vectors", {

  expect_true(check_sides(1:6))
  expect_true(check_sides(c(2, 4, 6, 8, 10, 12)))
})


test_that("check_sides fails with invalid lengths", {

  expect_error(check_sides(1:5))
  expect_error(check_sides(1:7))
})
```

To run the tests, you use the "devtools" function `test()`. This means that your typical packaging workflow will look like this:

```
library(devtools)

# creating documentation (i.e. the Rd files in man/)
devtools::document()

# checking documentation
devtools::check_man()

# running tests
devtools::test()

# building tarball (e.g. oski_0.1.tar.gz)
devtools::build()

# checking install
devtools::install()
```

Keep adding more tests and rebuild the package

Vignettes

To create your first vignette with "devtools", run:

```
devtools::use_vignette("introduction-roll-die")
```

This will:

1. Create a `vignettes/` directory.
2. Add the necessary dependencies to `DESCRIPTION` (i.e. it adds `knitr` to the `Suggests` and `VignetteBuilder` fields).
3. Make a draft vignette, `vignettes/introduction-roll-die.Rmd`.

Once you have this file, you need to modify the vignette. After editing the vignette, your expanded workflow will be like this:

```
library(devtools)

# creating documentation (i.e. the Rd files in man/)
devtools::document()

# checking documentation
devtools::check_man()

# run tests
devtools::test()

# checking documentation
devtools::build_vignettes()

# building tarball (e.g. oski_0.1.tar.gz)
devtools::build()

# checking install
devtools::install()
```