Advanced Topics

Clemens Schmid

Git

https://r-pkgs.org/git.html

Git is a version control system that documents all changes to files in a directory

- Documentation of all steps in the development process
- Browsing and rolling back to old stages
- Safe collaboration on the same files

```
?usethis::use_git()
```

Github

https://github.com

Github is an online platform to store, publish and manage Git projects

- Free and easy “backup” and publication
- Low lock-in, because based on git
- Manage collaborative working: Issues, Pull Requests
- Well implemented interaction with many coding web services
- Social media features: A community of coding archaeologists

```
?usethis::use_github()
```

Vignettes

https://r-pkgs.org/vignettes.html

Vignettes are a special part of the R package documentation, that explains the workflows a package supports

```
browseVignettes()
vignette(topic = "introduction", package = "cowplot")
```

- Free form: Tutorial/Blog post/Book chapter/Paper
- Less technical, more focussed on applications and usecases
- Usually written in Rmarkdown

```
?usethis::use_vignette()
```

Unit testing

https://r-pkgs.org/tests.html

A unit test is test code to check if a function returns what you expect given a certain input

- Guarantee correctness of complex functions by testing them and their smaller parts
- Facilitates refactoring and prevents you from accidentally breaking things
- Test code serves as a concrete example for how your functions can be used

```
?usethis::use_testthat()
```

CI

https://github.com/r-lib/actions

Continuous integration means automatic testing and checking of code changes. Multiple companies offer free web services for open source projects (Travis-CI, Gitlab, Github)

- Each change to your project triggers a full test
- Clean, virtual environments and multiple OS
- Not just for testing, but also for any other operations (e.g. deploying a static website)
- Should be used sparingly to save energy

```
?usethis::use_github_action
```

OOP and custom types

https://adv-r.hadley.nz/oo.html

R supports multiple different Object-oriented programming systems to implement packages with specific needs for their data types

- Competing systems: S3, S4, RC, R6, R.oo, proto, ?
- Many packages rely on custom S3, S4 or R6 classes
- Many functions in these packages are class methods, so they only work with specific input data types or work explicitly different for different input types
- base R relies extensively on S3 classes

OOP and custom types

```
?as.data.frame
```

```
## S3 method for class 'list'
as.data.frame(
  x, row.names = NULL, optional = FALSE, ...,
  cut.names = FALSE, col.names = names(x),
  fix.empty.names = TRUE,
  stringsAsFactors = default.stringsAsFactors()
)

## S3 method for class 'matrix'
as.data.frame(
  x, row.names = NULL, optional = FALSE,
  make.names = TRUE, ...,
  stringsAsFactors = default.stringsAsFactors()
)
```

Compiled code

https://r-pkgs.org/src.html

R packages can incorporate code from compiled languages to speed up processes

- C, C++, Fortran, . . .
- Orders-of-magnitude performance increases
- Steep learning curve
- C++ with Rcpp (https://adv-r.hadley.nz/rcpp.html)

```
?usethis::use_rcpp()
```

Releasing a package to CRAN

https://r-pkgs.org/release.html

At the very end of the initial package development process you can consider a submission to CRAN

- Intensive checks on different test systems
- The CRAN submission process
- There are serious alternatives to a CRAN submission (https://ropensci.org/r-universe)

```
?devtools::release()
```