Best Practices

Martin Morgan (mtmorgan@fhcrc.org)
Fred Hutchinson Cancer Research Center
Seattle, WA

7 February 2014

Best practices

1. Write organized, consistent *R* code
2. Use version control
3. Document functions
4. Write tests
5. Write vignettes
6. Create a package (!)
7. What about *Rstudio*? Makes this easy!

Example: utilities for working with *GRanges*, e.g., `isSimpleVariant`. See

```
system.file(package="SummerX", "GRangesUtilities.tar.gz")
```

Writing organized, consistent *R* code

- Organize frequently used commands into not-too-complicated functions
- Adopt consistent coding conventions
  - Function and variable names
  - Indentation
  - Line lengths
  - ...
- Organize functions into files, e.g., one-function-per-file

```
|-- GRangesUtilities
    |-- R
        |-- isSimpleVariant.R
        |-- stickFigure.R
    |-- vignettes
        |-- UsingGRangesUtilities.Rmd
```

Using version control

- Easily keep track of changes as your documents develop, without using confusing file-naming or other conventions.

Software

- *git* – modern, flexible, easy to use locally (no server required)
- *subversion* (*svn*) – used by *Bioconductor*, requires central server

Sharing with others

- Use github, or. . .
- Get your IT guys to set up a *git* or *svn* server for your group's use

Using version control – *git*

- Change into the directory where you've started your project

```
$ cd GRangesUtilities
```

- Initialize a *git* repository and check the status

```
$ git init
$ git status
```

- Create / edit files, directories. . . ; track in git

```
$ git add R/isSimpleVariant.R
$ git status
```

- Commit the changes

```
$ git commit
```

- See the commit log

```
$ git log
```

Document functions

- Purpose: document how to use *function*
- `man` directory with '`Rd`' files, organized like `R` files, e.g., `isSimpleVariant.Rd`
- Alternative: use *roxygen2* package to add 'annotations', e.g., to *R* code.

Write tests

Unit tests

- Short tests of specific parts of each function, implemented in a `tests` directory
- *testthat* framework
- *RUnit* framework, used in *Bioconductor*

Test-driven development

- Write unit tests that describe expected functionality *before* implementing the code.

Write vignettes

Why?

- Purpose: document how to use several functions in an integrated way
- 'Literate' programming: Text, figures, tables surrounding *R* script

How?

- Write documents in a directory `vignettes`
- `Rmd`: 'markdown' and *R* – easy
- `Rnw`: 'Sweave' combines LaTeX and *R* to produce PDF documents – flexible

Create a package

- Why? Easy to re-use, share with others (e.g., lab members)
- How? – `RShowDoc("R-exts")`

From what we've already done...

- Add a `DESCRIPTION` file
- Arrange for tests to be run when the package is checked

Additional (optional) steps

- `data` directory of *R* data objects
- `inst/script` of *R* scripts

Making a package available to your colleagues

roxygen2-ize to create NAMESPACE, `man` pages

```
$ R -e "roxygen2::roxygenize('GRangesUtilities')"
```

Build, check, and install the package

```
$ R CMD build GRangesUtilities
$ R CMD check GRangesUtilities_0.0.1.tar.gz
$ R CMD INSTALL GRangesUtilities_0.0.1.tar.gz
```

Final step within *R*:

```
install.package("GRangesUtilities_0.0.1.tar.gz",
    repos=NULL)
```

Windows: create a `.zip` file for easy installation

```
$ R CMD INSTALL --build GRangesUtilities_0.0.1.tar.gz
```

Use it!

```
library(GRangesUtilities)
?isSimpleVariant
vignette("UsingGRangesUtilities")
example(isSimpleVariant)
```