REVOLUTION ANALYTICS

Introduction to R: Working with packages

Revolution Analytics

Outline

Overview

In this session we cover loading and creating custom function libraries. The objectives are:

- Learn how to locate, install, load and update packages from the R community.
- Demonstrate how to create and import a library of custom functions.
- Show how to hack the source code of imported packages.

What is a package?

- One of R's key assets is its library of packages
- CRAN team makes sure that these packages are clean and safe (to some extent)
- In general, whatever the problem, "There is a package for that"
- Packages typically come with excellent documentation

Outline

Where to find packages?

- The primary package repository is CRAN
  - http://cran.r-project.org/
- Also see CRANtastic
  - http://crantastic.org/
- The CRAN website has a “Task Views” page that allows you to view packages according to subject area
  - http://cran.r-project.org/web/views/

Package example: random forests

For example, let's say we'd like to analyze the `mtcars` dataset using a random forest model.

The base R installation does not have this capability natively.

Fortunately, Andy Liaw and Matthew Wiener have written a package that encodes Brieman's (2001) algorithm:

```
install.packages("randomForest")  #install the package
```

Estimate a Random Forest

```
library(randomForest)  # load the package into the workspace
# ?randomForest # check package documentation
rf.mpg <- randomForest(mpg ~ ., data = mtcars)  # fit model
rf.mpg  # display model prediction performance
```

```
##
## Call:
##  randomForest(formula = mpg ~ ., data = mtcars)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
...
```

Loading packages

From the command line:

```
install.packages("package name")
```

The R GUI, RStudio and Revolution IDE also provide convenient menu options.

NOTE: Both these approaches rely on access to a CRAN mirror. If access to the mirror is fire-walled, you may need to download the package zip files and install them from the hard drive.

Exercise: Download a package

Your turn:

- What kinds of problems do your normally solve in your work/research?
- Locate a package designed for this kind of problem.
- Install it, load it, and view its documentation.
- Try running some of the simple examples included in the documentation.

Getting more info about packages

Never underestimate the power of Google

- Hint: Try the search term `CRAN random forest` to search for random forests

Vignettes

Many packages have vignettes.

- A vignette is a self-documenting example of functions in the package
- Use `vignette()` to see what's available.
- Use `vignette('vignette name')` to view a specific vignette.

```
vignette()
library(help = "randomForest")
```

See the pdf package manuals on the CRAN site.

Outline

Find the source code

Usually, you can display the source code of a function by simply entering its name in the command line without any arguments

```
lm
```

```
## function (formula, data, subset, weights, na.action, method = "qr",
##     model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
##     contrasts = NULL, offset, ...)
## {
##     ret.x <- x
##     ret.y <- y
##     cl <- match.call()
...
```

Find the (hidden) source code

This doesn't always give what you expect:

```
randomForest
```

```
## function (x, ...)
## UseMethod("randomForest")
## <environment: namespace:randomForest>
```

This indicates that the function uses object-orientated code. You need to find the default `method` for the `class`

Find the source

code for a non-exported function

Use the idiom `package.name:::function.name` to find the code:

```
randomForest:::randomForest.default
```

```
## function (x, y = NULL, xtest = NULL, ytest = NULL, ntree = 500,
##     mtry = if (!is.null(y) && !is.factor(y)) max(floor(ncol(x)/3),
##         1) else floor(sqrt(ncol(x))), replace = TRUE, classwt = NULL,
##     cutoff, strata, sampsize = if (replace) nrow(x) else ceiling(0.632 *
##         nrow(x)), nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
##     maxnodes = NULL, importance = FALSE, localImp = FALSE, nPerm = 1,
##     proximity, oob.prox = proximity, norm.votes = TRUE, do.trace = FALSE,
...
```

Unknown Package?

If you do not know where a function “lives”, you can use `getAnywhere(functionName)$where` to find out.

For example, you can identify “invisible”” functions using `methods()`:

```
methods(plot)
```

```
##  [1] plot.acf*           plot.data.frame*    plot.decomposed.ts*
##  [4] plot.default        plot.dendrogram*    plot.density*
##  [7] plot.ecdf           plot.factor*        plot.formula*
## [10] plot.function       plot.hclust*        plot.histogram*
## [13] plot.HoltWinters*   plot.isoreg*        plot.lm*
## [16] plot.margin*        plot.medpolish*     plot.mlm*
## [19] plot.ppr*           plot.prcomp*        plot.princomp*
...
```

Use `getAnywhere()`:

```
getAnywhere(plot.data.frame)$where
```

```
## [1] "registered S3 method for plot from namespace graphics"
## [2] "namespace:graphics"
```

Updating packages

Package authors continue to improve and debug them.

To update a package:

- Use `install.packages()`
- Use your IDE to automatically check all packages that are available for updating.

This is helpful when updates result in significant improvements.

Outline

Saving and re-using your own functions.

Your turn:

- Write the `mtcars` summary function from the last module on blank text document.
- Save the document as `customFunctions.R` in the course directory.
- Load that file into your workspace using the function `source()`. (Hint: If the file is in your working directory, you only need to provide the file name. Otherwise specify the file path.) Hint: You can save any function that you write in course labs onto this source file.

Outline

Review questions

- What is a ‘package?’
- Where do we find them?
- How do we install them?
- How can you search for packages or get more information about them?
- How do you view a package’s source code?
- How do you write, save, and use a library of your functions?

REVOLUTION ANALYTICS

Thank you

Revolution Analytics is the leading commercial provider of software and support for the popular open source R statistics language.

www.revolutionanalytics.com
1.855.GET.REVO
Twitter: @RevolutionR