Reproducibility

Recreating results using R-packages

Linh Tran, MPH

Group in Biostatistics
Univ of California, Berkeley

March 10, 2015

INTRODUCTION

Background

Fields requiring reproducibility:

- Biology
- Chemistry
- Physics

We're no exception!

In fact, our field benefits from exact reproducibility

- Given same data and code, we should always get the same answers.
- We should embrace this!

Background

Fields requiring reproducibility:

- Biology
- Chemistry
- Physics

We're no exception!

In fact, our field benefits from exact reproducibility

- Given same data and code, we should always get the same answers.
- We should embrace this!

Background

Ways to reproduce:

1. Series of R scripts
2. The `ProjectTemplate` R package
3. Creating your own R-package
   - What we're focusing on today
   - Using S3 object system (cf. S4 system)

R-PACKAGES

R-packages

R-package: a collection of R code and documentation.

Files/directories needed:

- DESCRIPTION: File describing package
- "R" subfolder: Stores R scripts

Files/directories possibly desired:

- NAMESPACE: File specifying public/private objects
- "data" subfolder: Stores datasets used by package
- "man" subfolder: Stores documentation files explaining code/data in package
- "inst" subfolder: Files/folders to be copied at build time
- "src" subfolder: C/Fortran source code (if used)
- "test" subfolder: Tests for R code
- "vignettes" subfolder: Further author customized documentation

Built packages can be stored on a CRAN mirror as source or binary (Windows/Mac)

DESCRIPTION

Specified in so called Debian-control-file format (i.e. Keyword: Value). e.g.

```
Package: randomForest
Title: Breiman & Cutler's random forests
Version: 4.6-10
Date: 2014-07-17
Depends: R (>= 2.5.0), stats
Suggests: RColorBrewer, MASS
Author: Fortran original by Leo Breiman and Adele C
        Andy Liaw and Matthew Wiener.
etc...
```

List of all fields can be found at cran.r-project.org under *"Writing R Extensions"*

NAMESPACE

Specifies R code/packages/objects/classes/methods to be imported or exported. e.g.

```
import(Biobase)
useDynLib(randomForest)
export(combine, getTree, grow, importance, margin, 
       partialPlot, randomForest, rfImpute, treesiz
       varImpPlot, varUsed, rfNews, outlier, classC
S3method(print, randomForest)
S3method(predict, randomForest)
S3method(plot, randomForest)
etc...
```

Sealed once package is installed, though hidden function can still be called with "`package:::function`"

Package documentation

Files stored in "man" subfolder and displayed anytime a user calls `help("function")`. Format similar to Latex, e.g. for `grow.RD` from `randomForest`:

```
\name{grow}
\alias{grow}
\alias{grow.default}
\alias{grow.randomForest}
\title{Add trees to an ensemble}
\description{
  Add additional trees to an existing ensemble of t
}
etc...
```

List of all fields can be found at cran.r-project.org under *"Writing R Extensions"*

data directory

Stores data sets that can be loaded with package

- Stored as ".rda" or ".RData" files
- Can be called with `data("name")` once package is loaded
- Documentation files can also be created for these saved data sets
    - Details for documenting can be found at cran.r-project.org under *"Writing R Extensions"*

Creating packages

Now that we understand what a package is, how do we create them?

1. Manually creating each file / subfolder
2. Using the `package.skeleton()` function
   - Sets up directory / file templates for us
3. Using the `devtools` R-package
   - Sets up directory / file templates for us
   - Developed by Hadley Wickham to facilitate package creation
   - Well documented from his website: http://r-pkgs.had.co.nz
   - My preferred approach

Creating packages

Now that we understand what a package is, how do we create them?

1. Manually creating each file / subfolder
2. Using the `package.skeleton()` function
    - Sets up directory / file templates for us
3. Using the `devtools` R-package
    - Sets up directory / file templates for us
    - Developed by Hadley Wickham to facilitate package creation
    - Well documented from his website: http://r-pkgs.had.co.nz
    - My preferred approach

Creating packages

Now that we understand what a package is, how do we create them?

1. Manually creating each file / subfolder
2. Using the `package.skeleton()` function
   - Sets up directory / file templates for us
3. Using the `devtools` R-package
   - Sets up directory / file templates for us
   - Developed by Hadley Wickham to facilitate package creation
   - Well documented from his website: http://r-pkgs.had.co.nz
   - My preferred approach

Creating packages

Now that we understand what a package is, how do we create them?

1. Manually creating each file / subfolder
2. Using the `package.skeleton()` function
    - Sets up directory / file templates for us
3. Using the `devtools` R-package
    - Sets up directory / file templates for us
    - Developed by Hadley Wickham to facilitate package creation
    - Well documented from his website: http://r-pkgs.had.co.nz
    - My preferred approach

DEVTOOLS

devtools

Using devtools allows us many functions which streamline different stages of creating an R package. e.g.

- `create("path/to/package/pkgname")`
  - Creates parent directory for package
- `load_all("path/to/package/pkgname")`
  - Loads package without having to build
- `document("path/to/package/pkgname")`
  - Uses Roxygen to document R code / data sets
- `use_vignette("filename")`
  - Creates vignette directory and file

Let's see it in action!

Simulation

Let's do a quick simulation to show that the estimator solving the efficient influence function has lower variance. Given

- $O = (W, A, Y) \sim P_0 \in \mathcal{M}$
- $P_0(O) = P_0(W)P_0(A|W)P_0(Y|A, W)$, where
  - $W \sim N(0, .75^2)$
  - $A|W \sim Ber(logit^{-1}(-0.5 - 1.5 * W))$
  - $Y|A, W \sim Ber(logit^{-1}(-0.75 + 1.2 * W - 0.75 * A))$

Our target parameter is

$$\Psi(P_0) = \mathbb{E}_W[\mathbb{E}_{P_0}[Y|A = 1, W]] \approx 0.2149 \quad (1)$$

Estimators

Estimators we'll be evaluating

- g-computation

$$\hat{\psi}_n^{gcomp} = \mathbb{E}_n \bar{Q}_n(Y|A=1,W)$$

- Inverse Probability Treatment Weights (IPTW)

$$\hat{\psi}_n^{IPTW} = \mathbb{E}_n \left[Y \frac{\mathbb{I}(A=1)}{g_n(A|W)} \right]$$

- Augmented-IPTW (A-IPTW)

$$\hat{\psi}_n^{AIPTW} = \mathbb{E}_n \left[(Y - \bar{Q}_n(Y|A=1,W)) \frac{\mathbb{I}(A=1)}{g_n(A|W)} \right] + \bar{Q}_n(Y|A=1,W)$$

nb. It would be very beneficial to work out the Efficient Influence Curve on your own.

VIGNETTES

Vignettes

A way of storing author customized documentation to support package

- Can be stored in multiple (pre-compiled) formats including
  - R markdown
  - Sweave
  - knitr
- Currently compiled to pdf at build time
- Can be called from R console

Let's look at examples and make one.