Surrogate Evaluation in R

Session 8

Dean Follmann, Peter Gilbert, Betz Halloran, Erin Gabriel,
Michael Sachs

July 19, 2016

About R

- `R` is a programming language for and by statisticians
- Open-source, free
- Design features (for statisticians) and quirks (by statisticians) aimed at data analysis

Download and stay up to date

- R: `https://cran.r-project.org`
- Rstudio: `https://www.rstudio.com/products/rstudio/download/`

Packages

Packages

- Groups of functions/data are organized into *packages*
- Some packages come with base `R`
- External sources:
  - CRAN: `https://cran.r-project.org/web/packages`
  - RForge: `https://r-forge.r-project.org/`
  - Bioconductor: `https://www.bioconductor.org/`
  - Github: `https://github.com`
  - Personal websites
  - ...

Disclaimer

- Packages are community-developed (base R excepted)
- CRAN only verifies code is organized correctly and doesn't do anything harmful
  - Does not check validity!
  - Bioconductor has a few more requirements
- *"How do I do x in R?"*
  - Is the package written by someone you know and trust?
  - Is it peer-reviewed in R Journal or JSS?
  - Is it current, and actively updated?
  - When in doubt, view the source, or contact the author...

Ultimately it is the users responsibility to verify the validity of their analysis.

Installation

From CRAN:

```
install.packages("pseval")
```

From Source:

```
install.packages("download.zip", repos = NULL, type = "sour
```

From Github:

```
devtools::install_github("sachsmc/pseval")
```

Loading

Functions defined in a package can be referenced by `packagename::functionname`
This can get cumbersome, so we often “attach” the package to the namespace:

```
pseval::psdesign
survival::Surv

library("pseval")
library("survival")
```

Then any function can be called directly (without the `::`)

```
psdesign
Surv
```

Objects and environments

Everything is an object

- Objects live in an *environment*
  - A group of objects in memory
  - "Global environment" is what we generally work in
- Objects are generally created by *functions*
  - Functions take objects as input, do something, then output other objects
- Objects have one or more *class*
  - The class determines how functions and operators interact with the object

Types of objects

- Vectors

```r
1:5
```

```
## [1] 1 2 3 4 5
```

```r
LETTERS[1:5]
```

```
## [1] "A" "B" "C" "D" "E"
```

```r
c(TRUE, FALSE, FALSE)
```

```
## [1]  TRUE FALSE FALSE
```

Objects

- Matrices

```
matrix(1:9, nrow = 3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
matrix(letters[1:9], nrow = 3)
```

```
##      [,1] [,2] [,3]
## [1,] "a"  "d"  "g"
## [2,] "b"  "e"  "h"
## [3,] "c"  "f"  "i"
```

Objects

- Data frames

```r
head(mtcars)
```

```
##                    mpg cyl disp  hp drat    wt  qsec
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02
## Valiant           18.1   6  225 105 2.76 3.460 20.22
```

Other

- Lists
- Functions
- ...

Data frames

- A data frame is a collection of vectors of objects, where each vector is the same length
- Rows = observations, columns = variables
- Variables can be different types

```
df <- data.frame(X = 1:3, Y = letters[1:3], Z = c(TRUE,
```

- Can refer to variables by name

```
df$X
```

```
## [1] 1 2 3
```

```
df$Y
```

```
## [1] a b c
## Levels: a b c
```

  - "Look for object `X` in `df`"

Operators and assignment

Operators

- Special functions
  - One (unary) or two (binary) inputs

```
?data.frame
help(data.frame)

-1
```

```
## [1] -1
```

```
`-`(1)
```

```
## [1] -1
```

Binary

```r
1 + 2
```

```
## [1] 3
```

```r
`+`(1, 2)
```

```
## [1] 3
```

```r
2 < 1
```

```
## [1] FALSE
```

```r
`<`(2, 1)
```

```
## [1] FALSE
```

What other kinds of objects can you add or compare?

```r
1:5 + 1
```

```
## [1] 2 3 4 5 6
```

```r
1:5 + 1:5
```

```
## [1]  2  4  6  8 10
```

```r
1:3 < 2:4
```

```
## [1] TRUE TRUE TRUE
```

```r
"a" < "b"
```

```
## [1] TRUE
```

Assignment

- Special assignment operator: `<-`

```
x <- 1.0
`<-`(x, 1.0)
```

“Store 1.0 in the environment and call it ‘x’ ”

```
df$N <- LETTERS[1:3]
```

Functions

Functions

Calling a function

```
function_name(arg1.name = arg1.value, arg2.name = arg2.valu
```

1. Function name is always unquoted
2. Don't forget open and close parentheses

Arguments

Arguments are key=value pairs separated by commas

```
function_name(arg1.name = arg1.value, arg2.name = arg2.valu
```

1. Arguments are matched by name or position
2. Argument names are always unquoted
3. A function may not have any arguments
4. Optional or unnamed arguments ...
5. Sometimes arguments have defaults
6. All specified in a function's help file

Return

- Most functions return an object
- Details in the “Value” section of the help file

Functions may behave differently based on what objects are given as arguments

Formulas

Formulas

- Special way to describe relationships between variables

```
Y ~ X + Y + Z + Y:Z
```

1. Outcome to the left of `~`, predictors to the right
2. Linear combinations separated by `+`
3. Interactions with `:`
4. `Y * Z` expands to `Y + Z + Y:Z`

Some details

- Variables in a formula are names of objects in a data frame or environment
- How does R know where to find the objects?

```
lm(mpg ~ wt)
lm(mpg ~ wt, data = mtcars)
```

- Use functions in a formula

```
lm(mpg ~ log(wt), data = mtcars)
lm(mpg ~ wt^2, data = mtcars)
```

Loading Data

Lots of options

- Base R functions
  - `read.table`, `read.csv`
- Packages
  - `foreign`, `readxl`
- Easy way

```r
install.packages("rio")
rio::import("data.csv")
rio::import("data.xlsx")
```

Getting Help

How **not** to ask for help

It doesn't work, what do I do?

Before asking for help

Do your homework:

- Read the error or warning message
- Read help files, documentation
- Make sure all software is up to date
- Search first: `https://stackoverflow.com/questions/tagged/r`

How to ask for help

1. State what you are trying to do
2. Find the minimal reproducible example that produces the error/problem
3. Describe or write the code that you used
4. Describe what you expected the result to be
5. Describe how the actual result differs from your expectation

Exercises

Install

Install the `pseval` package:

- `https://cran.r-project.org/package=pseval`

Read about and download one of the example data sets:

- `https://sachsmc.github.io/pseval-course`

Exercises

1. Create `psdesign` object appropriate to the study design
2. Add integration model to the object
3. Add risk model appropriate to the study and outcome
4. Fit the model with EML
5. Bootstrap using starting values from step 4.
6. Create a plot of the CEP that is of interest
7. Extract the appropriate statistics for tests of WEM from the model fit
8. Use a different integration model to see if it affects the results
9. Write up results in a way suitable for a clinical journal, including a plot
10. Bonus: make a plot using `ggplot2`