The Fundamentals of S3 Objects and Methods

Kenny Flagg
useR Bozeman
1 March 2017

https://github.com/kflagg/useRS3

Outline

- What is object-oriented (OO) programming?
- Some familiar classes of objects
- Writing methods to work with objects
- Case study: extending the `htest` class

What is object-oriented (OO) programming?

- System of programming that models relationships among abstract structures.
- Simplifies development and maintainence of complicated programs. (R Core Team 2017)
- Concepts:
  - *classes* define standard structures to store information.
  - *objects* are realized instances of classes.
  - *methods* are functions that act on objects according to their classes.
- R has several different systems of defining objects.
  - We will work with the S3 system, the least formal and by far most common.

What is object-oriented (OO) programming?

I'm not developing packages. Why should I learn this programming mumbo-jumbo?

- Understanding others' code/debugging.
- Recycling your own code.

Some familiar classes of objects

- You have probably used objects of these classes:
  - `data.frame`
  - `lm`
  - `glm`
  - `htest`
- Let's use the `str()` function to investigate!

```
str(iris)

versicolor <- glm(I(Species == 'versicolor') ~
                  Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
                   family = binomial, data = iris)
str(lm(Petal.Length ~ Species, data = iris))

petal <- t.test(iris$Petal.Length, mu = 4)
str(petal)
```

Some familiar classes of objects

```
str(iris)

# 'data.frame': 150 obs. of  5 variables:
#  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
#  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
#  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
#  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
#  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1
```

Some familiar classes of objects

```
versicolor <- glm(I(Species == 'versicolor') ~
                  Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
                   family = binomial, data = iris)
str(versicolor)

# List of 30
#  $ coefficients     : Named num [1:5] 7.378 -0.245 -2.797 1.314 -2.778
#   ..- attr(*, "names")= chr [1:5] "(Intercept)" "Sepal.Length" "Sepal.Wid
#  $ residuals        :Class 'AsIs'  Named num [1:150] -1.09 -1.39 -1.21 -1
#   .. ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
#  $ fitted.values    : Named num [1:150] 0.0849 0.2829 0.172 0.268 0.0671
#   ..- attr(*, "names")= chr [1:150] "1" "2" "3" "4" ...
#  $ effects          :Class 'AsIs'  Named num [1:150] 2.383 0.739 4.822 -0
#   .. ..- attr(*, "names")= chr [1:150] "(Intercept)" "Sepal.Length" "Sepa
...
#  - attr(*, "class")= chr [1:2] "glm" "lm"
```

Some familiar classes of objects

```
petal <- t.test(iris$Petal.Length, mu = 4)
str(petal)

# List of 9
#  $ statistic  : Named num -1.68
#   ..- attr(*, "names")= chr "t"
#  $ parameter  : Named num 149
#   ..- attr(*, "names")= chr "df"
#  $ p.value    : num 0.0953
#  $ conf.int   : atomic [1:2] 3.47 4.04
#   ..- attr(*, "conf.level")= num 0.95
#  $ estimate   : Named num 3.76
#   ..- attr(*, "names")= chr "mean of x"
#  $ null.value : Named num 4
#   ..- attr(*, "names")= chr "mean"
#  $ alternative: chr "two.sided"
#  $ method     : chr "One Sample t-test"
#  $ data.name  : chr "iris$Petal.Length"
#  - attr(*, "class")= chr "htest"
```

Some familiar classes of objects

- Many objects are actually lists with a `class` attribute.
- What happens when you change a data frame's class to "list"?

```
dframe <- data.frame(x = 1:4, y = 5:8, row.names = letters[1:4])
dframe
class(dframe) <- 'list'
dframe
```

Some familiar classes of objects

```
dframe <- data.frame(x = 1:4, y = 5:8, row.names = letters[1:4])
dframe

#   x y
# a 1 5
# b 2 6
# c 3 7
# d 4 8

class(dframe) <- 'list'
dframe

# $x
# [1] 1 2 3 4
#
# $y
# [1] 5 6 7 8
#
# attr(,"row.names")
# [1] "a" "b" "c" "d"
```

Writing methods to work with objects

- Objects are nouns.
- Methods are verbs.
  - They tell the computer what to do to objects.
- S3 methods are functions with different versions defined for each class.
  - `print`, `summary`, `plot` all behave according to the class of the object.

Writing methods to work with objects

- You can see what methods are available for a class.

```
methods(class = 'lm')
```

- Are there things you didn't know you could do with `lm` objects?

Writing methods to work with objects

- What do methods look like? (Recall you can usually type the name of a function to see its source code.)

```
summary
summary.data.frame
summary.lm
summary.default
```

- Some source code is hidden, but you can often find it with `getS3method`.

```
plot.lm
getS3method('plot', 'lm')
```

Writing methods to work with objects

- Generic methods are defined with `UseMethod`.

```
method <- function(x, ...) UseMethod('method')
```

- Methods for specific classes are named `method.class`.
- If a method isn't defined for a class, R uses `method.default`.

Case study

Let's extend the `htest` class to handle a one-sample simulation test!
Then we'll write a method to plot a histogram of the sampling distribution.

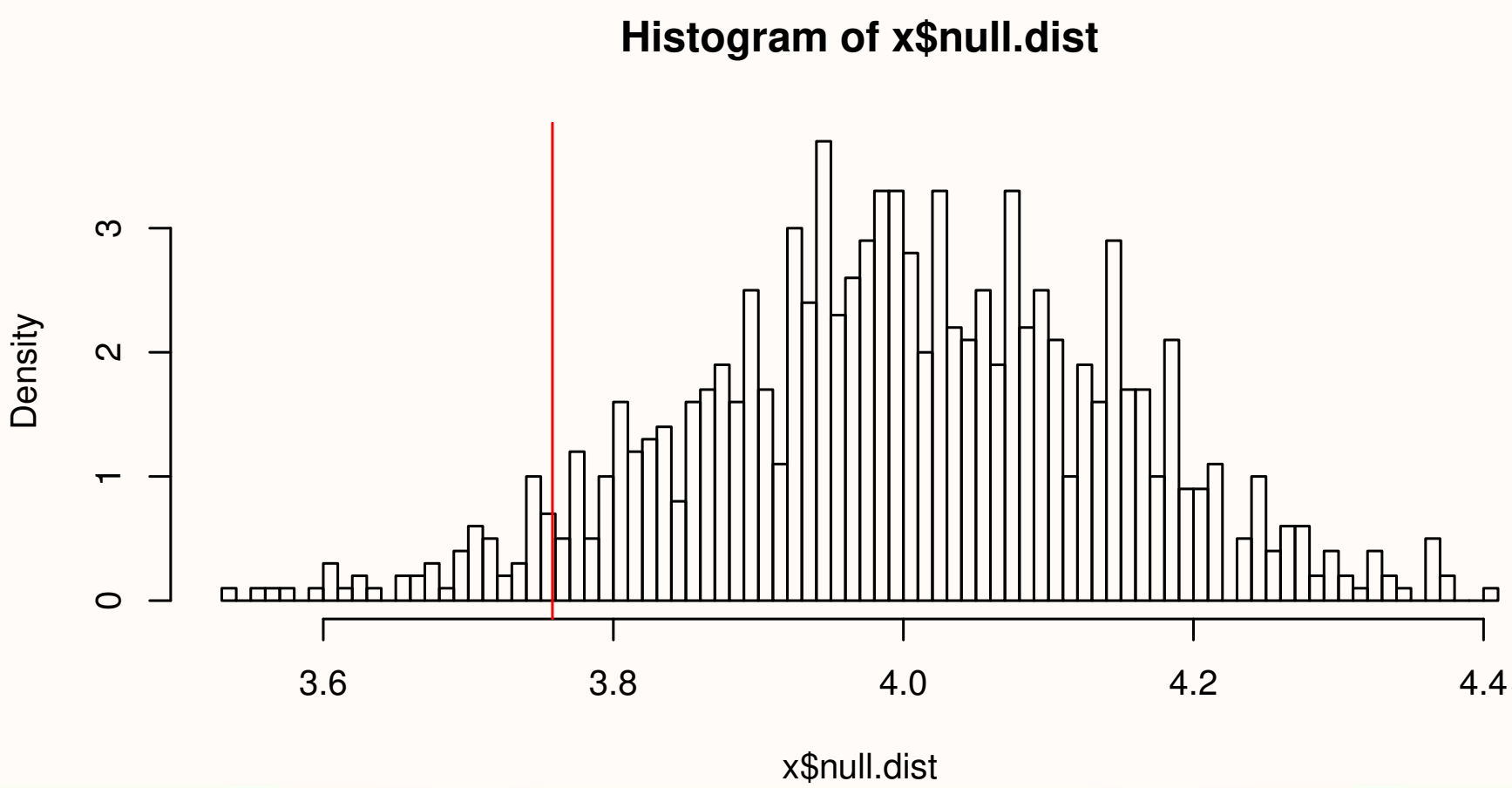
```
result <- sim.test(iris$Petal.Length, mu = 4)

# This uses the print.htest function.
result


#
#  One Sample Simulation Test
#
# data:  iris$Petal.Length
# number of simulations = 1000, p-value = 0.944
# alternative hypothesis: true  is less than 4
# sample estimates:
# mean of x
#     3.758
```

Case study

```
# This uses the plot.simtest function.
plot(result, breaks = 100, freq = FALSE)
```



Case study

- Write a function that does a one sample simulation test. Try it out on your favorite dataset.
- I put some example code to do a simulation test on GitHub: `https://github.com/kflagg/useRS3/blob/master/code.r`
- At the end of the function, combine the variables into a list (use the variable names in the code below).
- Set the class to a vector containing `"htest"` and a second class like `"simtest"`. Try running your function with and without setting the class.

```
result <- list(
  # These are standard elements of htest objects.
  estimate = ...,        # An estimate of the mean.
  parameter = ...,       # Optional. I put the number of simulations here.
  p.value = ...,         # Your p-value.
  null.value = ...,      # The value of the mean under the null hypothesis.
  alternative = ...,     # The text "less" for a left-tailed test.
  method = ...,          # A name for the method, like "Simulation Test"
  data.name = ...,       # The name of the data vector (text).

  # This is something new for the simtest class.
  null.dist = ...        # The vector of simulated sample means.
)

class(result) <- c(...) # Make the class "htest" and "simtest".
```

Case study

- Write a function that
  - takes at least one argument (a `simtest` object),
  - plots a histogram of the object's `null.dist` variable,
  - and draws a vertical line at the observed value (the `estimate`).
- Name the function `plot.simtest`.

```
plot.simtest <- function(...){
  ...
}

plot(result)
```

Summary

Today we learned...

- How object-oriented programming works in R.
  - Classes are created by changing `class(object) <- ...`.
  - Methods are functions named like `method.class`.
  - You run `method(object)` and R will automatically use `method.class(object)`.
- How some familiar objects are structured.
  - Most are just lists.
- How to create a new class and write a method.

References

R Core Team (2017). *R Language Definition*. Version 3.4.2. URL: https://cran.r-project.org/doc/manuals/r-release/R-lang.pdf.