Rbootcamp

Michael Kleinsasser

8/8/2019

Mike's Personal Introduction

- R programmer for the Department of Biostatistics
- Write and mantain R packages for faculty and students
- Consult faculty and students on writing R packages

Rbootcamp Introduction

Goals:

- R basics: syntax, common functions, etc., via Rstudio
- R functions and for loops (functions and advanced control structures)
- Basics of R on the cluster (non-interactive R using BATCH scripts)
- Simulation analysis with comment on efficiency

Materials

- All bootcamp materials online at https://github.com/umich-biostatistcs/Rbootcamp
- Handouts for each topic with examples to work through
- R scripts of our examples

Setup

Go to Rstudio cloud to follow along:

- Enter username, etc. for free account
- Follow along by typing commands in my slides
- If you have Rstudio/R, open that
- Recommended: Install R/Rstudio for days 2, 3

R Basics: Big Picture

- R is a sophisticated calculator for statistics

Chambers (2016) Extending R:

- Everything that exists in R is an object
- Everything that happens in R is a function call

Obtain a basic working knowledge of R objects and functions,

- Google the rest

R Basics: Goals

- Data types and functions
    - Create object having data types
    - Combine those into data structures
    - Write basic R function
- Learn parts of R most useful to statisticians
    - How do most modeling functions work in R, and
    - How to inspect structure and content of objects

Use R as a calculator

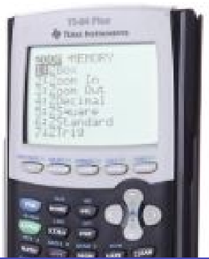
Standard operations:

```
# multiply *, divide /, add + subtract
18056.983 - 1005.118 + 22.53
```

```
## [1] 17074.4
```

```
( (pi - 3.14) / (3.14) )  * 100
```

```
## [1] 0.05072145
```



Assignment in R

```r
# x gets the number 3.14
x <- 3.14

x     # print x
```

```
## [1] 3.14
```

```r
# equivalently
x = 3.14

x     # print x
```

```
## [1] 3.14
```

R data types/structures

Five fundamental data types

- character, numeric, integer, logical, complex NOTE: insert table/picture of functions with examples

Combine to form data structures

- atomic vector (atomic - vector of single type)
- list
- matrix
- data.frame
- factor

R data types/structures

Five fundamental data types

- character, numeric, integer, logical, complex NOTE: insert table/picture of functions with information

R data types/structures

Combine to form data structures

- atomic vector (atomic - vector of single type)
- list
- matrix
- data.frame
- factor NOTE: insert table/picture of functions with information

Data types examples

3 ways to create numeric vector:

```
# empty numeric vector
y1 <- numeric(6)
y1     # print y1
```

```
## [1] 0 0 0 0 0 0
```

```
y2 <- vector(mode = "numeric", length = 6)
y2     # print y2
```

```
## [1] 0 0 0 0 0 0
```

```
y3 <- c(5, 13.222, 2, 0.001, 77.4, 31.9)
y3     # print y3
```

```
## [1]  5.000 13.222  2.000  0.001 77.400 31.900
```

Data structures examples

Create a data.frame out of the following “class” data:

- Has Master’s (logical): TRUE FALSE FALSE TRUE
- GPA (numeric): 3.1 4.0 2.9 3.6
- First Name (character): Mike Dan Sara Karen

```r
# store data
has_ms <- c(TRUE, FALSE, FALSE, TRUE)
gpa <- c(3.1, 4.0, 2.9, 3.6)
name <- c("Mike", "Dan", "Sara", "Karen")
# Create data.frame
dat <- data.frame(has_MS = has_ms, GPA = gpa, Name = name)
dat    # print data.frame
```

```
##   has_MS GPA  Name
## 1   TRUE 3.1  Mike
## 2  FALSE 4.0   Dan
## 3  FALSE 2.9  Sara
## 4   TRUE 3.6 Karen
```

Inspect an object

- Class() - what kind of object is it (high-level)?
- typeof() - what is the data type (low-level)?
- length() - how long is it?
- attributes() - does it have meta-data?

R functions

R function syntax:

```
NAME <- function(ARG1, ARG2, ARG3) {
      DO SOMETHING
      STORE RESULT
      return(RESULT)
}
```

```r
pow <- function(base, expon) { # power function
  prod(rep(base, expon)) # base^(expon)
}
# Use power function
pow(5, 2)
```

```
## [1] 25
```

```r
pow(10, 3)
```

```
## [1] 1000
```

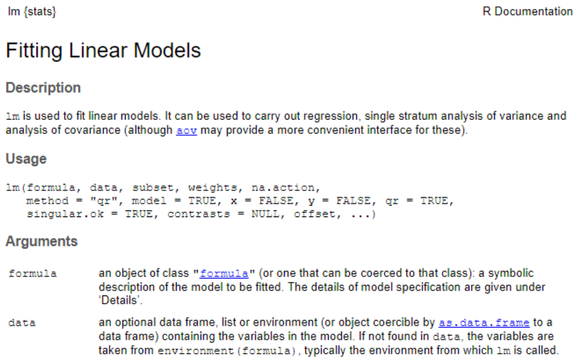
Common R functions

R has a huge collection of packages:

- 6,000+ packages for data analysis build (on CRAN alone)

Example: lm (linear models)

- Use ?lm to read help documentation



lm {stats} R Documentation

Fitting Linear Models

Description

`lm` is used to fit linear models. It can be used to carry out regression, single stratum analysis of variance and analysis of covariance (although `aov` may provide a more convenient interface for these).

Usage

```
lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

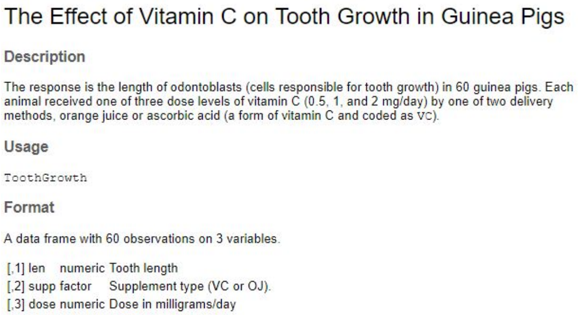
Arguments

`formula` an object of class "`formula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.

`data` an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `lm` is called.

Fit a linear model with lm

- Use built-in data set ToothGrowth
- ?ToothGrowth for help:



The Effect of Vitamin C on Tooth Growth in Guinea Pigs

Description

The response is the length of odontoblasts (cells responsible for tooth growth) in 60 guinea pigs. Each animal received one of three dose levels of vitamin C (0.5, 1, and 2 mg/day) by one of two delivery methods, orange juice or ascorbic acid (a form of vitamin C and coded as VC).

Usage

```
ToothGrowth
```

Format

A data frame with 60 observations on 3 variables.

[,1]	len	numeric	Tooth length
[,2]	supp	factor	Supplement type (VC or OJ).
[,3]	dose	numeric	Dose in milligrams/day

View the data

View data in new window:

```
View(ToothGrowth)
```

Or use head to view only first 6 rows:

```
head(ToothGrowth)
```

```
##    len supp dose
## 1  4.2   VC  0.5
## 2 11.5   VC  0.5
## 3  7.3   VC  0.5
## 4  5.8   VC  0.5
## 5  6.4   VC  0.5
## 6 10.0   VC  0.5
```

How big is the data?

```
dim(ToothGrowth)
```

```
## [1] 60  3
```

Using lm() function for linear models

Call lm on the data and formula, store result “lm” object:

```
tooth_fit = lm(formula = len ~ supp + dose,
               data = ToothGrowth)
```

Formulas in R:

```
len ~       # Response column name, ~ for "="
  supp +    # First predictor name + for "+"
  dose      # second predictor name
```

Many R functions use the formula argument.

Getting detailed information

Basic “print” of model:

```
print(tooth_fit)     # equivalent to tooth_fit
```

```
##
## Call:
## lm(formula = len ~ supp + dose, data = ToothGrowth)
##
## Coefficients:
## (Intercept)       suppVC         dose
##       9.272       -3.700        9.764
```

Detailed summary:

```
summary(tooth_fit)
```

```
##
## Call:
## lm(formula = len ~ supp + dose, data = ToothGrowth)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.600 -3.700  0.373  2.116  8.800
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   9.2725     1.2824   7.231 1.31e-09 ***
## suppVC       -3.7000     1.0936  -3.383   0.0013 **
## dose          9.7636     0.8768  11.135 6.31e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Understanding R classes

- What is this thing?

```
class(tooth_fit)
```

- What are the methods for this object?

```
methods(class = "lm")
```

- What is its structure? (i.e., what's in it)

```
str(tooth_fit)
```

Understanding R classes

- What is this thing?

```
class(tooth_fit)
```

```
## [1] "lm"
```

Understanding R classes

- What are the methods for this object?

```
methods(class = "lm")
```

```
##  [1] add1           alias          anova          case.names    
##  [5] coerce         confint        cooks.distance deviance      
##  [9] dfbeta         dfbetas        drop1          dummy.coef    
## [13] effects        extractAIC     family         formula       
## [17] hatvalues      influence      initialize     kappa         
## [21] labels         logLik         model.frame    model.matrix  
## [25] nobs           plot           predict        print         
## [29] proj           qr             residuals      rstandard     
## [33] rstudent       show           simulate       slotsFromS3   
## [37] summary        variable.names vcov          
## see '?methods' for accessing help and source code
```

Understanding R classes

- What is its structure? (i.e., what's in it)

```
str(tooth_fit)
```

```
## List of 13
##  $ coefficients : Named num [1:3] 9.27 -3.7 9.76
##   ..- attr(*, "names")= chr [1:3] "(Intercept)" "suppVC" "dose"
##  $ residuals    : Named num [1:60] -6.25 1.05 -3.15 -4.65 -4.05 ...
##   ..- attr(*, "names")= chr [1:60] "1" "2" "3" "4" ...
##  $ effects      : Named num [1:60] -145.73 14.33 47.16 -3.86 -3.26 ...
##   ..- attr(*, "names")= chr [1:60] "(Intercept)" "suppVC" "dose" "" ...
##  $ rank         : int 3
##  $ fitted.values: Named num [1:60] 10.5 10.5 10.5 10.5 10.5 ...
##   ..- attr(*, "names")= chr [1:60] "1" "2" "3" "4" ...
##  $ assign       : int [1:3] 0 1 2
##  $ qr           :List of 5
##   ..$ qr   : num [1:60, 1:3] -7.746 0.129 0.129 0.129 0.129 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:60] "1" "2" "3" "4" ...
##   .. .. ..$ : chr [1:3] "(Intercept)" "suppVC" "dose"
##   .. ..- attr(*, "assign")= int [1:3] 0 1 2
##   .. ..- attr(*, "contrasts")=List of 1
##   .. .. ..$ supp: chr "contr.treatment"
##   ..$ qraux: num [1:3] 1.13 1.11 1.11
##   ..$ pivot: int [1:3] 1 2 3
##   ..$ tol  : num 1e-07
##   ..$ rank : int 3
##   ..- attr(*, "class")= chr "qr"
##  $ df.residual  : int 57
##  $ contrasts    :List of 1
##   ..$ supp: chr "contr.treatment"
##  $ xlevels      :List of 1
```

Understanding R classes

- Pull something out of the “lm” fit object:

```
tooth_fit$fitted.values         # y_hat's for the linear model
```

```
##        1        2        3        4        5        6        7        8
## 10.45429 10.45429 10.45429 10.45429 10.45429 10.45429 10.45429 10.45429
##        9       10       11       12       13       14       15       16
## 10.45429 10.45429 15.33607 15.33607 15.33607 15.33607 15.33607 15.33607
##       17       18       19       20       21       22       23       24
## 15.33607 15.33607 15.33607 15.33607 25.09964 25.09964 25.09964 25.09964
##       25       26       27       28       29       30       31       32
## 25.09964 25.09964 25.09964 25.09964 25.09964 25.09964 14.15429 14.15429
##       33       34       35       36       37       38       39       40
## 14.15429 14.15429 14.15429 14.15429 14.15429 14.15429 14.15429 14.15429
##       41       42       43       44       45       46       47       48
## 19.03607 19.03607 19.03607 19.03607 19.03607 19.03607 19.03607 19.03607
##       49       50       51       52       53       54       55       56
## 19.03607 19.03607 28.79964 28.79964 28.79964 28.79964 28.79964 28.79964
##       57       58       59       60
## 28.79964 28.79964 28.79964 28.79964
```

Extracting data from model objects

Some generic extraction methods:

```
coef(tooth_fit)                # model coefficients

coef(summary(tooth_fit))       # adds test statistics, p-values

vcov(tooth_fit)                # variance/covariance matrix
```

Note: depending on implementation, these may not be available - Check methods with “methods(object)” before attempting

Extracting data from model objects (in detail)

Extract coefficients, test stats, and p-values

```
coef(summary(tooth_fit))    # adds test statistics, p-values
```

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)  9.272500  1.2823649  7.230781 1.312335e-09
## suppVC      -3.700000  1.0936045 -3.383307 1.300662e-03
## dose         9.763571  0.8768343 11.135025 6.313519e-16
```

Predict new values

- predict() function is generic and works with many models
- Pass in a new data.frame with the same column names:

```
to_predict = data.frame(dose = 0.5, supp = "VC")
```

```
predict(tooth_fit, newdata = to_predict)
```

```
##        1
## 10.45429
```

Predict new values (example 2)

- predict() function is generic and works with many models
- Pass in a new data.frame with the same column names:

```
to_predict = data.frame(dose = seq(0,1,0.1), supp = "OJ")
```

```
predict(tooth_fit, newdata = to_predict)
```

```
##        1        2        3        4        5        6        7
##  9.27250 10.24886 11.22521 12.20157 13.17793 14.15429 15.13064 16
##        9       10       11
## 17.08336 18.05971 19.03607
```

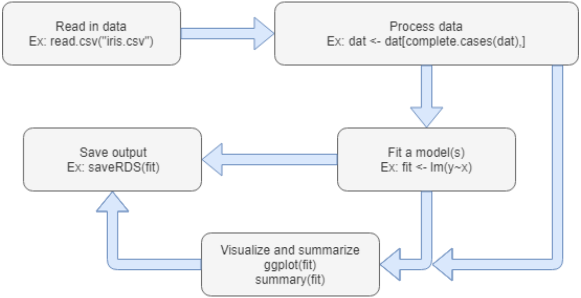
R on cluster (non-interactive R)

Cluster Computation

- Dan Barker danbarke at umich.edu
- Cluster System Administrator

R workflow for statistical analysis

- Bootcamps throughout semester on each area



Read in data
Ex: read.csv("iris.csv")

Process data
Ex: dat <- dat[complete.cases(dat),]

Fit a model(s)
Ex: fit <- lm(y~x)

Save output
Ex: saveRDS(fit)

Visualize and summarize
ggplot(fit)
summary(fit)

Advanced control structures

Common misconception: “loops in R are slow”

Explain

Loops: often not necessary

- Many R functions are “vectorized” (vector in, vector out)
- While most other languages require loops, R does not:

```
a = c(5, 2, 4, 12, 1)
b = c(2, 0, 3, -1, 2)

a + b      # vector + vector = vector
```

```
## [1]  7  2  7 11  3
```

Takehome: When possible, operate on vectors and matrices, don't loop over each row/position index

Problem: not all functions are vectorized

- Some functions, like read.table() for reading a table of data into R, are not vectorized
- But what if we have a list of files to read in? “data1.txt”, “data2.txt”, “data3.txt”, . . . , “data50.txt”

```
file_names = paste0("data", 1:50, ".txt")  # the 50 data set
```

Attempt it, will cause error:

```
read.table(file_names)  # error!
```

map() from purrr package can help

- map() allows you to apply a function to each element of a vector
- faster, easier to read than a loop

```
read.table(file_names)  # error!
```

```
    # list of 50 data sets
my_dat_list = map(file_names, read.table)
    # results from reading data1.txt
my_dat_list[[1]]
    # results from reading data50.txt
my_dat_list[[2]]
```

Simulation

Simulate lm

Need to come up with some question to answer.

Bootstrap lm results

Useful R packages

- ggplot2 - Best package for plotting data
  - getting started:
- dplyr - Best package for manipulating/processing data
  - getting started:

Future bootcamps

- dplyr: date _____
- ggplot2: date _____