The R Language

A Hands-on Introduction

Venkatesh-Prasad Ranganath

http://about.me/rvprasad

What is R?

- A dynamical typed programming language
  - http://cran.r-project.org/
  - Open source and free
- Provides common programming language constructs/features
  - Multiple programming paradigms
- Numerous libraries focused on various data-rich topics
  - http://cran.r-project.org/web/views/
- Ideal for statistical calculation; lately, the go-to tool for data analysis
- Accompanied by RStudio, a simple and powerful IDE
  - http://rstudio.org

Data Types (Modes)

- Numeric
- Character
- Logical (TRUE / FALSE)
- Complex
- Raw (bytes)

Data Structures

- Vectors
- Matrices
- Arrays
- Lists
- Data Frames
- Factors
- Tables

Data Structures: Vectors

- A sequence of objects of the same (atomic) data type
- Creation
  - `x <- c('a', 'b', 'c')` [<- is the assignment operator]
  - `y <- seq(5, 9, 2) = c(5, 7, 9)`
  - `y <- 5:7 = c(5, 6, 7)` [`m:n` is equivalent to `seq(m, n, 1)`]
  - `y <- c(1, 4:6) = c(1, 4, 5, 6)` [no nesting / always flattened]
  - `z <- rep(1, 3) = c(1, 1, 1)`

Data Structures: Vectors

- Accessing
  - `x[1] = c("a")` [1-based indexing]
  - `x[2:3] = c("b", "c")`
  - `x[c(2,3)] = x[2:3]`
  - `x[-1] = c("b", "c")` [Negative subscripts imply exclusion]
- Naming
  - `names(x) <- c('f1', 'f2', 'f3')` [Makes `x['f1']` equivalent to `x[1]`]

Data Structures: Vectors

- Operations
  - `x <- c(5, 6, 7)`
  - `x + 2 = c(7, 8, 9)`
  - `x > 5 = c(FALSE, TRUE, TRUE)`
  - `subset(x, x > 5) = c(6, 7)`
  - `which(x > 5) = c(2, 3)`
  - `ifelse(x > 5, NaN, x) = c(5, NaN, NaN)`

  - `sqr <- function (n) { n * n }`
  - `sapply(x,sqr) = c(25 ,36, 49)`
  - `sqr(x) = c(25, 36, 49)`

[Vectorized operations]

Data Structures: Vectors

- Operations
  - `x <- c(5, 6, 7)`
  - `any(x > 5) = TRUE` [How about `all(x > 5)`?]
  - `sum(c(1, 2, 3, NA), na.rm = TRUE) = 6` [Why is `na.rm` required?]
  - `sort(c(7, 6, 5)) = c(5, 6, 7)`
  - `order(c(7, 6, 5)) = ???`
  - `subset(x, x > 5) = c(6, 7)`
  - `head(1:100) = ???`
  - `tail(1:100) = ???`
  - How is `x == c(5, 6, 7)` different from `identical(x, c(5, 6, 7))`?
  - Try `str(x)`

Data Structures: Matrices

- A two dimensional matrix of objects of the same (atomic) data type
- Creation
  - `y <- matrix(nrow=2, ncol=3)` [empty matrix]
  - `y <- matrix(c(1, 2, 3, 4, 5, 6), nrow=2)` =

1	3	5
2	4	6

  - `y <- matrix(c(1, 2, 3, 4, 5, 6), nrow=2, byrow=T)` =

1	2	3
4	5	6

- Accessing
  - `y[1,2] = 2`
  - `y[,2:3]` =

2	3
5	6

[How about `y[1,]`?]

  - What's the difference between `y[2,]` and `y[2,, drop=FALSE]`?

Data Structures: Matrices

- Naming
  - `rownames()` and `colnames()`
- Operations
  - `nrow(y) = 2` [number of rows]
  - `ncol(y) = 3` [number of columns]
  - `apply(y, 1, sum) = c(6, 15)` [apply `sum` to each row]
  - `apply(y, 2, sum) = c(5, 7, 9)` [apply `sum` to each column]
  - `t(y) =` [transpose a matrix]

1	4
2	5
3	6

Data Structures: Matrices

- Operations
  - `rbind(y, c(7, 8, 9))` =

1	2	3
4	5	6
7	8	9

  - `cbind(y, c(7, 8))` =

1	2	3	7
4	5	6	8

  - Try `str(y)`

Data Structures: Matrices

- What will this yield?

```
m <- matrix(nrow=4, ncol=4)
m <- ifelse(row(m) == col(m), 1, 0.3)
```

Data Structures: Lists

- A sequence of objects (of possibly different data types)
- Creation
  - `k <- list(c(1, 2, 3), c('a', 'b'), c(5.0, 6.0))`
  - `l <- list(f1=c(1, 2, 3), f2=c('a', 'b'))` [f1 and f2 are tags]
- Accessing
  - `k[2:3] = list(c('a', 'b'), c(5.0, 6.0))`
  - `k[[2]] = c('a', 'b')` [How about `k[2]`?]
  - `l$f1 = c(1, 2, 3)` [Is it same as `l[1]` or `l[[1]]`?]

Data Structures: Lists

- Naming
  - `names(k) <- c("id", "name", "value")`
- Operations
  - `lapply(list(1:2, 9:10), sum) = list(3, 19)`
  - `sapply(list(1:2, 9:10), sum) = c(3, 19)`
  - `l$f1 <- NULL = ???`
  - `str(l) = ???`

Data Structures: Data Frames

- A two dimensional matrix of objects where different columns can be of different types.
- Creation
  - `x <- data.frame(names=c('jack', 'jill'), age=c(5, 6))`
- Accessing
  - `x$names = c('jack', 'jill')` [How about `x[[1]]`?]
  - `x[1] = ???`
  - `x[c(1,2)] = ???`
  - `x[1,] = ???`
  - `x[,1] = ???`

Data Structures: Data Frames

- Naming
  - `rownames()` and `colnames()`
- Operations
  - `x[x$age > 5,] = data.frame(names=c('jill'), age=c(6))`
  - `subset(x, age > 5) = ???`
  - `apply(x, 1, sum) = ???`

  - `y <- data.frame(1:3, 5:7)`
  - `apply(y, 1, mean) = ???`
  - `lapply(y, mean) = ???`
  - `sapply(y, mean) = ???`
  - Try `str(y)`

Factors (and Tables)

- Type for categorical/nominal values.
- Example
  - `xf <- factor(c(1:3, 2, 4:5))`
  - Try `xf` and `str(xf)`
- Operations
  - `table(xf)` = ???
  - `with(mtcars, split(mpg, cyl))` = ???
  - `with(mtcars, tapply(mpg, cyl, mean))` = ???
  - `by(mtcars, mtcars$cyl, function(m) { median(m$mpg) }` = ???
  - `aggregate(mtcars, list(mtcars$cyl), median)` = ???
  - You can use `cut` to bin values and create factors. Try it.

Basic Graphs

- `with(mtcars, boxplot(mpg))`
- `hist(mtcars$mpg)`
- `with(mtcars, plot(hp, mpg))`
- `dotchart(VADeaths)`
- Try `plot(aggregate(mtcars, list(mtcars$cyl), median))`

You can get the list of datasets via `ls("package.datasets")`

Stats 101 using R

- `mean`
- `median`
- *What about mode?*
- `fivenum`
- `quantile`
- `sd`
- `var`
- `cov`
- `cor`

Data Exploration using R

Let’s get out hands dirty!!