R Scripting

657.019 Scripting for Biotechnologists (WS 2018/19)

Outline

- Example statements
- Basic statistics
- Comparisons
- Vectors
- User-defined functions
- Plotting
- Matrices and data frames
- Simple statistical analysis
- Packages

How to use R

- As an interpreter (shell) environment
  - Linux command: **`R`**
  - Within R session, type a command (function call), hit Enter
  - Environment used today

- As a script file
  - With a shebang line (in our server): **`#!/bin/Rscript`**
  - Do **`chmod +x`** and run like Python or Bash scripts

R statement examples

```
> a <- 100
> b <- 100
> c <- a + b
> ls()
> help(ls)
> getwd()
> rm(a)
> a
> ls()
> d <- c(10.4, 5.6, 3.1, 6.4, 21.7)
> x <- c(d, a, b, c, d)
> ls()
> rm(list=ls())
> ls()
> mean(d)
> example(mean)
```

Three key things to remember

- Assignment
  - `<-`
- Function call
  - `funname(arg1, arg2, ...)`
- List creation by combining objects
  - `c(arg1, arg2, ...)`

Basic statistics

```
> x <- c(0,1,1,2,3,5,8,13,21,34)
> y <- log(x+1)
> mean(x)
> median(x)
> sd(x)                    # standard deviation
> var(x)                   # variance
> cor(x, y)                # correlation
> cov(x, y)                # covariance
```

Creating sequences

```
> 1:5                    # from 1 to 5 (by 1)
> seq(1, 5)              # from 1 to 5 (by 1)
> seq(1, 5, by=2)        # step by 2
> seq(length=5)          # seq has 5 elements
> seq(0, 20, length=5)
> s <- seq(0, 20, length=5)
> rep(pi, 5)             # repeat pi 5 times
```

- R counts from 1, not from 0 (as in Python or Perl)
- Start and end indices are inclusive

Comparisons

```
> v <- c(3, pi, 4)
> w <- c(pi, pi, pi)
> v == w                    # compare element-wise
> v != w
> v < w
> v <= w
> v > w
> v == pi
> v != pi
```

Vectors

```
> fib <- c(0,1,1,2,3,5,8,13,21,34) # Fibonacci
> fib
> fib[1]                  # 0:first index is 1
> fib[2]
> fib[4:9]                # indices 4 to 9
> fib[c(1,2,4,8)]         # indices 1, 2, 4, 8
> v=fib
```

- Select all elements greater than the median
  - `v[v > median(v)]`
- Select all elements in the lower and upper 5%
  - `v[(v < quantile(v,0.05))|(v > quantile(v,0.95))]`
- Select all elements more than 2 standard deviations away from the mean
  - `v[abs(v-mean(v)) > 2*sd(v)]`

Vectors and names

```
> years <- c(1960, 1964, 1976, 1994)
> names(years) <- c("Kennedy", "Johnson”,
  "Carter", "Clinton")
> years              # now has indices AND names
> Years[1]
> years["Carter"]
> years["Clinton"]
```

- Vectors can have names, making them accessible by names (keys)

User-defined functions

- Coefficient of variation

```
> cv <- function(x) sd(x)/mean(x)
> cv(1:10)
```

- Greatest common divisor

```
> gcd <- function(a,b) {          # start
+ if (b==0) return(a)             # continue
+ else return(gcd(b, a%%b))       # continue
+ }                                # end
> gcd(4,16)                       # call
```

Plotting

```
> X <- c(1:10)
> Y <- c(21:30)
> plot(X,Y)
> barplot(X)
> boxplot(X)
> K <- c(X,X,Y,Y,Y,Y,Y)
> hist(K, freq=FALSE)   # density histogram
> example(hist)
> plot(log2, 0, 7, type="b") # log2(x), 0-7
> example(plot)
```

Matrices

```
> theData <- c(10, 9, 8, 7, 6, 5)
> mat <- matrix(theData, 2, 3) # 2 rows, 3 cols
> mat
> mat <- matrix(theData, 2, 3, byrow=TRUE) #row first
> mat
> mat <- matrix(theData, 3)              # 3 rows
> mat
> mat <- matrix(theData, ncol=3)         # 3 cols
> mat
> dim(mat)            # dimension: nrow ncol
> mat + 4             # add 4 to all elements
> mat + mat           # element by element sum
> mat * mat           # element by element product
> mat %*% mat         # matrix product, error
> mat %*% t(mat)      # matrix product, with transpose
```

Data frames

- Holds table data
- Save the following in **people.txt** file:

```
Lastname Firstname Born Died
Fisher R.A. 1890 1962
Pearson Karl 1857 1936
Cox Gertrude 1900 1978
Yates Frank 1902 1994
Smith Kirstine 1878 1939
```

```
> dfrm <- read.table("people.txt", header=TRUE)
> # Default uses whitespaces to separate fields
> dfrm
> dfrm[2]
> dfrm[2,]
> dfrm[,2]
> dfrm[1:3,3:4]
```

Data frames

- Save the following in **bounds.csv** file:

```
Label,Lowerbound,Upperbound
low,0,0.674
mid,0.674,1.64
high,1.64,2.33
```

```
> dfrm2 = read.csv("bounds.csv")
> dfrm2                  # should have used header=TRUE
```

- If the table file was tab-delimited:

```
> dfrm <- read.table("bounds.tsv", sep="\t")
```

Simple statistical analysis

- Two vectors, x and y, that hold paired observations: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.
- We believe there is a linear relationship between x and y, and want to create a **linear regression** model of the relationship

```
> x <- c(10:20)
> y <- c(20:30)
> lm(y~x)          # fit linear model
> plot(lm(y~x))    # plot more info
```

Packages

- To list currently installed packages:

```
> library()
```

- To list currently loaded packages:

```
> search()
```

- Many packages are not part of standard R installations
  - Need to install additionally
  - And load to use them
- Bioconductor set of packages (www.bioconductor.org)
  - Packages with R functions to analyze biological data sets

Using Bioconductor packages

- Install a core set of Bioconductor packages:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite()
```

- Install an additional package as needed:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("DESeq2")
```

- Whenever you want to use a package after installing it, you need to load it into R:

```
> library("DESeq2")
```

Summary

- R is a language and also an environment for statistical computing and graphics.
- Mainly used for data manipulation, statistical analysis, plotting, among other things.
- Many R packages exist for biological data analysis.