The 'R' statistics environment

Bio14559  Thurs, March 26, 2015
Bill Pearson  wrp@virginia.edu  4-2818  Jordan 6-057

• A quick introduction to 'R'
  – Variable types:
    vector=c(0,1,2,3),
    mat1 = matrix(vector,nrow=2) (or ncol=2)
    dframe1 = data.frame(ved=vector,
                          vecx2= vector*2, vecsq=vector**2)
  – Input:
    read.table("filename",header=TRUE,sep="\t")
  – Output:
    plot(), hist(), boxplot()
  – Running 'R' ('R'-studio)

To learn more:

1. An introduction to 'R':
   cran.r-project.org/doc/manuals/R-intro.pdf
2. A "short" introduction:
   cran.r-project.org/doc/contrib/Torfs+Brauer-Short-R-Intro.pdf
3. Introducing 'R':
   http://data.princeton.edu/R/introducingR.pdf
4. A different introductory lecture on 'R' (that I borrow from):
   http://www.stat.cmu.edu/~cshalizi/statcomp/13/lectures/01--02/
      lecture-01--02.pdf
Why 'R'?

- Open source, statistical programming environment based on 'S' (Bell Labs statistical programming environment)
  - plotting functions, statistical distributions, summary statistics, linear models, etc., etc.
- Universally used for functional bioinformatics (Bioconductor)
- The standard platform for new statistical development (false discovery rate fdr/qvalue)
- Tools for program documentation/reproducibility (knitr)
- 'R' analyses on the WWW (shiny)

Introduction to 'R' – functional programming

Python is an object oriented "procedural" language. You specify in some detail how to read data into variables, which are then iterated on, or transformed in some way, or used to automate a task.

'K' is a functional language. In some sense, everything in 'R' happens to a vector.

Thus, in Python, to make square all the values in a vector (array), you might write:

```python
>>> array = [1, 2, 3, 4, 5]
>>> [x * x for x in array]
[1, 4, 9, 16, 25]
>>> [2 * x for x in array]
[2, 4, 6, 8]
```

in 'R':

```r
> vector
[1] 1 2 3 4
> vector^2
[1] 1 4 9 16
> 2*vector
[1] 2 4 6 8
```

while there are 'for()' loops and 'if/then/else' conditionals in 'R', you will almost never need them to use 'R'. You will need to define functions, and use "apply()" to apply a function to the values in a vector.
Introduction to 'R' – data types

• data types:
  – numbers: 1, 1.0, 12.345
    numbers are always double precision floating point unless forced to integer with as.integer()
  – boolean: TRUE, FALSE
    boolean values can be used to retrieve entries in vectors
    > v1<-1:10
    > v1
    [1]  1  2  3  4  5  6  7  8  9 10
    > v1<4
    [1] TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
    > v1[v1 < 4]
    [1] 1 2 3
  – characters: "Jane", "pre-cancerous"
  – NaN, NA - special no-data types

Introduction to 'R' – variable types

• Variable types:
  – vectors[] : arrays of the same type (number, string)
    v1 <- c(1,2,3,4)
    v12 <- c(v1,v1) -> 1 2 3 4 1 2 3 4 # c() "flattens"
    v2 <- 1:9
    v3 <- seq(1.5,0.1)
  – matrices[2,3] : arrays of arrays (of arrays), multi-dimensional
    mat1 <- matrix(1:9, nrow=3)
    mat2 <- matrix(1:9,nrow=3,byrow=TRUE)
    mat1
    [,1] [,2] [,3]
    [1,]  1 4 7
    [2,]  2 5 8
    [3,]  3 6 9
    mat2
    [,1] [,2] [,3]
    [1,]  1 2 3
    [2,]  4 5 6
  – lists[] : array that can have different types, including vectors and lists, has named entries (like dictionary)
  – data.frame[] : like a matrix with named columns (like dictionary), can contain different types
Introduction to 'R' – vector subsets

Selecting and sub-selecting data: vectors

- sub-part of vectors can be selected with vectors of indices
  ```r
  v1 <- c(1.1, 2.2, 4.3, 3.4, 5.5)
  v1[2, 3]
  Error in v1[2, 3] : incorrect number of dimensions
  v1[c(2,3)]
  # indices must be in vector
  [1] 2.2 4.3
  v1[c(2,4,3)]
  # indices can re-order
  [1] 2.2 3.4 4.3
  v1[-c(2,3)]
  # negative index deletes selection (cannot combine)
  [1] 1.1 3.4 5.5
  v1[order(v1)]
  # the order() function returns the indexes to sort
  [1] 1.1 2.2 3.4 4.3 5.5
  ```

- sub-parts of vectors can be selected using booleans (TRUE, FALSE)
  ```r
  v1 <- 1:10
  v1 <= 5
  [1] TRUE TRUE TRUE TRUE TRUE FALSE FALSE FALSE FALSE FALSE
  v1[v1<=5]
  [1] 1 2 3 4 5
  v1%%2 == 0
  [1] FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE FALSE TRUE
  v1[v1%%2==0]
  [1] 2 4 6 8 10
  ```

- in all of these examples, sub-setting a vector returned a vector.

Introduction to 'R' – matrix subsets

Selecting and sub-selecting data: matrices
  ```r
  mat1 <- matrix(1:12, nrow=3)
  mat1
  [1,]  1  4  7 10
  [2,]  2  5  8 11
  [3,]  3  6  9 12
  mat1[2,]
  # select all columns from one row
  [1]  2  5  8 11
  mat1[,4]
  # select all rows from one column
  [1] 10 11 12
  mat1[,4]^2
  # compute on resulting vector
  [1] 100 121 144
  mat1[1:2, 3:4]
  # for matrices, vectors select entries
  [,1] [,2]
  [1,]  7 10
  [2,]  8 11
  mat1[cbind(3, 1), cbind(3,1,2,4)]
  [1,]  9  3  6 12
  [2,]  7  1  4 10
  ```
Introduction to 'R' – variable types

• Selecting and sub-selecting data: matrices

```r
> mat1 <- matrix(1:12, nrow=3)
> mat1
[1,]  1  4  7 10
[2,]  2  5  8 11
[3,]  3  6  9 12
> mat1[mat1[2,]>=5,]
Error in mat1[mat1[2,]>=5,] (subscript)
logical subscript too long
mat1[2,]>=5
[1] FALSE  TRUE  TRUE  TRUE
> mat1[,mat1[2,]>=5]
[1,]  4  7 10
[2,]  5  8 11
[3,]  6  9 12
> mat1[,mat1[,2]>5]
[1]  7  8  9
> mat1[,mat1[,2]<5]
[,1] [,2]
[1,]  1 10
[2,]  2 11
[3,]  3 12
```

Introduction to 'R' – data.frames

• data.frames are tables (arrays) with different types, typically with labeled columns

```r
> head(GSE_FPKM)
Gene  MCF.7_Rep1  MCF.7_Rep2  MCF.7_Rep3  GM12892_Rep1  GM12892_Rep2  GM12892_Rep3
1  1/2-SBSRNA4  0.54253200  0.318766   0.2925300   0.268225   0.50125500   0.4364100
2    A1BG   0.75134200  1.080660  1.3224700   2.389740   0.42191900   0.5300680
3  A1BG-AS1  0.90314900  0.549146  1.5402100   0.701192   0.12630800   0.6629410
4    A1CF  0.00176153   0.000000  0.0000000  0.000000   0.00385721   0.0000000
5   A2LD1  1.37068000   1.040530  1.1445600   2.341310   2.41900000   1.8365700
6     A2M  0.00716990  1.435170   0.0510643  0.0000000  0.00385721   0.0000000
```

• typically, columns of the data are extracted by name (GSE_FPKM$MCF.7_Rep1) as vectors, but they can also be extracted by index (GSE_FPKM[,2])

```r
> head(GSE_FPKM[order(GSE_FPKM$MCF.7_Rep1, decreasing=TRUE),])
Gene  MCF.7_Rep1  MCF.7_Rep2  MCF.7_Rep3  GM12892_Rep1  GM12892_Rep2  GM12892_Rep3
17769 RPL41  9479.40  5999.73  8669.86  8774.13  5197.96  4536.55
17833 RPS29  6909.02  3113.50  3847.84 10579.00  7282.94  5614.69
17829 RPS27  5281.44  2321.00  2883.32 10689.70  9748.79  7855.76
17765 RPL39  5217.51  2396.75  2294.83  6122.56  5146.11  4554.45
```

```r
> head(GSE_FPKM[order(GSE_FPKM$MCF.7_Rep1, decreasing=TRUE),])
Gene  MCF.7_Rep1  MCF.7_Rep2  MCF.7_Rep3  GM12892_Rep1  GM12892_Rep2  GM12892_Rep3
17769 RPL41  9479.40  5999.73  8669.86  8774.13  5197.96  4536.55
17833 RPS29  6909.02  3113.50  3847.84 10579.00  7282.94  5614.69
17829 RPS27  5281.44  2321.00  2883.32 10689.70  9748.79  7855.76
17765 RPL39  5217.51  2396.75  2294.83  6122.56  5146.11  4554.45
```

```r
> head(GSE_FPKM[order(GSE_FPKM$MCF.7_Rep1, decreasing=TRUE),])
Gene  MCF.7_Rep1  MCF.7_Rep2  MCF.7_Rep3  GM12892_Rep1  GM12892_Rep2  GM12892_Rep3
17769 RPL41  9479.40  5999.73  8669.86  8774.13  5197.96  4536.55
17833 RPS29  6909.02  3113.50  3847.84 10579.00  7282.94  5614.69
17829 RPS27  5281.44  2321.00  2883.32 10689.70  9748.79  7855.76
17765 RPL39  5217.51  2396.75  2294.83  6122.56  5146.11  4554.45
```
Introduction to 'R' – variables

• to see what is in a variable, use: `str()

```r
> str(v1)
num [1:5] 1.1 2.2 4.3 3.4 5.5
> str(mat1)
int [1:3, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
> str(GSE_FPKM)
data.frame: 23197 obs. of 11 variables:
  $ Gene : Factor w/ 21648 levels "1/2-SBSRNA4",...
  $ MCF.7_Rep1: num 0.54253 0.75134 0.90315 0.00176 1.37068 ...
  $ MCF.7_Rep2: num 0.319 1.081 0.549 0 1.041 ...
  $ MCF.7_Rep3: num 0.293 1.322 1.54 0 1.145 ...
  $ GM12892_Rep1: num 0.268 2.39 0.701 0 2.341 ...
  $ GM12892_Rep2: num 0.50126 0.42192 0.12631 0.00386 2.419 ...
  $ GM12892_Rep3: num 0.436 0.53 0.663 0 1.837 ...
  $ H1.hESC_Rep1: num 0.6699 2.43029 0.42874 0.00798 0.40421 ...
  $ H1.hESC_Rep2: num 0.60306 2.65009 0.37343 0.00259 0.68117 ...
  $ H1.hESC_Rep3: num 0.54942 2.23051 0.44545 0.00053 0.50608 ...
  $ H1.hESC_Rep4: num 0.4247 1.199 0.5754 0.0125 0.6244 ...
```
### Reading in datasets (data.frame(s))

- for tab delimited files with headers:
  ```
  Gene  MCF-7_Rep1   MCF-7_Rep2   MCF-7_Rep3  
  1/2-SB  0.542532  0.318766   0.292530 
  A1BG    0.751342  1.080660   1.322470 
  A1BG-   0.903149  0.549146   1.540210 
  
  • you can read directly into a data.frame[] with read.table():
    ```
    ```
    > GSE_FPKM <- read.table('GSE49712_ENCODE_FPKM.txt', header=TRUE, sep="\t")
    ```
    ```
    > head(GSE_FPKM)
    ```
    ```
    Gene MCF.7_Rep1 MCF.7_Rep2 MCF.7_Rep3 GM12892_Rep1 GM12892_Rep2 GM12892_Rep3
    1 1/2-SBSRNA4 0.54253200   0.318766  0.2925300     0.268225   0.50125500    0.4364100
    2        A1BG 0.75134200   1.080660  1.3224700     2.389740   0.42191900    0.5300680
    3    A1BG-AS1 0.90314900   0.549146  1.5402100     0.701192   0.12630800    0.6629410
    4        A1CF 0.00176153   0.000000  0.0000000     0.000000   0.00385721    0.0000000
    5       A2LD1 1.37068000   1.040530  1.1445600     2.341310   2.41900000    1.8365700
    6         A2M 0.00716990   1.435170  0.0510643     0.137600   0.03139180    0.0299176
    ```
    ```
  - If every column is not labeled, you may get an error:
    ```
    Error in read.table("GSE49712_ENCODE_FPKM.txt", header = TRUE, sep = "\t") : duplicate 'row.names' are not allowed
    ```
    ```
  - If you do not have a header, you can provide names:
    ```
    > fpe = read.table("noheader.dat", +
    + col.names=c("setting","effort","change")) # + for continuation
    ```
    ```
```

### Plotting data

One of the great strengths of 'R' is its ability to plot data in many different ways (this is also why you will be running it on your laptop, rather than on franklin.achs from the command line)

- x-y plots : plot(x-vector, y-vector)
  ```
  > high_samps <- GSE_FPKMSMCF.7_Rep1 > 100
  ```
  ```
  > plot(MCF.7_Rep1[high_samps], MCF.7_Rep2[high_samps],log="xy")
  ```
Plotting data

- histograms: `hist(vector)`
  ```r
  > hist(log(MCF.7_Rep1[MCF.7_Rep1 > 10]))
  ```

- boxplots `boxplot(vector1, vector2, vector3)`
  ```r
  > boxplot(log(GSE_FPKM[GSE_FPKM[2:4]>100,2:4]))
  ```
'R' functions

Functions may have arguments specified or unspecified when the function is defined

- There may be an arbitrary number of unspecified arguments
- Unspecified arguments denoted by ...
- Specified arguments may be supplied in the same order in which they occurred in the function definition
- Specified arguments may be supplied as name=value in which case their order is not important

```r
> help(t.test) # if you know the name of the R built in function, you can use help()
> x = rnorm(10) # 10 numbers randomly drawn from a normal distribution; x ~ N(0, 1)
> y = rnorm(10) # 10 numbers randomly drawn from a normal distribution; y ~ N(0, 1)
> t.test(x, y, "greater") # arguments in same order in which they are defined in function
> t.test(x=x, alternative="greater", y=y) # argument names specified but in wrong order
Welch Two Sample t-test data: x and y
t = 1.1862, df = 16.896, p-value = 0.1260 alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
-0.2838161  Inf
sample estimates:
mean of x  mean of y
0.02149336 -0.58618035
```

Contributed Packages: Currently, the CRAN package repository has more than 1700 packages:

 Croatian packages: Currently, the CRAN package repository has more than 1700 packages:

 Specialized packages implementing the latest methods developed in computational statistics.

Use help() for assistance on usage!

`fasta.bioch.virginia.edu/biol4559`
'R' functions – apply()

The apply() function allows you to apply functions, like mean() or var(), which apply to a vector, to a row (or row subset) of a matrix or data.frame.

```r
> GSE_FPKM[11:15,2:4]
MCF.7_Rep1 MCF.7_Rep2 MCF.7_Rep3
11 0.000000 0.0000000
12 0.014162 0.0000000
13 29.783700 38.1064000
14 20.810500 21.7803000
15 0.104898 0.0610452
> var(GSE_FPKM[13,2:4])  # does NOT work – should report one variance per row
MCF.7_Rep1 MCF.7_Rep2 MCF.7_Rep3
MCF.7_Rep1 NA NA NA
MCF.7_Rep2 NA NA NA
MCF.7_Rep3 NA NA NA
> apply(GSE_FPKM[13,2:4],1,var)  # does work – variance of row 13 is 56.42433
13 56.42433
> apply(GSE_FPKM[11:15,2:4],1,var)  # five rows, five variances
11 12 13 14 15
0.000000e+00 6.685408e-05 5.642433e+01 4.477427e+01 2.775529e-03
```

Introduction to 'R'

- 'R' works on vectors, matrices, and data.frames()
- subsets of vectors/matrices/data.frames can be specified:
  - vectors of indices (c(4,3,1,2), order(v1))
  - boolean vectors ( $rep1>10 & rep2 > 10)
  - [,1:3] : all rows, columns 1:3
  - [1:4,] : all columns, rows 1:4
- columns of data.frames() can be named or indexed
- read.table()
- plot, hist, boxplot