Additional exercesis R bioinfo programming. Master ISCIII. 2014-2015. Second set.

Ramon Diaz-Uriarte*

2014-11-19 (Release 1.1: Rev: b2ab80f)

Contents

1 Recommendations

2 Purpose of the exercises

The purpose of this exercise is to get used to obtaining summary statistics from data sets and from subpieces of those data sets.

3 General hints

You will use the leukemia data (including the data, the class and the sex objects we used in the exercise for figures). Here are those instructions again:

3.1 The data

- Read the data "leukemia.data.txt". Note that there are row names that we do not want to be part of the data itself (look at the help for "read.table", the option "row.names"). Call this "leuk.dat"These data are based on the famous leukemia data from Golub et al. I've modified a few things below: the gene names and the sex of the patients are invented.

*Dept. of Biochemistry, Universidad Autónoma de Madrid, Spain, http://ligarto.org/rdiaz, rdiaz02@gmail.com

- That will be a data frame. For most figures your life will be simpler if you convert it into a matrix, using `data.matrix`. Call this data matrix "leuk.dat.m". (Why is it a good idea to convert this into a matrix? Try doing the figures using the original data frame.)
- Read the data "leukemia.class.txt". Use scan for this (do you know why?). Since these are labels, use `what = ""` in `scan`. Convert the classes into a factor. Call this "leuk.class".
- Create a vector for the sex of the patients. The patients are Male and Female, alternating, and staring with Male. Call this factor "sex".

4 Tables and cross-tabs

Obtain a simple table of how many males and females are there. Use `table` on the "sex" object:

```
## sex
## Female   Male
##     19     19
```

Now, a table of how many of each sex are there in each class, or viceversa. This requires you to pass two arguments to `table`

```
##         leuk.class
## sex      ALL AML
##   Female  13   6
##   Male    14   5
```

(if your table looks transposed with respect to the above ... well, change the order of arguments).

Now, do the same as above, but with a formula interface, which some times is more useful. Use the `xtabs` function

```
##         leuk.class
## sex      ALL AML
##   Female  13   6
##   Male    14   5
```

Finally, rerun the code from xtabs and assign that to an object, call it "mytable"
and convert mytable to a data frame using `as.data.frame` and print it:

```
as.data.frame(mytable)
```

```
##      sex leuk.class Freq
## 1 Female        ALL   13
## 2   Male        ALL   14
## 3 Female        AML    6
## 4   Male        AML    5
```

Notice that this is just a very simple but extremely powerful way of getting frequencies and saving them as a data frame for further analysis!! With the output from the tables we could do a chi-square. Or we could use the data frame for some other statistical analysis (e.g., logistic regressions, etc). Oh and for real we do not need to save all those intermediate objects.

5 Two subsetting operations

Since getting access to selected parts of matrices, data frames, etc, is so important, two quick exercises.

First, obtain the mean of all the genes for the third subject, but only for those genes with a p-value less than 0.01. Or, to put it another way, you want to see the average (mean) expression of the genes in subject 3, but only for the genes that have a p-value < 0.01:

```
## [1] 0.01816
```

And now the median of the second gene, but ONLY for males

```
## [1] -1.045
```

6 Gene summaries by condition and sex

The next exercises all involve summaries of specific genes. Remember that genes were originally the rows of the data set. Since we will be using aggregate and similar, you will want to have genes as columns. How? Doing a transposition:

```
leuk.dat.t <- t(leuk.dat.m)
```

Make sure you understand what happened. For instance, do:

```
dim(leuk.dat.t)

## [1]   38 3051
```

and now check the first few columns (you do not want to use `head`, because it would show all 3051 columns)

```
leuk.dat.t[1:5, 1:6]

##         G1       G2       G3     G4     G5      G6
## V2 -1.458 -0.75161  0.45695 3.1353  2.766  2.6434
## V3 -1.394 -1.26278 -0.09654 0.2142 -1.270  1.0142
## V4 -1.428 -0.09052  0.90325 2.0875  1.604  1.7048
## V5 -1.407 -0.99596 -0.07194 2.2347  1.532  1.6384
## V6 -1.427 -1.24245  0.03232 0.9381  1.637 -0.3608
```

6.1 aggregate: The median of three genes by condition

Use aggregate to obtain the median of genes in positions 1, 2124, and 2600 (HK-1, PTEN, and the other gene from the figures in the exercises from last week)

```
##   type     G1   G2124   G2600
## 1  ALL -1.361 -0.2167 -0.1101
## 2  AML -1.230  1.5763  1.3253
```

6.2 aggregate: The median of three genes by condition and sex

Like the previous exercise, but you want to obtain those summaries by condition and sex:

```
##   type    sex     G1    G2124    G2600
## 1  ALL Female -1.368  0.07727 -0.00771
## 2  AML Female -1.320  1.71842  1.13218
## 3  ALL   Male -1.321 -0.46043 -0.16511
## 4  AML   Male -1.089  1.45710  1.55580
```

6.3 aggregate: The median of all the genes by condition and sex

This is like the previous one, but for **every** gene. So you do not want to print it. Assign the result of aggregate to an object, lets call it "all.median".

Now, show all the rows and the first 10 columns

```
##   type    sex     G1      G2      G3     G4      G5      G6    G7
## 1  ALL Female -1.368 -1.0321 0.23381 0.6418  0.2285  0.9687 3.288
## 2  AML Female -1.320 -0.9223 0.14707 0.3602 -0.5765 -0.1641 3.126
## 3  ALL   Male -1.321 -1.0346 0.26632 2.2069  1.7145  1.6147 3.087
## 4  AML   Male -1.089 -1.0890 0.04609 1.3030  1.0160  0.5127 3.279
##      G8
## 1 3.040
## 2 2.901
## 3 2.985
## 4 2.961
```

and the dimensions of "all.median"

```
dim(all.median)

## [1]    4 3053
```

6.4 aggregate: The mean and standard deviations of three genes by condition and sex

This is like the above ones, but we do not want just one statistic (the median); here we want the mean and standard deviation for all our three genes (1, 2124, 2600). And, if possible, with clear column names, so you will have to be explicit, when you pass a function to `aggregate`, about the names of the components you return. (In other words, DO NOT assign the return value of aggregate and change the column names; you have to play around with the names in the return object from the function you pass to aggregate)

```
##   type    sex G1.mean   G1.sd G2124.mean G2124.sd G2600.mean
## 1  ALL Female -1.3180  0.1567    -0.1633   0.5816   -0.23135
## 2  AML Female -1.1570  0.3504     1.6656   0.3161    1.30764
## 3  ALL   Male -1.2283  0.4254    -0.4169   0.8363   -0.09673
## 4  AML   Male -0.3260  1.1930     1.4919   0.4386    1.42469
##   G2600.sd
## 1  0.63892
## 2  0.50587
## 3  0.78109
## 4  0.40976
```

6.5 by and aggregate: those three genes again, but now use "summary"

As it says. First, do not use `aggregate` now, but use `by`, and the function you want to have applied to each subset is `summary`, which will give you different summary statistics.

```
## type: ALL
## sex: Female
##        G1            G2124             G2600
##  Min.   :-1.61   Min.   :-1.3683   Min.   :-1.4009
##  1st Qu.:-1.40   1st Qu.:-0.4010   1st Qu.:-0.6645
##  Median :-1.37   Median : 0.0773   Median :-0.0077
##  Mean   :-1.32   Mean   :-0.1633   Mean   :-0.2313
##  3rd Qu.:-1.22   3rd Qu.: 0.2502   3rd Qu.: 0.1770
##  Max.   :-1.03   Max.   : 0.6263   Max.   : 0.5734
```

```
## ------------------------------------------------
## type: AML
## sex: Female
##        G1              G2124          G2600
##  Min.   :-1.444   Min.   :1.23   Min.   :0.816
##  1st Qu.:-1.410   1st Qu.:1.46   1st Qu.:0.929
##  Median :-1.320   Median :1.72   Median :1.132
##  Mean   :-1.157   Mean   :1.67   Mean   :1.308
##  3rd Qu.:-0.894   3rd Qu.:1.81   3rd Qu.:1.728
##  Max.   :-0.665   Max.   :2.11   Max.   :1.977
## ------------------------------------------------
## type: ALL
## sex: Male
##        G1              G2124            G2600
##  Min.   :-1.568   Min.   :-1.476   Min.   :-1.2305
##  1st Qu.:-1.427   1st Qu.:-1.225   1st Qu.:-0.5857
##  Median :-1.321   Median :-0.460   Median :-0.1651
##  Mean   :-1.228   Mean   :-0.417   Mean   :-0.0967
##  3rd Qu.:-1.234   3rd Qu.: 0.392   3rd Qu.: 0.3977
##  Max.   : 0.176   Max.   : 0.579   Max.   : 1.4982
## ------------------------------------------------
## type: AML
## sex: Male
##        G1              G2124          G2600
##  Min.   :-1.262   Min.   :1.05   Min.   :0.814
##  1st Qu.:-1.230   1st Qu.:1.19   1st Qu.:1.218
##  Median :-1.089   Median :1.46   Median :1.556
##  Mean   :-0.326   Mean   :1.49   Mean   :1.425
##  3rd Qu.: 0.849   3rd Qu.:1.58   3rd Qu.:1.737
##  Max.   : 1.101   Max.   :2.18   Max.   :1.800
```

(Of course, if you wanted, you could do that to all of the genes in the array)
Now, use aggregate:

```
##   type    sex G1.Min. G1.1st Qu. G1.Median G1.Mean G1.3rd Qu.
## 1  ALL Female  -1.610     -1.400    -1.370  -1.320     -1.220
## 2  AML Female  -1.440     -1.410    -1.320  -1.160     -0.894
## 3  ALL   Male  -1.570     -1.430    -1.320  -1.230     -1.230
## 4  AML   Male  -1.260     -1.230    -1.090  -0.326      0.849
##   G1.Max. G2124.Min. G2124.1st Qu. G2124.Median G2124.Mean
## 1  -1.030    -1.3700       -0.4010       0.0773    -0.1630
## 2  -0.665     1.2300        1.4600       1.7200     1.6700
## 3   0.176    -1.4800       -1.2300      -0.4600    -0.4170
## 4   1.100     1.0500        1.1900       1.4600     1.4900
##   G2124.3rd Qu. G2124.Max. G2600.Min. G2600.1st Qu. G2600.Median
## 1        0.2500     0.6260   -1.40000      -0.66400     -0.00771
## 2        1.8100     2.1100    0.81600       0.92900      1.13000
## 3        0.3920     0.5790   -1.23000      -0.58600     -0.16500
## 4        1.5800     2.1800    0.81400       1.22000      1.56000
##   G2600.Mean G2600.3rd Qu. G2600.Max.
## 1   -0.23100       0.17700    0.57300
## 2    1.31000       1.73000    1.98000
## 3   -0.09670       0.39800    1.50000
## 4    1.42000       1.74000    1.80000
```

Please, notice the differences in output. They are different kinds of objects, something you can see if you assign the output to an object and do, for instance

```
class(objeto)
```