HPBBM. Exercises R. Summaries, tables, aggregate, by.

Ramon Diaz-Uriarte[*]

2016-11-17 (Release: Rev: fe4c981)

Contents

A Recommendations

- Return the exercise as a single text file with commented code.
- Use comments if you think it is justified.

B 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.

C 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:

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

C.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.

- 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”.

C.2 Working directory

All of your code should run. Create a directory called `R-ex-1`; place in this directory the data files and the script with code. I will run your scripts from this directory. So, when reading data, you do NOT have to specify paths. For instance, do not do

```r
read.something("~/R-ex-1/somedata")
```

but do

```r
read.something("somedata")
```

C.3 Grading

The grading is defined in the “Rúbrica”.

1 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.

2 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] 0.01816454
```

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

```
## [1] -1.04533
```

3 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 -1.45769 -0.75161  0.45695 3.13533  2.76569  2.64342
## V3 -1.39420 -1.26278 -0.09654 0.21415 -1.27045  1.01416
## V4 -1.42779 -0.09052  0.90325 2.08754  1.60433  1.70477
## V5 -1.40715 -0.99596 -0.07194 2.23467  1.53182  1.63845
## V6 -1.42668 -1.24245  0.03232 0.93811  1.63728 -0.36075
```

3.1 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 -1.36149 -0.21672 -0.11015
## 2  AML -1.22961  1.57629  1.32525
```

3.2 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 -1.368320  0.077270 -0.00771
## 2  AML Female -1.320115  1.718425  1.13218
## 3  ALL   Male -1.321430 -0.460435 -0.16511
## 4  AML   Male -1.089020  1.457100  1.55580
```

3.3 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
## 1  ALL Female -1.368320 -1.032090 0.233810 0.641800  0.22853
## 2  AML Female -1.320115 -0.922255 0.147070 0.360200 -0.57654
## 3  ALL   Male -1.321430 -1.034565 0.266325 2.206875  1.71448
## 4  AML   Male -1.089020 -1.089020 0.046090 1.302970  1.01596
##          G6       G7       G8
## 1  0.968700 3.287650 3.039720
## 2 -0.164105 3.126385 2.901165
## 3  1.614675 3.087135 2.984870
## 4  0.512660 3.279340 2.961040
```

and the dimensions of "all.median”

```
dim(all.median)

## [1]    4 3053
```

3.4 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
## 1  ALL Female -1.3180008  0.1566979 -0.1633385  0.5815847
## 2  AML Female -1.1569633  0.3503542  1.6656317  0.3161407
## 3  ALL   Male -1.2283407  0.4253546 -0.4168571  0.8363303
## 4  AML   Male -0.3259880  1.1929901  1.4919120  0.4385808
##    G2600.mean    G2600.sd
## 1 -0.23134692  0.63891862
## 2  1.30763500  0.50586581
## 3 -0.09672786  0.78109489
## 4  1.42469200  0.40976226
```

3.5 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.   :-1.608   Min.   :-1.36832   Min.   :-1.40095
##  1st Qu.:-1.401   1st Qu.:-0.40103   1st Qu.:-0.66446
##  Median :-1.368   Median : 0.07727   Median :-0.00771
##  Mean   :-1.318   Mean   :-0.16334   Mean   :-0.23135
##  3rd Qu.:-1.216   3rd Qu.: 0.25025   3rd Qu.: 0.17697
##  Max.   :-1.032   Max.   : 0.62632   Max.   : 0.57343
## ---------------------------------------------------
## type: AML
## sex: Female
##        G1               G2124            G2600
##  Min.   :-1.4443   Min.   :1.228   Min.   :0.8156
##  1st Qu.:-1.4104   1st Qu.:1.463   1st Qu.:0.9295
##  Median :-1.3201   Median :1.718   Median :1.1322
##  Mean   :-1.1570   Mean   :1.666   Mean   :1.3076
##  3rd Qu.:-0.8941   3rd Qu.:1.810   3rd Qu.:1.7284
##  Max.   :-0.6646   Max.   :2.110   Max.   :1.9767
## ---------------------------------------------------
## type: ALL
## sex: Male
##        G1               G2124              G2600
##  Min.   :-1.5678   Min.   :-1.4765   Min.   :-1.23051
##  1st Qu.:-1.4275   1st Qu.:-1.2253   1st Qu.:-0.58571
##  Median :-1.3214   Median :-0.4604   Median :-0.16511
##  Mean   :-1.2283   Mean   :-0.4169   Mean   :-0.09673
##  3rd Qu.:-1.2341   3rd Qu.: 0.3919   3rd Qu.: 0.39772
##  Max.   : 0.1763   Max.   : 0.5792   Max.   : 1.49820
## ---------------------------------------------------
## type: AML
## sex: Male
##        G1              G2124            G2600
##  Min.   :-1.262   Min.   :1.054   Min.   :0.8136
##  1st Qu.:-1.230   1st Qu.:1.189   1st Qu.:1.2176
##  Median :-1.089   Median :1.457   Median :1.5558
##  Mean   :-0.326   Mean   :1.492   Mean   :1.4247
##  3rd Qu.: 0.849   3rd Qu.:1.576   3rd Qu.:1.7369
##  Max.   : 1.101   Max.   :2.183   Max.   :1.7997
```

(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 -1.607670  -1.400950 -1.368320 -1.318001  -1.215830
## 2  AML Female -1.444340  -1.410423 -1.320115 -1.156963  -0.894055
## 3  ALL   Male -1.567830  -1.427512 -1.321430 -1.228341  -1.234088
## 4  AML   Male -1.261830  -1.229610 -1.089020 -0.325988   0.849050
##     G1.Max. G2124.Min. G2124.1st Qu. G2124.Median G2124.Mean
## 1 -1.032090 -1.3683200    -0.4010300    0.0772700 -0.1633385
## 2 -0.664650  1.2281400     1.4631700    1.7184250  1.6656317
## 3  0.176280 -1.4764900    -1.2252900   -0.4604350 -0.4168571
## 4  1.101470  1.0537100     1.1894700    1.4571000  1.4919120
##   G2124.3rd Qu. G2124.Max.  G2600.Min. G2600.1st Qu. G2600.Median
## 1     0.2502500  0.6263200 -1.40095000   -0.66446000  -0.00771000
## 2     1.8098725  2.1103600  0.81563000    0.92949500   1.13218000
## 3     0.3919050  0.5792000 -1.23051000   -0.58571500  -0.16511000
## 4     1.5762900  2.1829900  0.81357000    1.21756000   1.55580000
##    G2600.Mean G2600.3rd Qu.  G2600.Max.
## 1 -0.23134692    0.17697000  0.57343000
## 2  1.30763500    1.72844250  1.97669000
## 3 -0.09672786    0.39772000  1.49820000
## 4  1.42469200    1.73686000  1.79967000
```

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)
```