Statistical Bioinformatics as simple as possible

Guillermo Ayala

5/11/23

Table of contents

Introduction

Abstract

This vignette is concerned with the use of `tami`. We propose different ways to do the same things. All functionalities are used. This is our purpose.

Differential expression analysis for microarrays

Packages

The following code download and install the package.

```
url_tami = "http://www.uv.es/ayala/docencia/tami/tami_1.0.tar.gz"
install.packages(url_tami,repos=  NULL,type="source")
```

We load the package.

```
pacman::p_load(tami)
```

We will use data in the package **tamidata**. This package can be installed with

```
url_tamidata = "http://www.uv.es/ayala/docencia/tami/tamidata_0.6.tar.gz"
install.packages(url_tamidata,repos=  NULL,type="source")
```

We need some additional packages.

```
pacman::p_load("Biobase","hgu133a.db","ggplot2","ggfortify")
```

How to create an `ExpressionData` starting from scratch

An `ExpressionData`is an S4 class with three slots. The first one is matrix `exprm` and we use the attribute `rownames` of the matrix (`rownames(exprm)`) as annotation. The second slot is the phenotypic variable, covariable or experimental factor with two levels indicating the classification of each column (sample) in one experimental group.

Let us make an `ExpressionData` starting from scratch. The expression matrix is just a matrix. The expression values (no real interest) will be random values generated according a standard normal distribution (null mean and standard deviation one)

```
exprm0 = matrix(rnorm(100),ncol=5)
```

Now `exprm0` has no attribute `rownames`.

```
rownames(exprm0)
```

```
NULL
```

We are going to define as `rownames`

```
rownames(exprm0) = letters[1:20]
```

This is the (small) expression matrix.

```
exprm0
```

```
         [,1]        [,2]        [,3]        [,4]       [,5]
a -0.13395778 -0.52694857 -1.62055255  0.76921858  0.1778992
b -0.19847157  1.40729154 -0.78984780 -1.11245003  0.2684750
c  0.02942477  0.06365622  0.68599039 -0.86263981  0.7119576
d  0.70182517  0.17876688  0.95758190  0.27424972 -0.6066414
e -0.01556446 -1.19474417 -0.08523030  0.73606553  0.0494633
f  0.47088521  0.89509217 -0.32369767 -1.89714583  0.9158007
g -1.07647246 -1.39430697  0.86033954  0.21380849  0.5444653
h  0.24547289  0.13391582 -0.38755958 -0.67868435  1.1338404
i  0.95261822 -1.78834569 -0.52905443 -0.02855158  1.6575092
j -0.12453040 -1.28912722 -0.23469254 -1.62505039 -0.8494635
k  1.48368478 -1.41425128 -0.67015673 -0.35315417  1.2475888
l  0.43712072  0.38737223  0.72754472  0.96984977 -1.5179842
m -1.67478938  0.66272055  0.09224186  1.06563964 -0.2136650
n -0.31457780  0.27625934 -0.15737684  1.70668545 -1.0151194
o -0.10427078  0.60232294 -0.07849946  0.23142902 -0.4011241
p  1.17104783 -1.30075687  1.50518174  1.19072294  1.3003441
q -0.42979387 -1.09950255 -0.29279656 -0.49743631 -0.7497046
r -1.40603336  0.16666155  0.93965379  0.54379823 -0.1409341
s  0.40313081 -0.14196465  0.98892105 -0.12999754  1.4466878
t  0.41833103 -2.01837845  0.75990847  0.31668069 -0.5252003
```

The second slot is the covariable `groups`, the experimental factor with two levels. The first level will be labelled "A" and the second levels as "B". The samples corresponding to group "A" occupies the columns 1, 4 and 5. The other columns (samples) corresponding to group "B".

```
groups0 = factor(c(1,2,2,1,1),levels = 1:2,labels=c("A","B"))
```

Note that the variable or factor `groups0` takes numerical values 1 and 2 but R knows that it represents categories or groups. The labels of the categories will be used in plots and summaries.

Now we can construct the `ExpressionData`. We have two possibilities. The simplest and preferred is (we use the `constructor`).

```
x  = new("ExpressionData",exprm = exprm0,groups=groups0)
```

A simple summary can be obtained with

```
x
```

```
An object of class ExpressionData
The primary identifiers are
 a b c d e f
The expression matrix is
 -0.1339578 -0.1984716 -0.5269486 1.407292 -1.620553 -0.7898478 0.7692186 -1.11245 0.1778992
The experimental factor is
 1 2 2 1 1
```

The other equivalent procedure to make the `ExpressionData` is

```
x = new("ExpressionData",exprm = exprm0, groups = groups0)
```

From ExpressionSet to ExpressionData

Usually, we will start our analysis with a data set organized using a `Biobase::ExpressionSet`. For instance, `tamidata::gse21942`. It is easy to construct an `ExpressionData`. First we load `tamidata::gse21942`.

```
data(gse21942,package="tamidata")
x = ExpressionData(exprm = exprs(gse21942),
                   groups = pData(gse21942)[,"FactorValue..DISEASE.STATE."],
                   type="microarray")
```

Using `ExpressionData`

We can recover the different slots using the corresponding `accessors`.

```
head(exprm(x)) ## to limit the output
```

```
          GSM545846.CEL GSM545845.CEL GSM545844.CEL GSM545843.CEL GSM545842.CEL
1007_s_at      6.701185      6.770585      6.896115      6.913170      7.055230
1053_at        6.904241      7.160595      6.986157      7.252437      6.820581
```

```
117_at         8.347887      8.309832      8.042841      7.973542      7.905076
121_at         7.527261      7.732676      7.534203      7.582493      7.624345
1255_g_at      2.741237      2.785420      2.705875      2.712574      2.874974
1294_at        8.512001      8.697264      8.471296      8.609792      8.932694
           GSM545841.CEL GSM545840.CEL GSM545839.CEL GSM545838.CEL GSM545837.CEL
1007_s_at      7.090447      7.161786      6.990521      7.143398      7.019538
1053_at        6.983571      6.801948      7.131738      6.941465      6.829286
117_at         7.656960      7.661755      8.849415      7.781422      7.431034
121_at         7.462491      7.757673      7.771086      7.728931      7.507087
1255_g_at      2.731671      3.032517      2.957291      2.908258      2.801843
1294_at        8.466908      8.613721      8.996203      8.312664      8.444103
           GSM545836.CEL GSM545835.CEL GSM545834.CEL GSM545833.CEL GSM545832.CEL
1007_s_at      7.263345      7.183820      7.068422      7.018434      7.134651
1053_at        7.227540      6.779245      7.141372      6.908670      7.484561
117_at         8.128334      7.638504      8.336013      8.466568      8.495304
121_at         7.604226      7.721856      7.660064      7.735517      7.514174
1255_g_at      2.949537      2.747506      2.634829      2.995889      2.631472
1294_at        9.093238      9.048817      9.141685      8.805013      8.557823
           GSM545831.CEL GSM545830.CEL GSM545829.CEL GSM545828.CEL GSM545827.CEL
1007_s_at      7.164015      7.079536      6.879690      6.577880      6.928483
1053_at        6.997132      6.890387      7.179972      7.303770      7.410144
117_at         8.036675      8.228937      7.237569      8.042661      8.141817
121_at         7.553159      7.623796      7.507285      7.698743      7.696693
1255_g_at      2.708215      2.840438      2.723045      2.651039      2.906055
1294_at        8.761252      8.737342      8.528587      8.587533      8.761697
           GSM545826.CEL GSM545825.CEL GSM545824.CEL GSM545823.CEL GSM545822.CEL
1007_s_at      6.653279      6.900510      6.436211      7.008971      6.834951
1053_at        7.120752      7.109658      7.085381      7.046270      7.324719
117_at         8.033017      7.955324      8.457568      7.863781      7.965000
121_at         7.522826      7.725928      7.733425      7.846924      7.534118
1255_g_at      2.803210      2.889357      2.781139      2.868890      2.632601
1294_at        8.669705      8.470591      8.596620      8.660778      8.730843
           GSM545821.CEL GSM545820.CEL GSM545819.CEL GSM545818.CEL
1007_s_at      6.747209      7.052194      6.785745      6.715101
1053_at        7.176329      7.224084      7.369654      7.277721
117_at         7.776154      7.825469      8.342164      7.998972
121_at         7.443340      7.714347      7.434026      7.699178
1255_g_at      2.947213      2.765574      2.898040      2.709309
1294_at        8.648626      8.640482      8.730575      8.606553
```

```
groups(x)
```

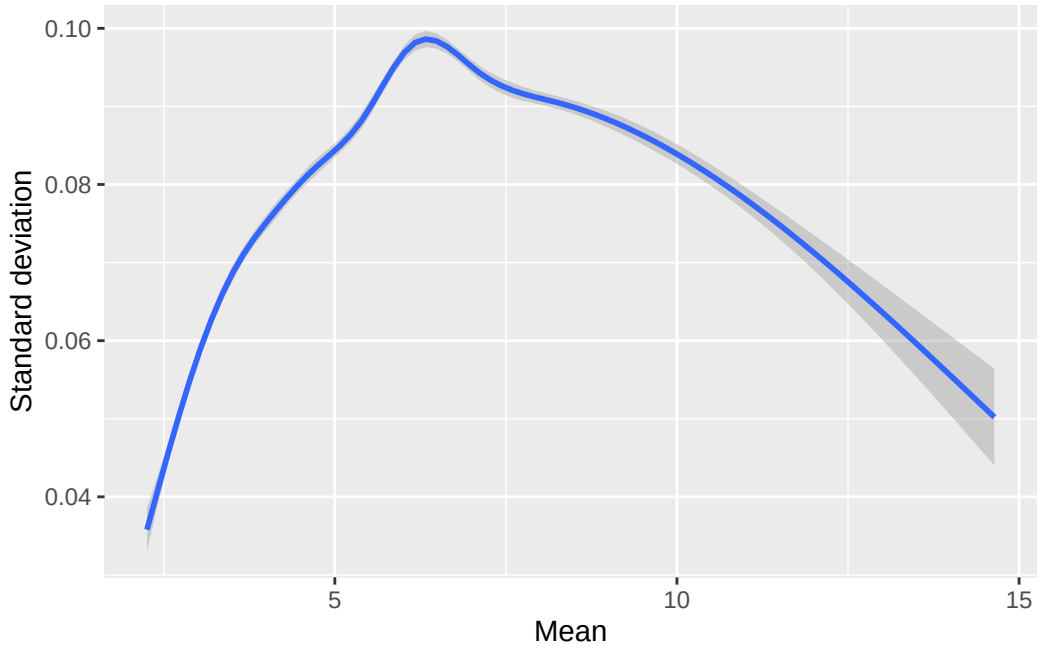
```
 [1] multiple sclerosis multiple sclerosis multiple sclerosis multiple sclerosis
 [5] multiple sclerosis multiple sclerosis multiple sclerosis multiple sclerosis
 [9] multiple sclerosis multiple sclerosis multiple sclerosis multiple sclerosis
[13] multiple sclerosis multiple sclerosis healthy            healthy
[17] healthy            healthy            healthy            healthy
[21] healthy            healthy            healthy            healthy
[25] healthy            healthy            healthy            healthy
[29] healthy
Levels: healthy multiple sclerosis
```

Plots for `ExpressionData`

The first plot shows the mean versus the standard deviation of the gene expression profile.

```
plot(x,1)
```

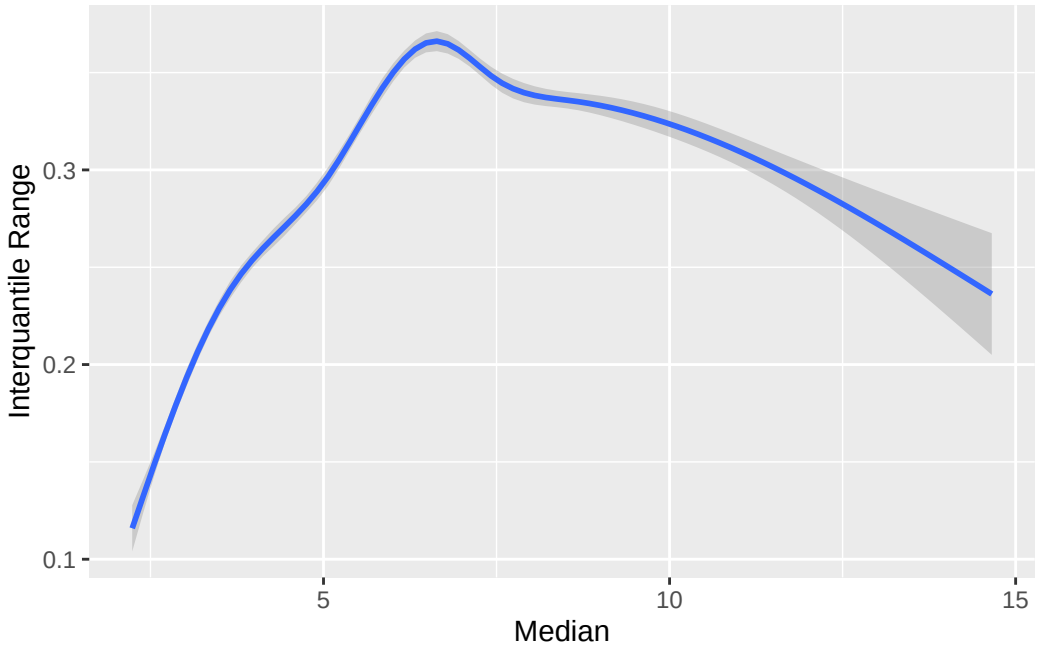
```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



The second plot is the median versus the interquantile range of the expression profile.

```
plot(x,2)
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



The third plot displays the two first principal components. The package `ggfortify` has to be loaded.

```
library(ggfortify)
plot(x,3)
```

The fourth plot displays the **Tukey mean-difference** plot to compare the mean expressión of the gene per condition. We consider the mean expression per condition for each gene: (x_i, y_i) where i is the gene and x_i and y_i and the corresponding means per condition. The `x` axis is the mean $\frac{x_i+y_i}{2}$ and the `y` axis is the difference $x_i - y_i$. The left plot corresponds to the original scale and the right plot uses the natural logarithms of the original data. Perhaps, it is more usual in the omics data literature to use the logarithm with base 2 but it is more natural our choice.

```
plot(x,4)
```

Differential Expression Input

The S4 class `DifferentialExpressionInput` provides two additional slots to control the statistical analysis to be performed.

We are going to apply a t-test per gene with the p-values adjusted using the Benjamini-Hochberg correction. Also the q-values from the original (raw) p-values are calculated.

The first option is to apply the method `rowtt` to an **ExpressionData**.

We have to define an `DifferentialExpressionInput` object. There are two options.

```
x_dei = DifferentialExpressionInput(exprm =exprm(x),
                groups =  groups(x),association ="pvalue",
                correction = "BH")
```

```
x_dei = new("DifferentialExpressionInput",exprm =exprm(x),
                groups =  groups(x),association ="pvalue",
                correction = "BH")
```

Again a short review of the object is returned typing its name.

```
x_dei
```

```
An object of class DifferentialExpressionInput
The primary identifiers are
 1007_s_at 1053_at 117_at 121_at 1255_g_at 1294_at
The expression matrix is
 6.701185 6.770585 6.896115 6.91317 7.05523 7.090447 7.161786 6.990521 7.143398 7.019538 7.2
The experimental factor is
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
The method to correct for multiple comparisons is
 BH
The test of the statistical method is
```

```
x_t = rowt(x= exprm(x),y=groups(x))
```

```
x_t2 = rowtest(x= exprm(x),y=groups(x),test = rowt,correction = "BH")
```

The value is a `data.frame`.

```
head(x_t2)
```

```
           statistic        rawp       adjp       qval
1007_s_at -2.29869931 0.029492008 0.11049371 0.05708447
```

```
1053_at    3.44084102 0.001901206 0.01513959 0.00782158
117_at    -0.08505071 0.932848609 0.96675733 0.49945673
121_at    -0.53362792 0.597965094 0.76778314 0.39666051
1255_g_at -1.01536731 0.318943716 0.53372043 0.27573648
1294_at   -1.05030996 0.302886126 0.51746083 0.26733627
```

Note that the rows are identified using the original Affymetrix identifiers. They are the row names of the data frame.

```
head(rownames(x_t2))
```

```
[1] "1007_s_at" "1053_at"   "117_at"    "121_at"    "1255_g_at" "1294_at"
```

They will be used as **primary** gene identifiers. Instead of the t-tests where the standard error of the difference of means is estimated using the profile expression of each gene, other possible procedure is contained in the Limma R package where an hierarchical model is used. Again the same results can be obtained using two procedures implemented in tami.

```
x_tm = rowtmod(x= exprm(x),y=groups(x))
head(x_tm)
```

```
  statistic        rawp
1 -2.31854450 0.027119463
2  3.40762889 0.001819455
3 -0.08980406 0.929015315
4 -0.49077813 0.627007830
5 -0.93447609 0.357214419
6 -1.06854428 0.293449716
```

The same analysis and the adjusted p-values and q-values can be obtained using tami::rowtest.

```
x_tm2 = rowtest(x= exprm(x),y=groups(x),test = rowtmod,correction = "BH")
```

The first rows of the resulting data frame can be inspected with

```
head(x_tm2)
```

```
            statistic        rawp      adjp        qval
1007_s_at -2.31854450 0.027119463 0.1023791 0.053520473
1053_at    3.40762889 0.001819455 0.0138995 0.007266206
117_at    -0.08980406 0.929015315 0.9653334 0.504644933
121_at    -0.49077813 0.627007830 0.7885372 0.412221609
1255_g_at -0.93447609 0.357214419 0.5729244 0.299506288
1294_at   -1.06854428 0.293449716 0.5089248 0.266049350
```

Similarly we can applied the `rowt` option.

```
x_tm2 = rowtest(x= exprm(x),y=groups(x),test = rowt,correction = "BH")
```

The first rows of the resulting data frame can be inspected with

```
head(x_tm2)
```

```
            statistic        rawp       adjp       qval
1007_s_at -2.29869931 0.029492008 0.11049371 0.05708447
1053_at    3.44084102 0.001901206 0.01513959 0.00782158
117_at    -0.08505071 0.932848609 0.96675733 0.49945673
121_at    -0.53362792 0.597965094 0.76778314 0.39666051
1255_g_at -1.01536731 0.318943716 0.53372043 0.27573648
1294_at   -1.05030996 0.302886126 0.51746083 0.26733627
```

Differential Expression Output

We have now a data frame with the annotation data and a data frame with the statistical results of the differential expresion analysis. We create a **DifferentialExpressionOutput**. We can made it with two procedures.

```
x_t2_deo = DifferentialExpressionOutput(GeneData=fData(gse21942),
                                        GeneStat = x_t2,
                                        foutput ="gse21942_rowtt",fdr = .2)
```

We can access or modify the different slots using

```
GeneData(x_t2_deo)
GeneStat(x_t2_deo)
foutput(x_t2_deo)
fdr(x_t2_deo)
```

```
x_t2_deo = new("DifferentialExpressionOutput",
                          GeneData = fData(gse21942), GeneStat = x_t2,
                          foutput = "gse21942_rowtt",fdr = .2)
```

We generate the corresponding tidy report where only genes with an adjusted p-value lower than `FDR` are included in the report.

```
tami::tidy(x_t2_deo)
```

We can see the report at this file.

However, the report with all genes can be obtained with

```
tami::augment(x_t2_deo)
```

The corresponding report can be found here.

Using rowttmod

We can repeat the previous analysis by replacing tami::rowt with the moderated t-tests (from limma package). Note that we have to modify the statistical results contained in the data frame `gse21942_rowt` by the data frame `gse21942_rowtmod`.

```
x_tm2_deo = new("DifferentialExpressionOutput",
                          GeneData = fData(gse21942), GeneStat = x_tm2,
                          foutput = "gse21942_rowttmod",fdr = .2)
```

Generate the report.

```
tami::tidy(x_tm2_deo)
```

The results are included in this new file

```
dema(x=gse21942,y="FactorValue..DISEASE.STATE.",correction = "BH",test = rowt,
     foutput = "gse21942",fdr = .01)
```

Gene set analysis

We have to choose a gene set collection. The package `EnrichmentBrowser` have some useful functions to construct these gene set collections. Note that we are going to analyze human and yeast data sets.

KEGG

First, we can use the gene sets from KEGG. We download the collections and save them. The species names can be found at .

```
pacman::p_load(EnrichmentBrowser)
```

```
hsa_kegg = get.kegg.genesets("hsa")
```

```
sce_kegg = get.kegg.genesets("sce")
```

Gene Ontology

```
hsa_go = get.go.genesets(org="hsa", onto="BP", mode="GO.db")
```

```
sce_go = get.go.genesets(org="sce", onto="BP", mode="GO.db")
```

As we are going to see the genes are coded using the `ENTREZID`. ## Exploring a gene set collection

What kind of data we have?

```
class(hsa_go)
```

```
[1] "list"
```

We can use generic functions for `list`.

```
length(hsa_go)
```

```
[1] 22934
```

We can access to each element in the `list` using the position

```
hsa_go[1]
```

```
$`GO:0000002`
 [1] "5428"   "6742"   "11232"  "55186"  "56652"  "84275"  "92667"  "201973"
 [9] "1763"   "7157"   "9093"   "10891"  "80119"  "83667"  "201163" "142"
[17] "1890"   "2021"   "3980"   "4358"   "4976"   "6240"   "7156"   "10000"
[25] "50484"  "64863"  "219736" "4205"   "9361"   "291"
```

or the name

```
hsa_go["GO:0000002_mitochondrial_genome_maintenance"]
```

```
$<NA>
NULL
```

If we want the elements then

```
hsa_go[[1]]
```

```
 [1] "5428"   "6742"   "11232"  "55186"  "56652"  "84275"  "92667"  "201973"
 [9] "1763"   "7157"   "9093"   "10891"  "80119"  "83667"  "201163" "142"
[17] "1890"   "2021"   "3980"   "4358"   "4976"   "6240"   "7156"   "10000"
[25] "50484"  "64863"  "219736" "4205"   "9361"   "291"
```

or

```
hsa_go[["GO:0000002_mitochondrial_genome_maintenance"]]
```

```
NULL
```

The first elements of the `list`.

```
head(hsa_go)
```

```
$`GO:0000002`
 [1] "5428"   "6742"   "11232"  "55186"  "56652"  "84275"  "92667"  "201973"
 [9] "1763"   "7157"   "9093"   "10891"  "80119"  "83667"  "201163" "142"
[17] "1890"   "2021"   "3980"   "4358"   "4976"   "6240"   "7156"   "10000"
[25] "50484"  "64863"  "219736" "4205"   "9361"   "291"
```

```
$`GO:0000003`
   [1] "49"     "167"    "190"    "268"    "301"    "367"
   [7] "638"    "699"    "701"    "993"    "994"    "995"
  [13] "1060"   "1654"   "1761"   "2072"   "2177"   "2348"
  [19] "2350"   "2352"   "2488"   "2492"   "2515"   "2528"
  [25] "2622"   "3010"   "3024"   "3248"   "3267"   "3364"
  [31] "3619"   "3973"   "4342"   "4361"   "4438"   "4439"
  [37] "5021"   "5048"   "5139"   "5617"   "5620"   "5660"
  [43] "5819"   "5888"   "5889"   "5892"   "6117"   "6406"
  [49] "6407"   "6847"   "6865"   "6869"   "6870"   "6954"
  [55] "7141"   "7142"   "7153"   "7155"   "7222"   "7225"
  [61] "7272"   "7283"   "7432"   "7783"   "7784"   "8438"
  [67] "8468"   "8521"   "8653"   "8747"   "8748"   "8749"
  [73] "9232"   "9319"   "9576"   "9700"   "9918"   "9985"
  [79] "10018"  "10111"  "10370"  "10388"  "10426"  "10655"
  [85] "10744"  "10844"  "10942"  "11022"  "11055"  "11057"
  [91] "11085"  "11086"  "11105"  "11144"  "22862"  "22917"
  [97] "23291"  "23310"  "23424"  "23626"  "25788"  "25858"
 [103] "26108"  "26255"  "26271"  "26528"  "26998"  "27175"
 [109] "27229"  "27443"  "29781"  "29893"  "43847"  "50511"
 [115] "51298"  "51314"  "54514"  "54558"  "54586"  "54742"
 [121] "54763"  "54970"  "55521"  "56154"  "56155"  "56158"
 [127] "56159"  "56165"  "56907"  "56979"  "57113"  "57151"
 [133] "57587"  "57820"  "57829"  "58524"  "58531"  "63948"
 [139] "63950"  "63951"  "64100"  "64753"  "79084"  "79703"
 [145] "79846"  "79925"  "80010"  "80198"  "80217"  "81626"
 [151] "81833"  "83449"  "83540"  "83639"  "83893"  "83990"
 [157] "84057"  "84071"  "84223"  "84225"  "84229"  "84464"
 [163] "84501"  "84690"  "84944"  "85376"  "85378"  "85438"
 [169] "89765"  "89869"  "90780"  "91646"  "114791" "117144"
 [175] "117155" "119710" "124626" "124783" "124817" "124912"
 [181] "126549" "128153" "128497" "130951" "131375" "132141"
 [187] "135458" "135927" "136242" "139212" "140894" "145645"
 [193] "146378" "146849" "146956" "147650" "147872" "150221"
 [199] "150365" "151246" "152015" "154313" "157695" "157855"
 [205] "158401" "163589" "164045" "168391" "170370" "171169"
 [211] "171482" "171483" "171484" "197342" "201254" "203102"
 [217] "221400" "221711" "246777" "254528" "255101" "255220"
 [223] "257044" "257062" "283129" "283417" "283847" "284067"
 [229] "284071" "284359" "284680" "285498" "285588" "286151"
 [235] "286207" "286826" "317761" "339168" "339345" "339834"
 [241] "340069" "340719" "342977" "346673" "347732" "349152"
 [247] "374768" "375337" "378708" "378807" "387885" "388649"
```

```
[253] "389320"    "389852"    "390243"    "400629"    "440804"    "494551"
[259] "644186"    "728637"    "729201"    "768239"    "100125288" "100130988"
[265] "100131137" "100507650" "100996631" "101928601" "107983988" "93426"
[271] "91"        "397"       "480"       "796"       "928"       "1235"
[277] "1392"      "1393"      "2516"      "2661"      "2693"      "3965"
[283] "4838"      "4880"      "5047"      "5864"      "6469"      "6662"
[289] "6736"      "7490"      "9126"      "10497"     "10635"     "22999"
[295] "51738"     "57647"     "64750"     "66037"     "84868"     "130399"
[301] "255061"    "284382"    "373861"    "406949"    "2"         "18"
[307] "51"        "90"        "92"        "100"       "113"       "117"
[313] "133"       "150"       "151"       "174"       "181"       "182"
[319] "183"       "207"       "249"       "269"       "285"       "330"
[325] "338"       "361"       "374"       "383"       "409"       "412"
[331] "472"       "493"       "538"       "546"       "551"       "552"
[337] "558"       "572"       "578"       "581"       "582"       "595"
[343] "596"       "598"       "599"       "604"       "639"       "646"
[349] "652"       "653"       "654"       "655"       "657"       "659"
[355] "666"       "668"       "675"       "682"       "694"       "718"
[361] "734"       "771"       "780"       "811"       "824"       "835"
[367] "836"       "867"       "875"       "881"       "898"       "952"
[373] "989"       "991"       "1027"      "1028"      "1047"      "1069"
[379] "1164"      "1268"      "1271"      "1364"      "1390"      "1399"
[385] "1459"      "1495"      "1499"      "1508"      "1539"      "1543"
[391] "1545"      "1588"      "1602"      "1617"      "1618"      "1620"
[397] "1621"      "1636"      "1642"      "1718"      "1730"      "1731"
[403] "1738"      "1739"      "1785"      "1788"      "1812"      "1816"
[409] "1822"      "1828"      "1869"      "1906"      "1909"      "1910"
[415] "1912"      "2026"      "2056"      "2057"      "2067"      "2099"
[421] "2120"      "2175"      "2176"      "2182"      "2185"      "2189"
[427] "2192"      "2241"      "2253"      "2254"      "2255"      "2288"
[433] "2296"      "2353"      "2354"      "2475"      "2560"      "2583"
[439] "2591"      "2593"      "2620"      "2623"      "2660"      "2662"
[445] "2674"      "2683"      "2695"      "2706"      "2707"      "2709"
[451] "2735"      "2741"      "2743"      "2796"      "2827"      "2908"
[457] "2956"      "2959"      "3014"      "3037"      "3066"      "3073"
[463] "3074"      "3148"      "3169"      "3171"      "3172"      "3205"
[469] "3206"      "3207"      "3209"      "3235"      "3236"      "3239"
[475] "3291"      "3295"      "3298"      "3301"      "3305"      "3306"
[481] "3309"      "3371"      "3400"      "3417"      "3480"      "3482"
[487] "3485"      "3488"      "3490"      "3516"      "3549"      "3552"
[493] "3553"      "3566"      "3623"      "3625"      "3633"      "3640"
[499] "3643"      "3645"      "3673"      "3675"      "3678"      "3688"
[505] "3690"      "3691"      "3726"      "3753"      "3791"      "3814"
```

```
[511] "3815"  "3856"  "3857"  "3880"  "3948"  "3952"
[517] "3955"  "3976"  "3985"  "4033"  "4086"  "4089"
[523] "4090"  "4142"  "4179"  "4188"  "4201"  "4216"
[529] "4221"  "4240"  "4254"  "4292"  "4313"  "4318"
[535] "4323"  "4327"  "4436"  "4487"  "4488"  "4521"
[541] "4627"  "4678"  "4683"  "4693"  "4735"  "4751"
[547] "4808"  "4809"  "4824"  "4846"  "4851"  "4861"
[553] "4882"  "4889"  "4914"  "4920"  "4926"  "4948"
[559] "4956"  "4957"  "4986"  "4987"  "4988"  "4991"
[565] "5000"  "5010"  "5016"  "5017"  "5020"  "5023"
[571] "5028"  "5050"  "5066"  "5079"  "5087"  "5111"
[577] "5154"  "5156"  "5159"  "5176"  "5224"  "5228"
[583] "5232"  "5238"  "5241"  "5266"  "5268"  "5270"
[589] "5324"  "5327"  "5347"  "5360"  "5367"  "5368"
[595] "5469"  "5515"  "5518"  "5535"  "5591"  "5608"
[601] "5618"  "5619"  "5719"  "5724"  "5727"  "5729"
[607] "5730"  "5743"  "5781"  "5806"  "5887"  "5901"
[613] "5914"  "5916"  "5925"  "5926"  "5932"  "5972"
[619] "5997"  "6045"  "6194"  "6196"  "6198"  "6305"
[625] "6382"  "6414"  "6422"  "6423"  "6461"  "6477"
[631] "6498"  "6513"  "6522"  "6532"  "6573"  "6595"
[637] "6615"  "6647"  "6654"  "6670"  "6674"  "6676"
[643] "6677"  "6690"  "6714"  "6715"  "6716"  "6751"
[649] "6752"  "6753"  "6768"  "6777"  "6781"  "6788"
[655] "6789"  "6790"  "6794"  "6795"  "6812"  "6833"
[661] "6874"  "6875"  "6895"  "6950"  "7004"  "7013"
[667] "7016"  "7026"  "7043"  "7046"  "7048"  "7054"
[673] "7056"  "7067"  "7068"  "7073"  "7079"  "7080"
[679] "7098"  "7103"  "7110"  "7130"  "7182"  "7203"
[685] "7226"  "7257"  "7258"  "7301"  "7314"  "7320"
[691] "7337"  "7345"  "7349"  "7351"  "7372"  "7421"
[697] "7425"  "7458"  "7473"  "7474"  "7476"  "7484"
[703] "7528"  "7536"  "7704"  "7707"  "7855"  "8061"
[709] "8086"  "8204"  "8322"  "8398"  "8399"  "8434"
[715] "8528"  "8531"  "8546"  "8549"  "8614"  "8626"
[721] "8633"  "8654"  "8743"  "8751"  "8820"  "8852"
[727] "8879"  "8894"  "8924"  "8932"  "9104"  "9133"
[733] "9134"  "9148"  "9156"  "9184"  "9191"  "9241"
[739] "9271"  "9289"  "9389"  "9403"  "9420"  "9444"
[745] "9468"  "9509"  "9510"  "9514"  "9519"  "9612"
[751] "9667"  "9724"  "9825"  "9897"  "10049" "10051"
[757] "10096" "10097" "10116" "10134" "10155" "10179"
[763] "10184" "10403" "10461" "10468" "10481" "10491"
```

```
 [769] "10524"    "10549"    "10560"    "10574"    "10575"    "10576"
 [775] "10592"    "10653"    "10657"    "10661"    "10694"    "10735"
 [781] "10765"    "10818"    "10881"    "10919"    "10927"    "10935"
 [787] "10959"    "10991"    "11189"    "11218"    "11251"    "11315"
 [793] "11331"    "22836"    "22933"    "22948"    "22994"    "23064"
 [799] "23157"    "23205"    "23230"    "23304"    "23345"    "23353"
 [805] "23394"    "23397"    "23399"    "23411"    "23414"    "23492"
 [811] "23542"    "23598"    "23609"    "23633"    "23641"    "23705"
 [817] "23762"    "24145"    "24149"    "25777"    "25809"    "25831"
 [823] "25911"    "25932"    "25945"    "25976"    "26003"    "26064"
 [829] "26165"    "26206"    "26471"    "27030"    "27125"    "27127"
 [835] "27136"    "27285"    "27306"    "29844"    "29924"    "29974"
 [841] "29988"    "49855"    "50487"    "50846"    "51087"    "51247"
 [847] "51343"    "51361"    "51460"    "51465"    "51542"    "51665"
 [853] "51742"    "51807"    "53340"    "54106"    "54361"    "54407"
 [859] "54457"    "54466"    "54585"    "54851"    "54852"    "54888"
 [865] "54993"    "55064"    "55120"    "55124"    "55231"    "55329"
 [871] "55342"    "55366"    "55585"    "55636"    "55706"    "55723"
 [877] "55811"    "55815"    "55818"    "55840"    "55870"    "55905"
 [883] "56163"    "56729"    "56848"    "56956"    "57054"    "57055"
 [889] "57095"    "57097"    "57135"    "57178"    "57599"    "57728"
 [895] "57731"    "57828"    "59272"    "59338"    "59343"    "59350"
 [901] "63894"    "63946"    "63978"    "64224"    "64321"    "64395"
 [907] "64396"    "64478"    "64591"    "64645"    "64847"    "64848"
 [913] "65010"    "65110"    "79582"    "79727"    "79747"    "79820"
 [919] "79893"    "79969"    "79977"    "79989"    "80000"    "80025"
 [925] "81539"    "81616"    "81623"    "81671"    "81892"    "81930"
 [931] "83700"    "83890"    "83943"    "84056"    "84132"    "84152"
 [937] "84159"    "84168"    "84172"    "84215"    "84221"    "84687"
 [943] "84688"    "84694"    "84812"    "84930"    "85315"    "85413"
 [949] "85417"    "89766"    "89887"    "91746"    "91978"    "93649"
 [955] "113746"   "114112"   "115948"   "116369"   "117154"   "120892"
 [961] "121355"   "122042"   "122258"   "122664"   "124404"   "125972"
 [967] "128637"   "130497"   "132243"   "132612"   "133558"   "135138"
 [973] "135935"   "139886"   "143471"   "143689"   "144195"   "144535"
 [979] "146310"   "146852"   "147912"   "148229"   "148327"   "149685"
 [985] "150159"   "151056"   "151195"   "151449"   "152006"   "152586"
 [991] "157506"   "157777"   "158062"   "161829"   "164091"   "164395"
 [997] "164684"   "166378"   "169981"   "170690"   "192670"   "199720"
[1003] "200232"   "200373"   "202051"   "203523"   "204474"   "219670"
[1009] "219793"   "219938"   "221656"   "221823"   "222698"   "225689"
[1015] "259266"   "261734"   "283471"   "283629"   "283677"   "284338"
[1021] "285335"   "286128"   "286234"   "317719"   "338879"   "339906"
```

```
[1027] "340784"    "344758"    "353189"    "373863"    "387712"    "388799"
[1033] "389730"    "389761"    "389762"    "389763"    "431707"    "440699"
[1039] "440822"    "441452"    "449520"    "474343"    "619556"    "642623"
[1045] "642636"    "644150"    "644890"    "645832"    "645961"    "646480"
[1051] "647060"    "727830"    "727905"    "728132"    "728137"    "728395"
[1057] "728403"    "729967"    "100130958" "100289087" "102724560" "1396"
[1063] "1594"      "2626"      "2627"      "3714"      "5310"      "6092"
[1069] "6299"      "6586"      "7022"      "7482"      "8433"      "8510"
[1075] "8644"      "8909"      "9353"      "10046"     "10361"     "10732"
[1081] "22803"     "22873"     "26292"     "29127"     "54456"     "56977"
[1087] "84073"     "116832"    "132625"    "138474"    "140732"    "140801"
[1093] "3624"      "4762"      "5810"      "6627"      "140947"    "497189"
[1099] "226"       "708"       "1672"      "1767"      "2013"      "2295"
[1105] "2529"      "2697"      "3479"      "6658"      "6691"      "6926"
[1111] "7076"      "7417"      "7422"      "8701"      "8890"      "8892"
[1117] "8893"      "8912"      "9083"      "9898"      "10265"     "10699"
[1123] "10734"     "10916"     "23639"     "25790"     "25836"     "25981"
[1129] "27019"     "27161"     "51673"     "51759"     "55036"     "55743"
[1135] "55779"     "57119"     "57122"     "57697"     "60675"     "64220"
[1141] "79816"     "80314"     "84074"     "84660"     "84733"     "85360"
[1147] "89876"     "122402"    "131118"    "144132"    "144406"    "146845"
[1153] "148281"    "150921"    "151648"    "154197"    "159686"    "199223"
[1159] "219990"    "253943"    "286464"    "338323"    "339829"    "346288"
[1165] "347688"    "377630"    "378948"    "401024"    "402573"    "406950"
[1171] "442867"    "442868"    "100506013" "109"       "142"       "351"
[1177] "406"       "583"       "585"       "658"       "676"       "1051"
[1183] "1080"      "1435"      "1525"      "1811"      "1843"      "1958"
[1189] "2054"      "2069"      "2263"      "2625"      "2649"      "2678"
[1195] "2712"      "2801"      "2879"      "3622"      "3953"      "3975"
[1201] "4036"      "4184"      "4192"      "4603"      "4753"      "4867"
[1207] "4878"      "5049"      "5104"      "5125"      "5127"      "5350"
[1213] "5414"      "5566"      "5764"      "5798"      "5872"      "5950"
[1219] "5990"      "6098"      "6652"      "6665"      "6774"      "6901"
[1225] "6943"      "7042"      "7584"      "7589"      "7917"      "8085"
[1231] "8195"      "8243"      "8320"      "8372"      "8382"      "9025"
[1237] "9421"      "9425"      "9575"      "9633"      "9665"      "9698"
[1243] "9702"      "9940"      "10038"     "10409"     "10420"     "10519"
[1249] "10733"     "11020"     "11063"     "11077"     "22887"     "23139"
[1255] "23198"     "23213"     "23236"     "23315"     "23318"     "23617"
[1261] "23627"     "26038"     "26091"     "26140"     "26330"     "27120"
[1267] "28981"     "29118"     "29947"     "30812"     "51441"     "51547"
[1273] "51574"     "51668"     "51804"     "54760"     "54890"     "54937"
[1279] "55063"     "55726"     "55810"     "56262"     "56339"     "56603"
```

```
[1285] "56776"     "57721"     "58494"     "63979"     "64147"     "64207"
[1291] "64518"     "78995"     "79173"     "79645"     "79670"     "79733"
[1297] "81492"     "81629"     "83447"     "83853"     "83942"     "83983"
[1303] "84072"     "84236"     "84515"     "84519"     "84678"     "84691"
[1309] "90410"     "90853"     "91603"     "94107"     "126206"    "127579"
[1315] "133308"    "136332"    "136991"    "143678"    "144455"    "147700"
[1321] "149095"    "150280"    "151254"    "153218"    "157680"    "160762"
[1327] "161142"    "161514"    "161931"    "162540"    "162979"    "164714"
[1333] "170506"    "200162"    "200558"    "201164"    "202500"    "203074"
[1339] "221481"    "245711"    "254394"    "255626"    "256006"    "256710"
[1345] "326340"    "338773"    "341277"    "341567"    "346653"    "375189"
[1351] "375341"    "388336"    "388553"    "391714"    "399949"    "402381"
[1357] "431705"    "441161"    "642658"    "643376"    "646799"    "730249"
[1363] "101927581" "56"        "247"       "259"       "283"       "996"
[1369] "1081"      "1394"      "1538"      "2302"      "2692"      "4117"
[1375] "5069"      "5670"      "5675"      "5885"      "6013"      "7356"
[1381] "7455"      "8605"      "8697"      "8881"      "9130"      "10393"
[1387] "10658"     "10983"     "23742"     "23780"     "25847"     "25906"
[1393] "26256"     "29882"     "29945"     "51433"     "51434"     "51529"
[1399] "55339"     "56648"     "56853"     "57082"     "57446"     "64682"
[1405] "79400"     "80237"     "84203"     "84750"     "93185"     "113451"
[1411] "116138"    "117285"    "119504"    "219771"    "246184"    "256126"
[1417] "344018"    "406991"    "441531"    "728695"    "100137049" "116"
[1423] "841"       "983"       "1017"      "1082"      "1307"      "1586"
[1429] "1833"      "2491"      "2834"      "3293"      "3593"      "3620"
[1435] "3972"      "4012"      "4486"      "4543"      "5073"      "5467"
[1441] "5568"      "5571"      "5669"      "5671"      "5672"      "5673"
[1447] "5676"      "5678"      "5680"      "5737"      "5744"      "5858"
[1453] "5890"      "5987"      "6019"      "6046"      "6159"      "6696"
[1459] "6732"      "6863"      "6866"      "7005"      "7156"      "7216"
[1465] "7516"      "7812"      "7932"      "7993"      "8031"      "8239"
[1471] "8287"      "8607"      "8900"      "8999"      "9082"      "9085"
[1477] "9125"      "9183"      "9210"      "9426"      "9463"      "10007"
[1483] "10017"     "10149"     "10343"     "10406"     "10407"     "10522"
[1489] "10566"     "10609"     "10766"     "10863"     "10876"     "23764"
[1495] "26476"     "26609"     "26664"     "30014"     "51207"     "51686"
[1501] "53405"     "80705"     "94027"     "203611"    "253175"    "728712"

$`GO:0000009`
[1] "79087" "55650"

$`GO:0000010`
[1] "23590" "57107"
```

```
$`GO:0000012`
 [1] "54840"     "55775"     "1161"      "2074"      "3981"      "7141"
 [7] "7515"      "100133315" "142"       "23411"     "200558"    "7014"

$`GO:0000014`
 [1] "2021"  "2072"  "4361"  "6419"  "9941"  "2067"  "10111" "64421" "7515"
[10] "5932"
```

Using `lapply` and `sapply`

If we want to know the number of genes for each gene set.

```
ngs = lapply(hsa_go,length)
```

What is `ngs`?

```
class(ngs)
```

```
[1] "list"
```

Probably a better choice to calculate the lengths would be

```
ngs = sapply(hsa_go,length)
```

Now `ngs` is

```
class(ngs)
```

```
[1] "integer"
```

A summary of the lengths is

```
summary(ngs)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     2.0     6.0    90.3    24.0 19518.0
```

Perhaps a kernel density estimator could show the gene set length distribution.

```
df = data.frame(ngs)
ggplot(data=df,aes(x=ngs)) + geom_density()
```