Lecture 9 - Bioconductor

Bioconductor is a collection of libraries to help analyze genomic data.
Currently, it is a fairly eclectic collection. It seems most complete in tools to help analyze affymatrix arrays.

Information about bioconductor is at www.bioconductor.org.

Click on “Release 1.3 Packages” to see what packages are available.

Currently the following look interesting:

- affy - analyzing affymatrix arrays
- marray - analyzing cDNA arrays
- SNPtools - download SNP data - how complete?

Under “metadata” you’ll see packages that contain data - GO data, location data, etc.

Using metadata

In order to use metadata, we first have to load the right library:

> library(hgu95av2); library(annotate)
  Loading required package: Biobase

Welcome to Bioconductor
  Vignettes contain introductory material. To view,
  simply type: openVignette()
  For details on reading vignettes, see
  the openVignette help page.

Synching your local package management information ...

Note: reposTools can not access /usr/lib/R/site-library.
This will not affect your R session unless you wish
to install/update/remove packages from this directory

Note: reposTools can not access /usr/lib/R/library.
This will not affect your R session unless you wish
to install/update/remove packages from this directory

> hgu95av2()
Quality control information for hgu95av2
Date built: Wed Jan 14 22:11:00 2004
Number of probes: 12625
Probe number missmatch: None
Probe missmatch: None
Mappings found for probe based rda files:
  hgu95av2ACCNUM found 12625 of 12625
  hgu95av2CHRLOC found 11426 of 12625
  hgu95av2CHR found 12183 of 12625
  hgu95av2ENZYME found 1619 of 12625
  hgu95av2GENENAME found 12208 of 12625
  hgu95av2GO found 9437 of 12625
  hgu95av2GRIF found 6554 of 12625
  hgu95av2HGID found 11337 of 12625
  hgu95av2LOCUSID found 12280 of 12625
  hgu95av2MAP found 12145 of 12625
  hgu95av2NM found 11400 of 12625
  hgu95av2NP found 11400 of 12625
  hgu95av2OMIM found 9528 of 12625
  hgu95av2PATH found 2234 of 12625
  hgu95av2PMID found 12033 of 12625
  hgu95av2SUMFUNC found 656 of 12625
  hgu95av2SYMBOL found 12208 of 12625
  hgu95av2UNIGENE found 11940 of 12625
Mappings found for non-probe based rda files:
  hgu95av2ENZYME2PROBE found 565
  hgu95av2GO2ALLPROBES found 4405
  hgu95av2GO2PROBE found 3212
  hgu95av2PATH2PROBE found 122
  hgu95av2PMID2PROBE found 48146

These define environments. Environments are areas in which variables are stored:

> ls()

[1] "last.warning"

> a=1:100
> b="hello"
> ls()

[1] "a"               "b"               "last.warning"

> env1=new.env()
> ls(env=env1)

character(0)

> with(env1, a<~4 )
> a

with(env1, a)
[1] 4
ls(env=env1)
[1] "a"
get("a",env1)
[1] 4

Back to the library:

hgu95av2()

Quality control information for  hgu95av2
Date built: Wed Jan 14 22:11:00 2004
Number of probes: 12625
Probe number mismatch: None
Probe mismatch: None
Mappings found for probe based rda files:
  hgu95av2ACCCNUM found 12625 of 12625
  hgu95av2CHRLOC found 11426 of 12625
  hgu95av2CHR found 12183 of 12625
  hgu95av2ENZYME found 1619 of 12625
  hgu95av2GENENAME found 12208 of 12625
  hgu95av2GO found 9437 of 12625
  hgu95av2GRIF found 6554 of 12625
  hgu95av2HGID found 11337 of 12625
  hgu95av2LOCUSID found 12280 of 12625
  hgu95av2MAP found 12145 of 12625
  hgu95av2NM found 11400 of 12625
  hgu95av2NP found 11400 of 12625
  hgu95av2OMIM found 9528 of 12625
  hgu95av2PATH found 2234 of 12625
  hgu95av2PMID found 12033 of 12625
  hgu95av2SUMFUNC found 656 of 12625
  hgu95av2SYMBOL found 12208 of 12625
  hgu95av2UNIGENE found 11940 of 12625
Mappings found for non-probe based rda files:
  hgu95av2ENZYME2PROBE found 565
  hgu95av2GO2ALLPROBES found 4405
  hgu95av2GO2PROBE found 3212
  hgu95av2PATH2PROBE found 122
  hgu95av2PMID2PROBE found 48146

x=ls(env=hgu95av2CHRLOC)
length(x)
[1] 12625
x[1]
This is an R environment (hash table like) object containing key and value pairs for the mappings between probe identifiers (key) and transcription starting positions of genes on chromosomes (value). Keys can be accessed using `ls(name of the environment)` and values using `get(key, name of the environment)` or `multiget(keys, name of the environment)`. Values are named vectors of length 1 or more depending on whether a given probe id can be mapped to a single or multiple chromosomes. The names give the chromosome number of concern. NA is assigned to probe identifiers that cannot be mapped to any chromosomal location data at this time. Starting positions for genes on the antisense strand have a leading "-" sign (e.g. -1234567). The starting positions for both the sense and antisense strand are number of base pairs measured from the p (5' end of the sense strand) to q (3' end of the sense strand) arms. When a gene cannot be placed on a chromosome with confidence, "random" is appended to the end of the name for a chromosomal location value. Mappings were obtained using reflint.txt.gz and refGene.txt.gz file from Golden Path (<URL: http://www.genome.ucsc.edu/goldenPath/>) from the latest release.
Examples:

```r
require("annotate") || stop("annotate unavailable")
xx <- ls(env = hgu95av2CHRLOC)
if(length(xx) > 0){
  # Using get for value of the first key
  get(xx[1], hgu95av2CHRLOC )
  #Using multiget for a few keys
  if(length(xx) >= 3){
    multiget(xx[1:3], hgu95av2CHRLOC )
    #Using lookUp of annotate(> 1.3.4)
    lookUp(xx[1:3],"hgu95av2","CHRLOC")
  }
}
```

We see that the name is the chromosome, and the value is the location. The location is negative if the gene is on the antisense strand. Location is measured on the sense strand.

Let us look at the GO annotation:

```r
x=ls(env=hgu95av2GO)
length(x)
[1] 12625
x[1]
[1] "1000_at"
> x[2]
[1] "1001_at"
> get(x[1],env=hgu95av2GO)
[1] NA
> get(x[2],env=hgu95av2GO)
$"GO:0004714"
$"GO:0004714"$GOID
[1] "GO:0004714"

$"GO:0004714"$Evidence
[1] "TAS"

$"GO:0004714"$Ontology
[1] "MF"

$"GO:0005524"
$"GO:0005524"$GOID
[1] "GO:0005524"

$"GO:0005524"$Evidence
[1] "IEA"
```
a[1]

library(GO)

quality control information for GO
Date built: Tue Jan 13 13:48:19 2004
Mappings found for non-probe based rda files:
  GOBPCHILDREN found 3751
  GOBPPARENTS found 8026
  GOCCCHILDREN found 462
  GOCCPARENTS found 1363
  GOGO2LL found 5785
  GULL2GO found 36295
  GOMFCCHILDREN found 1376
  GOMFPARENTS found 7268
  GOTERM found 16660

get(a[1]$GOID, env=GOTERM)
> MF
"transmembrane receptor protein tyrosine kinase activity"

> ?hgu95av2GO2ALLPROBES

hgu95av2GO2ALLPROBES R Documentation

An annotation data file for GO2ALLPROBES in the hgu95av2 package

Description:

This is an R environment (hash table like) object containing key and value pairs for the mappings between Gene Ontology ids (key) and probe identifiers (value) associated with a given GO id and all the offspring of that GO id. Keys can be accessed using ls(name of the environment) and values using get(key, name of the environment) or multiget(keys, name of the environment). Values may be vectors of length 1 or greater depending on whether a given GO id can be mapped to only one or more probe identifiers. Names for values are the evidence codes for the GO ids (if evidence code was provided by source data). The evidence codes in use include:

IMP inferred from mutant phenotype
IGI inferred from genetic interaction
IPI inferred from physical interaction
ISS inferred from sequence similarity
IDA inferred from direct assay
IEP inferred from expression pattern
IEA inferred from electronic annotation
TAS traceable author statement
NAS non-traceable author statement
ND no biological data available
IC inferred by curator GO ids can not be mapped to any probe identifier is assigned a value of NA. Mappings between Gene Ontology ids and Gene Ontology terms and other information are available in a separate data package named GO.

Details:

Mappings were based on data provided by LocusLink

Source data built: LocusLink built: January 13, 2004.<URL:

References:


Examples:

```r
require("annotate") || stop("annotate unavailable")
xx <- ls(env = hgu95av2GO2ALLPROBES)
if(length(xx) > 0){
  # Using get for value of the first key
  get(xx[1], hgu95av2GO2ALLPROBES )
  # Using multiget for a few keys
  if(length(xx) >= 3){
    multiget(xx[1:3], hgu95av2GO2ALLPROBES )
    # Using lookUp of annotate(> 1.3.4)
    lookUp(xx[1:3],"hgu95av2","GO2ALLPROBES")
  }
}

> get(a[[1]]$GOID, env=hgu95av2GO2ALLPROBES)

  IEA   IEA   IEA   IEA   IEA
"1108_s_at" "1234_at" "1606_at" "1485_at" "34331_at"
TAS   TAS   TAS   TAS   TAS
"39930_at" "34573_at" "898_s_at" "39947_at" "469_at"
TAS   TAS   TAS   TAS   TAS
"1604_at" "1605_g_at" "34564_at" "2088_s_at" "41678_at"
TAS   NR    NR    TAS   TAS
"902_at" "1537_at" "37327_at" "1585_at" "1723_g_at"
TAS   TAS   TAS   TAS   TAS
"1742_at" "2089_s_at" "32787_at" "33638_at" "33639_g_at"
TAS   IEA   IEA   IEA   IEA
"1727_at" "1335_at" "31335_at" "34718_at" "1572_s_at"
IEA   E     E     E     E
"33162_at" "2056_at" "2057_g_at" "36168_at" "424_s_at"
E     E     TAS   TAS   TAS
"31805_at" "637_at" "1291_s_at" "1608_at" "1609_g_at"
TAS   TAS   TAS   TAS   TAS
"1812_s_at" "35684_at" "1335_at" "31335_at" "34718_at"
TAS   TAS   TAS   TAS   E
"160027_s_at" "40936_at" "160022_at" "1317_at" "1802_s_at"
E     E     P     E     E
"1901_s_at" "33218_at" "36805_s_at" "1354_at" "1355_g_at"
E     E     E     TAS   TAS
"33182_at" "36042_at" "38280_s_at" "1771_s_at" "36993_at"
TAS   TAS   TAS   TAS   TAS
"1731_at" "1968_g_at" "1987_at" "1988_at" "36157_at"
TAS   TAS   TAS   TAS   TAS
"1761_at" "1545_g_at" "1567_at" "1963_at" "1964_g_at"
TAS   TAS   TAS   TAS   TAS
"990_at" "991_g_at" "1065_at" "34583_at" "1954_at"
TAS   IEA   IEA   IEA   IEA
```
Installing a library

Installing a library is very easy in R. You download the file, and do

R CMD INSTALL filename

For the annotation libraries, or other ones, you might want to install a library locally. In that case you do the following

R CMD INSTALL -l /home/user/lib/R filename

You also want the following in the file ~/.Renviron:

?] R_LIBS="$HOME/lib/R:/usr/local/lib/R/site-library"

Analyzing Affy data

> library(affy)
> ?ReadAffy

read.affybatch package:affy R Documentation

Read CEL files into an AffyBatch

Description:

Read CEL files into an AffyBatch

Usage:

read.affybatch(..., filenames = character(0),
    phenoData = new("phenoData"),
    description = NULL,
    notes = "",
    compress = getOption("BioC")$affy$compress.cel,
    rm.mask = FALSE, rm.outliers = FALSE, rm.extra = FALSE,
    verbose = FALSE)
ReadAffy(..., filenames=character(0),
widget=getOption("BioC")$affy$use.widgets,
compress=getOption("BioC")$affy$compress.cel,
celfile.path=getwd(),
sampleNames=NULL,
phenoData=NULL,
description=NULL,
notes="",
rm.mask=FALSE, rm.outliers=FALSE, rm.extra=FALSE,
verbose=FALSE)

Arguments:

...: file names separated by comma.

filenames: file names in a character vector.

phenoData: a 'phenoData' object

description: a 'MIAME' object

notes: notes

compress: are the CEL files compressed ?

rm.mask: should the spots marked as 'MASKS' set to 'NA' ?

rm.outliers: should the spots marked as 'OUTLIERS' set to 'NA'

rm.extra: if 'TRUE', overrides what is in 'rm.mask' and 'rm.outliers'

verbose: verbosity flag

widget: a logical specifying if widgets should be used.

celfile.path: a character denoting the path 'ReadAffy' should look for cel files

sampleNames: a character vector of sample names to be used in the 'AffyBatch'

Details:

'ReadAffy' is a wrapper for 'read.affybatch' that permits the user to read in phenoData, MIAME information, and CEL files using widgets. One can also define files where to read phenoData and MIAME information.

If the function is call with no arguments 'ReadAffy()' all the CEL files in the working directory are read and put into an 'AffyBatch'. However, the arguments give the user great flexibility.
'phenoData' is read using 'link[Biobase]{read.phenoData}'. If a character is given it tries to read the file with that name to obtain the phenoData object as described in 'link[Biobase]{read.phenoData}'. If left 'NULL' but 'widget=TRUE' then widgets are used. If left 'NULL' and 'widget=FALSE' then a default object is created. It will be a data frame with 'new("phenoData",pData=data.frame(x=1:length(CELfiles)),varLabels=list(x="arbitrary number"))'

Value:

An 'AffyBatch' object.

Author(s):

Ben Bolstad bolstad@stat.berkeley.edu (read.affybatch), Laurent Gautier, and Rafael A. Irizarry (ReadAffy)

See Also:

'AffyBatch'

Examples:

> A=ReadAffy(widget=T)
> A

AffyBatch object
size of arrays=640x640 features (6405 kb)
cdf=HG_U95Av2 (12625 affyids)
number of samples=2

number of genes=12625
annotation=hgu95av2

> descript

$pData

subject species area
a_h1a "1" "hu" "a"
a_h1b "1" "hu" "b"

$varLabels

Description
subject ""
species ""
area ""

First we want to assess the chips in general:

> image(A)

Hit <Return> to see next plot:
> boxplot(A)
> deg=AffyRNAdeg(A)
> plotAffyRNAdeg(deg)

Now we want to calculate expression values
> exA=expresso(A,widget=T)

Loading required package: tkWidgets

Attaching package 'DynDoc':

The following object(s) are masked from package:base:

vignette

Loading required package: tcltk

Error in setCorrections() : Aborted by user

> exA=rma(A)

Now we have expression values. We can save them:

> write.table(exA,file="test.txt")
> exprs2excel(exA,"test.csv")
> a=read.table("test.csv",sep="",row.names=1,head=T)
> a[1:3,]

You can also convert the data to a table using the exprs command.

To analyze the expression data, the function iter is useful. It is similar to apply():

> iter( exA[1:3,, function(x) {x} ]

A more complicated version of iter allows us to subset the data:

> a=iter( exA[1:3,, "area", function(x,y) { print(x);print(y) } ]

[1] a b
Levels: a b

/mnt/alexandria/home/dirk/teaching/cel/a_h1a.CEL
8.425136
/mnt/alexandria/home/dirk/teaching/cel/a_h1b.CEL
8.055096
[1] a b
Levels: a b
/mnt/alexandria/home/dirk/teaching/cel/a_h1a.CEL
5.027576
/mnt/alexandria/home/dirk/teaching/cel/a_h1b.CEL
5.054206
[1] a b
Levels: a b
/mnt/alexandria/home/dirk/teaching/cel/a_h1a.CEL
4.219405
/mnt/alexandria/home/dirk/teaching/cel/a_h1b.CEL
4.385835

> a=iter( exA[1:3], "area", function(x,y) { a=split(y,x); sapply(a,mean) } )
> a
1000_at 1001_at 1002_f_at
a 8.425136 5.027576 4.219405
b 8.055096 5.054206 4.385835

> a=iter( exA[1:3], "species", function(x,y) { a=split(y,x); sapply(a,mean) } )
> a
1000_at 1001_at 1002_f_at
8.240116 5.040891 4.302620

> a=iter( exA[1:3], "species", function(x,y) { a=split(y,x); sapply(a,var) } )
> a
1000_at 1001_at 1002_f_at
0.0684650913 0.0003545645 0.0138496078

> x=1:10
> y=sample(4,10,rep=T)
> split(x,y)

    $"1"
[1] 1 4 6

    $"2"
[1] 2 3 10

    $"3"
[1] 5

    $"4"
[1] 7 8 9

> sapply(split(x,y),mean)
   1 2 3 4
3.666667 5.000000 5.000000 8.000000

>
To calculate p-values we use:

> callA=mas5calls(A)