

An Introduction to Bioconductor

In this laboratory we will introduce some of the basic interactions with Bioconductor.

```
> library(Biobase)
```

```
Welcome to Bioconductor
```

```
Vignettes contain introductory material. To view,  
simply type: openVignette()
```

```
For details on reading vignettes, see
```

```
the openVignette help page.Creating a new generic function for "summary" in pa
```

```
Biobase
```

```
Attaching package ‘Biobase’:
```

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

```
split summary
```

```
> library(annotate)
```

```
> library(golubEsets)
```

The package `golubEsets` contains three data sets that were obtained from the web and slightly massaged. They represent the data analysed in ? to perform class prediction using microarray data. The data were collected on Affymetrix Hu 6800 chip and which contains probes for 7129 genes.

An `exprSet` basically consists of the gene expression matrix (optionally a set of standard errors for those estimates), the related experimental metadata (who did what when and to what), and the phenotypic data. Here phenotype is interpreted quite broadly – it represents any physical characteristics of the sample.

```
> golubTrain
```

```
Expression Set (exprSet) with
```

```
7129 genes
```

```
38 samples
```

```
phenoData object with 11 variables and 38 cases
```

```
varLabels
```

```
Samples: Sample index
```

```
ALL.AML: Factor, indicating ALL or AML
```

```
BM.PB: Factor, sample from marrow or peripheral blood
```

```
T.B.cell: Factor, T cell or B cell leuk.
```

FAB: Factor, FAB classification
Date: Date sample obtained
Gender: Factor, gender of patient
pctBlasts: pct of cells that are blasts
Treatment: response to treatment
PS: Prediction strength
Source: Source of sample

```
> golubTrain[, 1:10]
```

Expression Set (exprSet) with

7129 genes

10 samples

phenodata object with 11 variables and 10 cases

varLabels

Samples: Sample index

ALL.AML: Factor, indicating ALL or AML

BM.PB: Factor, sample from marrow or peripheral blood

T.B.cell: Factor, T cell or B cell leuk.

FAB: Factor, FAB classification

Date: Date sample obtained

Gender: Factor, gender of patient

pctBlasts: pct of cells that are blasts

Treatment: response to treatment

PS: Prediction strength

Source: Source of sample

```
> golubTrain[1:100, ]
```

Expression Set (exprSet) with

100 genes

38 samples

phenodata object with 11 variables and 38 cases

varLabels

Samples: Sample index

ALL.AML: Factor, indicating ALL or AML

BM.PB: Factor, sample from marrow or peripheral blood

T.B.cell: Factor, T cell or B cell leuk.

FAB: Factor, FAB classification

Date: Date sample obtained

Gender: Factor, gender of patient

pctBlasts: pct of cells that are blasts

Treatment: response to treatment

PS: Prediction strength
Source: Source of sample

Notice that when subsetting we have arranged it so that the *rows* correspond to genes and the *columns* correspond to samples.

The phenotypic data are stored in a separate, but linked, data frame. You can obtain it and interact with it using specific methods.

```
> pD <- phenoData(golubTrain)
> pD
```

```
phenoData object with 11 variables and 38 cases
varLabels
```

```
Samples: Sample index
ALL.AML: Factor, indicating ALL or AML
BM.PB: Factor, sample from marrow or peripheral blood
T.B.cell: Factor, T cell or B cell leuk.
FAB: Factor, FAB classification
Date: Date sample obtained
Gender: Factor, gender of patient
pctBlasts: pct of cells that are blasts
Treatment: response to treatment
PS: Prediction strength
Source: Source of sample
```

```
> pd <- pData(pD)
> pd
```

	Samples	ALL.AML	BM.PB	T.B.cell	FAB	Date	Gender	pctBlasts	Treatment
1	1	ALL	BM	B-cell	<NA>	9/4/1996	M	NA	<NA>
2	2	ALL	BM	T-cell	<NA>	<NA>	M	NA	<NA>
3	3	ALL	BM	T-cell	<NA>	<NA>	M	NA	<NA>
4	4	ALL	BM	B-cell	<NA>	<NA>	<NA>	NA	<NA>
5	5	ALL	BM	B-cell	<NA>	<NA>	<NA>	NA	<NA>
6	6	ALL	BM	T-cell	<NA>	<NA>	M	NA	<NA>
7	7	ALL	BM	B-cell	<NA>	3/25/1983	F	NA	<NA>
8	8	ALL	BM	B-cell	<NA>	<NA>	F	NA	<NA>
9	9	ALL	BM	T-cell	<NA>	<NA>	M	NA	<NA>
10	10	ALL	BM	T-cell	<NA>	7/23/1987	M	NA	<NA>
11	11	ALL	BM	T-cell	<NA>	6/25/1985	M	NA	<NA>
12	12	ALL	BM	B-cell	<NA>	9/17/1985	F	NA	<NA>
13	13	ALL	BM	B-cell	<NA>	7/27/1988	F	NA	<NA>
14	14	ALL	BM	T-cell	<NA>	11/27/1987	M	NA	<NA>

15	15	ALL	BM	B-cell	<NA>	3/25/1989	F	NA	<NA>
16	16	ALL	BM	B-cell	<NA>	2/12/1990	M	NA	<NA>
17	17	ALL	BM	B-cell	<NA>	9/26/1990	M	NA	<NA>
18	18	ALL	BM	B-cell	<NA>	<NA>	F	NA	<NA>
19	19	ALL	BM	B-cell	<NA>	<NA>	<NA>	NA	<NA>
20	20	ALL	BM	B-cell	<NA>	<NA>	<NA>	NA	<NA>
21	21	ALL	BM	B-cell	<NA>	1/24/1984	M	NA	<NA>
22	22	ALL	BM	B-cell	<NA>	5/27/1988	M	NA	<NA>
23	23	ALL	BM	T-cell	<NA>	7/9/1991	M	NA	<NA>
24	24	ALL	BM	B-cell	<NA>	5/19/1981	M	NA	<NA>
25	25	ALL	BM	B-cell	<NA>	2/18/1982	M	NA	<NA>
26	26	ALL	BM	B-cell	<NA>	<NA>	F	NA	<NA>
27	27	ALL	BM	B-cell	<NA>	<NA>	F	NA	<NA>
34	34	AML	BM	<NA>	M2	<NA>	<NA>	77	Success
35	35	AML	BM	<NA>	M1	<NA>	<NA>	67	Success
36	36	AML	BM	<NA>	M5	<NA>	<NA>	76	Success
37	37	AML	BM	<NA>	M2	<NA>	<NA>	44	Success
38	38	AML	BM	<NA>	M1	<NA>	<NA>	80	Success
28	28	AML	BM	<NA>	M2	<NA>	<NA>	79	Failure
29	29	AML	BM	<NA>	M2	<NA>	<NA>	34	Failure
30	30	AML	BM	<NA>	M5	<NA>	<NA>	93	Failure
31	31	AML	BM	<NA>	M4	<NA>	<NA>	77	Failure
32	32	AML	BM	<NA>	M1	<NA>	<NA>	86	Failure
33	33	AML	BM	<NA>	M2	<NA>	<NA>	70	Failure

PS Source

1	1.00	DFCI
2	0.41	DFCI
3	0.87	DFCI
4	0.91	DFCI
5	0.89	DFCI
6	0.76	DFCI
7	0.78	DFCI
8	0.77	DFCI
9	0.89	DFCI
10	0.56	DFCI
11	0.74	DFCI
12	0.20	DFCI
13	1.00	DFCI
14	0.73	DFCI
15	0.98	DFCI
16	0.95	DFCI
17	0.49	DFCI

```

18 0.59 DFCI
19 0.80 DFCI
20 0.90 DFCI
21 0.76 DFCI
22 0.37 DFCI
23 0.77 DFCI
24 0.92 DFCI
25 0.43 DFCI
26 0.89 DFCI
27 0.82 DFCI
34 0.64 CALGB
35 0.21 CALGB
36 0.94 CALGB
37 0.95 CALGB
38 0.73 CALGB
28 0.44 CALGB
29 0.74 CALGB
30 0.80 CALGB
31 0.61 CALGB
32 0.47 CALGB
33 0.89 CALGB

```

An object of class `phenoData` is a combination of a dataframe containing the various data elements and a list that explains what each variable represents. This information is usually relegated to a help page but we felt that it was important to keep it more closely associated with the data.

The `$` operator performs the job of extracting particular variables from an object of class `phenoData`. It also can be used directly on the `exprSet`.

```
> table(pD$ALL.AML)
```

```
ALL AML
 27  11
```

```
> table(golubTest$ALL.AML)
```

```
ALL AML
 20  14
```

The S4 methods package has introduced substantial new capabilities into R. To obtain the manual pages for S4 classes you should use the following syntax `class?exprSet`. Please do that now and we will look at help page.

Annotation

Now we will look at how to map from Affymetrix identifiers to the different biological meta data available from the Bioconductor Web site. Other data can be obtained from other sources and used in a similar fashion.

Because these data sources are very large, are constantly evolving and are similar across species we have adopted the strategy of distributing data in R packages. Each mapping is contained in an *environment*. For our purposes an environment is simply a *hash* table. It provides a very fast way of looking up values for specific text keys. This implementation is not the best and we hope to develop real hash tables for R but that will take some time.

Load the data and obtain the Affymetrix identifiers.

```
> library(hu6800)
> ls(pos = 2)

 [1] "hu6800"           "hu6800ACCNUM"      "hu6800AFFYCOUNTS"
 [4] "hu6800CHR"       "hu6800CHRLLOC"    "hu6800CHRORI"
 [7] "hu6800ENZYME"    "hu6800ENZYME2AFFY" "hu6800GENENAME"
[10] "hu6800GO"        "hu6800GO2AFFY"    "hu6800GO2ALLAFFY"
[13] "hu6800GRIF"      "hu6800LOCUSID"    "hu6800MAP"
[16] "hu6800PATH"      "hu6800PATH2AFFY"  "hu6800PMID"
[19] "hu6800PMID2AFFY" "hu6800QC"         "hu6800SUMFUNC"
[22] "hu6800SYMBOL"    "hu6800UNIGENE"
```

```
> affynames <- ls(env = hu6800CHR)
> length(affynames)
```

```
[1] 7129
```

```
> hu6800()
```

Quality control information for hu6800:

```
Date built: Wed Oct 2 15
Number of probes: 7129
Probe number mismatch: None
Probe mismatch: None
Mappings found for probe based rda files:
  hu6800ACCNUM found 7092 of 7129
  hu6800CHRLLOC found 6342 of 7129
  hu6800CHRORI found 6342 of 7129
  hu6800CHR found 6854 of 7129
```

```
hu6800ENZYME found 1072 of 7129
hu6800GENENAME found 6889 of 7129
hu6800GO found 5729 of 7129
hu6800GRIF found 2921 of 7129
hu6800LOCUSID found 6898 of 7129
hu6800MAP found 6843 of 7129
hu6800PATH found 1454 of 7129
hu6800PMID found 6805 of 7129
hu6800SUMFUNC found 6087 of 7129
hu6800SYMBOL found 6889 of 7129
hu6800UNIGENE found 6801 of 7129
```

Mappings found for non-probe based rda files:

```
hu6800AFFYCOUNTS found 2145
hu6800ENZYME2AFFY found 473
hu6800GO2AFFY found 2145
hu6800GO2ALLAFFY found 2145
hu6800PATH2AFFY found 118
hu6800PMID2AFFY found 26337
```

After loading the data we see that there are a number of different data sets that have been loaded. These provide the many different mappings from Affymetrix identifiers to other biological data, such as chromosomal location which is found in `hu6800CHR`.

We see that there are 7129 identifiers (which matches the `exprSets` we have).

Finally, quality control data and counts etc. of the mappings found is available by calling the function `hu6800()` and by examining the help page, `?hu6800`.

```
> mygene <- affynames[4001]
> get(mygene, env = hu6800ACCNUM)

[1] "U18237"

> get(mygene, env = hu6800LOCUSID)

[1] 23456

> get(mygene, env = hu6800SYMBOL)

[1] "ABCB10"

> get(mygene, env = hu6800GENENAME)

[1] "ATP-binding cassette, sub-family B (MDR/TAP), member 10"

> get(mygene, env = hu6800SUMFUNC)
```

```

[1] NA

> get(mygene, env = hu6800UNIGENE)

[1] "Hs.1710"

> get(mygene, env = hu6800CHR)

[1] "1"

> get(mygene, env = hu6800CHRLoc)

      1
228099428

> get(mygene, env = hu6800CHRORI)

  1
"_"

> get(mygene, env = hu6800MAP)

[1] "1q42"

> get(mygene, env = hu6800PMID)

[1] 10922475 10748049 7766993

> get(mygene, env = hu6800GO)

[1] NA

> multiget(affynames[1:10], env = hu6800PMID)

$"A28102_at"
[1] 8391122 2574000 2561974

$"AB000114_at"
[1] NA

$"AB000115_at"
[1] NA

$"AB000220_at"
[1] 12174914 9405678 9168980

```

```
 $"AB000381_s_at"
```

```
 [1] 9169150 8934543
```

```
 $"AB000409_at"
```

```
 [1] 10859165 9155018
```

```
 $"AB000410_s_at"
```

```
 [1] 12189194 12164330 12119232 12117782 12034821 11992556 11927502 11902834
```

```
 [9] 11837743 10449904 10233168 9681819 9348312 9321410 9223306 9223305
```

```
 [17] 9207108 9197244 9190902 9187114
```

```
 $"AB000449_at"
```

```
 [1] 9344656
```

```
 $"AB000450_at"
```

```
 [1] 9344656
```

```
 $"AB000460_at"
```

```
 [1] 9734812
```

We have arbitrarily selected the probe set with Affymetrix identifier "U18237_at". And then obtained all the mappings for it. A variety of information about this gene can now easily be obtained.

In some cases we need to obtain data on several genes at once. To do that we wrote a special function called `multiget`. Notice that in some cases there are a number of PubMed abstracts associated with the different genes.

We can do some other interesting things as well.

```
> whChrom <- multiget(affynames, env = hu6800CHR)
```

```
> table(unlist(whChrom))
```

```
 1  10  11  12  13  14  15  16  17  18  19   2  20  21  22   3   4   5   6   7
676 252 415 424 104 230 179 266 442  98 421 439 150  99 171 362 272 292 417 321
 8   9   X   Y
235 253 320  26
```

```
> vv <- sapply(whChrom, length)
```

```
> table(vv)
```

```
vv
```

```
 1   2
7119 10
```

```

> whChrom[vv == 2]

$"D49410_at"
[1] "X" "Y"

$"HG2868-HT3012_s_at"
[1] "X" "Y"

$"HG3936-HT4206_at"
[1] "X" "Y"

$"J03592_at"
[1] "X" "Y"

$"L39064_rna1_at"
[1] "X" "Y"

$"M16279_at"
[1] "X" "Y"

$"U11090_at"
[1] "X" "Y"

$"U13706_at"
[1] "X" "Y"

$"U82668_rna1_at"
[1] "X" "Y"

$"X17648_at"
[1] "X" "Y"

```

So we see the distribution of probe sets according to chromosome. Also note that there are 10 genes that have two chromosomes assigned. We might want to explore these further by examining resources at the NCBI.

Querying PubMed

For more details please look at the short article in R News,

The National Library of Medicine (NLM) provides a great deal of information on biological data. We have only just started to explore these data. We need further functionality on MeSH and other data that are there; but for now we concentrate on PubMed.

A mapping has been done between LocusLink and PubMed. This associates specific articles with specific genes. These can be queried interactively using tools available in R and other systems. Full text abstracts are readily available, in some cases other full text articles are available (we just don't have the tools to make use of them). We next demonstrate how you could interact with these.

```
> pmids <- get(mygene, env = hu6800PMID)
> pubmed(pmids, disp = "browser")
> absts2 <- pm.getabst(mygene, "hu6800")
```

Loading required package: XML

```
> pm.titles(absts2)
[[1]]
[1] "M-ABC2, a new human mitochondrial ATP-binding cassette membrane protein."
[2] "Characterization of ABCB9, an ATP binding cassette protein associated with lysosomes"
[3] "Characterization and mapping of three new mammalian ATP-binding transporter genes"
> sapply(absts2, function(x) pm.abstGrep("[Pp]rotein", x))
      U18237_at
[1,]      TRUE
[2,]      TRUE
[3,]     FALSE
```

We obtain the abstracts and then search for the word `protein`.

Another source of data is GenBank. We can explore interactions with it in much the same way as with PubMed.

```
> g10 <- affynames[1:10]
> gbacc <- multiget(g10, hu6800ACCNUM)
> genbank(gbacc, disp = "browser")
> gb <- genbank(gbacc[1], disp = "data")
```

Finally we consider interactions with LocusLink. Here, we use local data (LocusLink does not provide a good interface for automatic querying). We generate HTML output. This is often an easy way to communicate your results with other researchers working on the same analysis.

```
> llid <- multiget(affynames[1:10], hu6800LOCUSID)
> map <- multiget(affynames[1:10], hu6800MAP)
> symb <- multiget(affynames[1:10], hu6800SYMBOL)
> res <- data.frame(affynames[1:10], cbind(unlist(symb), unlist(map)))
> names(res) <- c("Affy ID", "Gene symbol", "Chromosomal location")
> ll.htmlpage(llid, filename = "LocusLink.html", title = "HTML report",
+   othernames = res, table.head = c("LocusID", names(res)),
+   table.center = TRUE)
```