Seminar III: R/Bioconductor

Leonardo Collado Torres
lcollado@lcg.unam.mx
Bachelor in Genomic Sciences
www.lcg.unam.mx/~lcollado/

August - December, 2009

Public Data

- Intro
- biomaRt
- GEOquery
- ArrayExpress
- annotate
- KEGG

To start off

- On this class we'll learn how to download public data using tools such as biomaRt, GEOquery and ArrayExpress
- Please install the following packages if you are working on your laptop.

```
> source("http://bioconductor.org/biocLite.R")
> biocLite(c("biomaRt", "GEOquery",
+     "ArrayExpress"))
> biocLite(c("annotate", "KEGG.db",
+     "KEGGSOAP"))
```

- You might need to install dependencies such as `RCurl` and `XML` for `biomaRt`.

Why learn to use these packages?

- Simply, because lots of data is publicly available.
- These tools help you avoid getting lost while fetching for data.
- Once the data is in R. . . graphs, statistical tests, etc.

Announcements

- Assitance schedule defined:
  1. José on Tuesday 3 to 5
  2. Víctor on Wednesday 3 to 5
  3. Alejandro on Thursday 3 to 5
- Use Montealban preferably
- 1st homework :)

Intro

- We'll dedicate most of the class to this package.
- Who is the author? He works with James Bullard :)
- There is a recent and very neat paper on biomaRt which uses other packages: lattice, affy and gplots.
  `http://www.ncbi.nlm.nih.gov/pubmed/19617889`

What is biomart?

`http://www.biomart.org/`

- An `OCR` and `EBI` initiative.
- Access to 31 databases and growing![1]
- Not too hard to use :) You build a mart by choosing a database, some filters and you retrieve some attributes.
- Which databases do you find more *interesting*?

[1] 29 back in July 09

ENSEMBL

- One of the biggest public databases!
- Genes, GOs, regions, expression info, proteins, etc.
- If you want to learn more, take a look at the tutorials site.
- Pay attention to Module 5: BioMart and the BioMart section :)

InterPro

- Huge one as well!
- Integrates data from several databases.
- Taxonomy, proteins, ...
- More info at the InterPro and the mart help sites.

biomaRt

- Overall, BioMart is a web service tool to obtain tabular data.
- Is biomaRt the only way to access BioMart?

listMarts

- biomaRt basically builds SQL queries for you and is more simple to use that doing the queries directly.
- To start, load the library and lets look at the available databases:

```
> library(biomaRt)
> head(listMarts())
```

```
              biomart                                    version
1             ensembl                ENSEMBL 55 GENES (SANGER UK)
2                 snp           ENSEMBL 55 VARIATION  (SANGER UK)
3 functional_genomics ENSEMBL 55 FUNCTIONAL GENOMICS (SANGER UK)
4               vega                          VEGA 35  (SANGER UK)
5                msd                     MSD PROTOTYPE (EBI UK)
6  bacterial_mart_54          ENSEMBL BACTERIA 54 GENES (EBI UK)
```

Datasets

- Once we know which *mart* to use, we select it using useMart.
- Then, we can take a look at the available datasets.

```
> mart <- useMart("bacterial_mart_54")
> head(listDatasets(mart))
```

```
          dataset
1    str_57_gene
2    esc_20_gene
3 myc_25994_gene
4 sta_29522_gene
5     bac_6_gene
6 esc_31791_gene
                                    description
1 Streptococcus pneumoniae TIGR4 genes (EB 1)
```

Datasets

```
2  Escherichia coli RIMD 0509952 genes (EB 1)
3        Mycobacterium bovis BCG genes (EB 1)
4      Staphylococcus aureus JH1 genes (EB 1)
5              Bacillus subtilis genes (EB 2)
6          Escherichia coli SE11 genes (EB 1)
  version
1    EB 1
2    EB 1
3    EB 1
4    EB 1
5    EB 2
6    EB 1
```

Filters

- Now, we load the dataset with useDataset or we re-use useMart to subset our mart.
- And then we explore the available filters.

```
> bsub <- useDataset("bac_6_gene",
+     mart = mart)
> bsub <- useMart("bacterial_mart_54",
+     dataset = "bac_6_gene")
> head(listFilters(bsub))
```

Filters

```
                  name
1    chromosome_name
2              start
3                end
4             strand
5 chromosomal_region
6  with_arrayexpress
              description
1         Chromosome name
2         Gene Start (bp)
3           Gene End (bp)
4                 Strand
5      Chromosome Regions
6 with ArrayExpress ID(s)
```

Filters

- A filter is our query, meaning, what we know.
- How many filters does our dataset have?

Attributes

- We also need to choose what we want to know: the attributes.

```
> head(listAttributes(bsub))
```

```
                      name
1       ensembl_gene_id
2 ensembl_transcript_id
3    ensembl_peptide_id
4          description
5       chromosome_name
6        start_position
             description
1       Ensembl Gene ID
2 Ensembl Transcript ID
3    Ensembl Protein ID
```

Attributes

```
4           Description
5    Chromosome/plasmid
6       Gene Start (bp)
```

- How may attributes we could potentially retrieve on this case?

Class exercise

- Using our bsub `Mart` object, retrieve the genes from the first 100000pb on the Bacillus chromosome with getBM[2].
- For every gene, get the start position, end position, strand, and *status*.
- Then use lattice and make a plot using `xyplot` with one panel for every type of status, the start on the x axis and the end on the y axis. Plot the points with different colors for every strand.

[2] `getBM` is the main `biomaRt` function

Solution: getting the data

- The tricky part is using a `list` for the filter values; a must when using more than one filter.

```
> res <- getBM(attributes = c("start_position",
+     "end_position", "strand", "status"),
+     filters = c("start", "end"),
+     values = list("1", "100000"),
+     mart = bsub)
> head(res)
```

Solution: getting the data

	start_position	end_position	strand
1	43921	44799	1
2	64099	64635	1
3	40213	40653	1
4	55866	56159	1
5	25221	25766	-1
6	53183	53368	1

	status
1	KNOWN
2	KNOWN
3	KNOWN
4	KNOWN
5	KNOWN
6	KNOWN

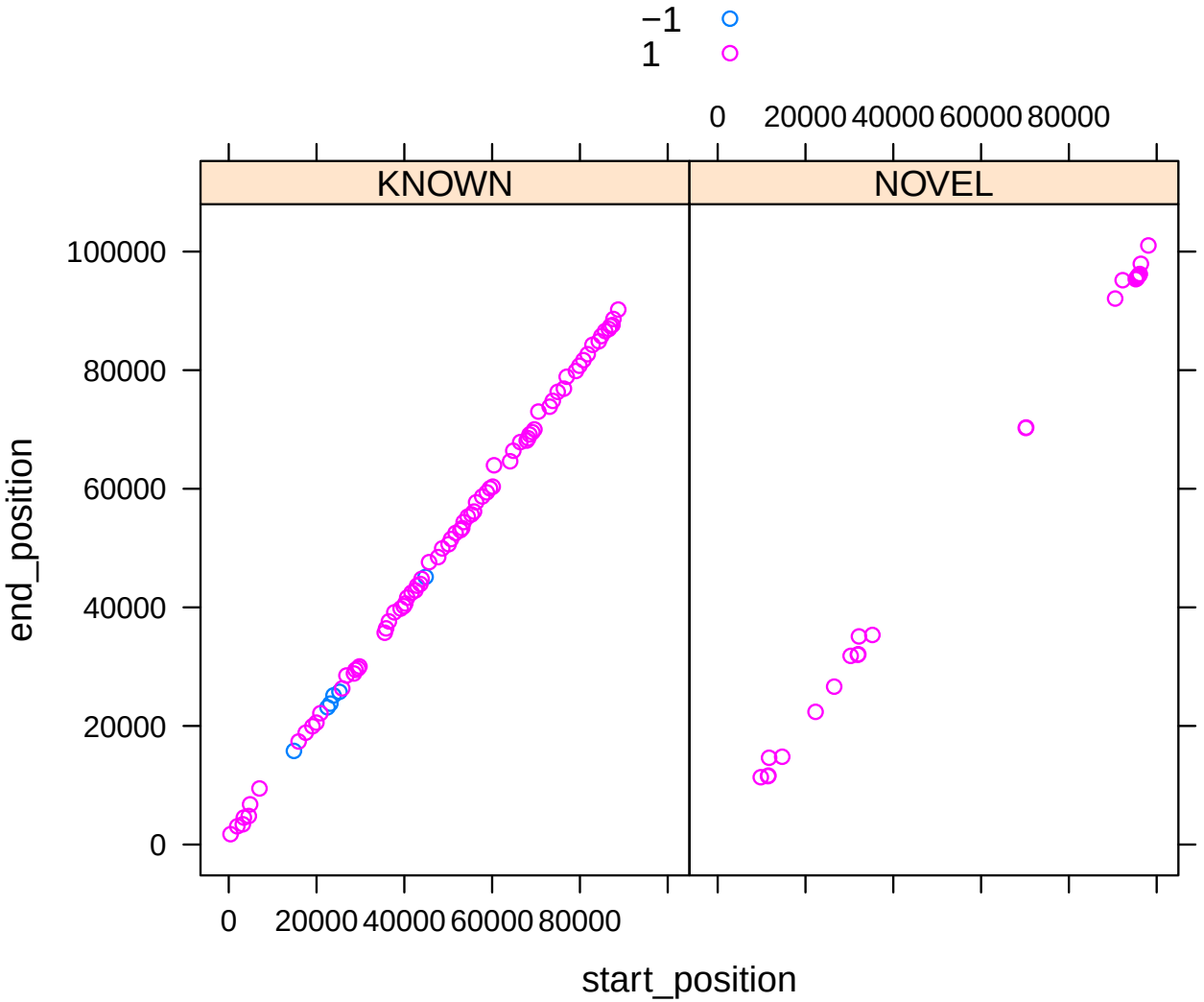
Solution: getting the data

- How many genes did we get?

Solution: plot

```
> library(lattice)
> print(xyplot(end_position ~ start_position |
+     status, group = strand, data = res,
+     auto.key = T))
```

Solution: plot



Class Exercise II

- Use biomaRt to access the Homo Sapiens dataset from the `ENSEMBL` database.
- For `entrez` gene ids 999, 1000 and 1001 get the first 100 upstream bases.
- Use the getSequence function.
- Set the `seqType` argument equal to *coding_gene_flank*.
- What is the GC percentage for every upstream sequence? Use gsub and nchar.[3]

[3] The answer is 72, 80, and 75

For further learning

- The biomaRt vignette is quite complete and presents several *tasks* which are great for learning.
- Additionally, Steffen gave a lab at BioC2009 which has more tasks.
- Then again, the paper I mentioned earlier is quite elegant :)

Intro

`www.ncbi.nlm.nih.gov/geo/`

- GEO, or the Gene Expression Omnibus, is a public data repository hosted at NCBI that mainly contains microarray data meeting MIAME requirements.
- There is some SAGE, mass spec data, and some high throughput data.
- GEOquery is simply the package to access this database from R.[4]

[4] As far as I know, it doesn't work with HTS data for now

Overview

- The main function is getGEO.

```
> library(GEOquery)
```

```
> `?`(getGEO)
```

An example

- As it says on the package help, GEOquery is the bridge between R and GEO.
- Lets look at some recent data.
- Download the 2nd sample data.

```
> info <- getGEO("GSM377792")
```

```
File stored at:
/tmp/RtmpJTF9ag/GSM377792.soft
```

- Then check the attributes of our info object

```
> attributes(info)
```

Cont.

- What was it last updated?
- What organism did they study?
- What type of data is it?
- How old?
- Tissue type?
- Number of unique tags?

More info

- You might want to save the data on a non-temp folder. To do so use the destdir arg.
- GDS data can be transformed into expression sets.
- If you work with microarrays, this package will be *very* useful to you :)

Intro

`www.ebi.ac.uk/microarray-as/ae/`

- It's another public repository for public functional genomics data.
- Quite a few microarrays as well

Querying

- First of all, you might want to look up for related data sets.
- We can do so using queryAE:[5]

```
> library(ArrayExpress)
> sets = queryAE(keywords = "pneumonia",
+     species = "homo+sapiens")
```

- How many sets did we get? Check the `class` of `sets` first.
- When were multiple sets released the same day? We know that a `unique` function exists, so using apropos find out the quickest solution :)

[5] Note the use of a + for multiple words

ArrayExpress

- Once you identify the set you want to download, you can get it with ArrayExpress
- Note that one of its arguments is useful if you don't want to lose the data once you close the R session.

```
> rawset = ArrayExpress("E-MEXP-1422")
```

- Around 15 mb of data on this case

Whole data

- The previous function extracts some data from the array files, and if you want the whole data then you need to use getAE

```
> mexp1422 = getAE("E-MEXP-1422",
+      type = "full")
```

Loading into R

- Once you have downloaded the files, you need to load them in R.
- magetab2bioc does the job for you :)

```
> rawset = magetab2bioc(files = mexp1422)
```

- On this case, `rawset` will be an `AffyBatch` object.
- For processed data, you'll need to use procset.

We'll be back

- We'll most likely re-use `GEOquery` and `ArrayExpress` on our microarray related classes.

Intro

- Great to interact with NCBI!
- It uses `XML` heavily
- You can download info on papers, sequences, make links to NCBI and much more :)
- There is a drawback. . . if you access NCBI too frequently you'll get banned.

Getting a sequence

- For example, we can download sequences as `character` objects using getSEQ

```
> library(annotate)
```

```
> seq <- getSEQ("CY045495.1")
```

- How long is our `seq` object?

NCBI links

- For `getSEQ` I used a accession number, and if we want to find the related UID, simply use accessionToUID.
- Then, we can construct a link to NCBI using getQueryLink

```
> id <- accessionToUID("CY045495.1")
> id
[1] "257127071"
> getQueryLink(id, repository = "gb")
[1] "http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=N
```

- Where did our sequence come from?[6]
- Does the length of `seq` match the reported length?
- Find a way to get the NCBI link by using the accession number directly.

[6] Check the `definition`

Paper info

- By using the functions pubmed, xmlRoot and buildPubMedAbst we can get info such as abstracts, authors, ...
- Lets find the authors of our `sets` object from the `ArrayExpress` section.

```
> ids <- (sets[, "PubmedID"])
> ids <- as.character(ids[ids !=
+     "NA"])
> x <- pubmed("19679224")
> a <- xmlRoot(x)
> abs <- buildPubMedAbst(a[[1]])
```

- Once we have our `abs` object[7], we can retrieve information using its attributes.

Paper info

```
> pubDate(abs)
[1] "Aug 2009"
```

- How many authors does the first paper have?

[7] Note that I only built the abstract for the 1st paper

End

- By using `annotate`, its relatively easy to query abstracts for a gene name or some other keyword.
- I encourage you to explore it :)

Quick overview

`www.genome.jp/kegg/kegg2.html`

- Finally, we'll take a very quick look at the KEGG packages.

```
> library(KEGG.db)
> library(KEGGSOAP)
> apropos("KEGG")
```

```
 [1] "KEGG"
 [2] "KEGG2heatmap"
 [3] "KEGG_dbconn"
 [4] "KEGG_dbfile"
 [5] "KEGG_dbInfo"
 [6] "KEGG_dbschema"
 [7] "KEGGENZYMEID2GO"
 [8] "KEGGEXTID2PATHID"
```

Quick overview

```
 [9] "KEGGGO2ENZYMEID"
[10] "KEGGMAPCOUNTS"
[11] "KEGGmnplot"
[12] "KEGGPATHID2EXTID"
[13] "KEGGPATHID2NAME"
[14] "KEGGPATHNAME2ID"
[15] ".__M__KEGG2heatmap:annotate"
[16] ".__M__KEGGmnplot:annotate"
[17] ".__T__KEGG2heatmap:annotate"
[18] ".__T__KEGGmnplot:annotate"
```

- What is the name for the Path id 00010?

Quick answer

- Easy, just use KEGGPATHID2NAME:

```
> KEGGPATHID2NAME$"00010"
```

```
[1] "Glycolysis / Gluconeogenesis"
```

Genes

- Next, if we want to find the genes involved on a certain pathway, we use get.genes.by.pathway[8]

```
> genes <- get.genes.by.pathway("path:eco00010")
```

- How many genes did we get?
- For a deeper learning use:

```
> help(package = KEGGSOAP)
> help(package = KEGG.db)
```

[8] quite a long name!

SessionInfo

```
> sessionInfo()

R version 2.10.0 Under development (unstable) (2009-07-25 r48998)
i686-pc-linux-gnu

locale:
 [1] LC_CTYPE=en_US.UTF-8
 [2] LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=C
 [6] LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8
 [8] LC_NAME=C
 [9] LC_ADDRESS=C
[10] LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8
[12] LC_IDENTIFICATION=C

attached base packages:
```

SessionInfo

```
[1] stats     graphics  grDevices
[4] utils     datasets  methods
[7] base

other attached packages:
 [1] KEGGSOAP_1.19.1
 [2] KEGG.db_2.3.0
 [3] RSQLite_0.7-1
 [4] DBI_0.2-4
 [5] XML_2.6-0
 [6] annotate_1.23.1
 [7] AnnotationDbi_1.7.7
 [8] ArrayExpress_1.5.5
 [9] GEOquery_2.9.4
[10] RCurl_0.98-1
[11] bitops_1.0-4.1
[12] Biobase_2.5.5
[13] lattice_0.17-25
[14] biomaRt_2.1.0
```

SessionInfo

```
loaded via a namespace (and not attached):
[1] affy_1.23.5
[2] affyio_1.13.3
[3] grid_2.10.0
[4] limma_2.19.2
[5] preprocessCore_1.7.4
[6] SSOAP_0.4-6
[7] xtable_1.5-5
```