Bioconductor & Biomart Tutorial

SGDP Summer School

Dr Amos Folarin

Session

1. Overview of
   - the bioconductor website www.bioconductor.org
   - documentation and help etc.
   - installing bioconductor packages

2. Using bioconductor to access annotation
   - biomaRt -- programmatic access to biological annotation

3. A couple of tutorials

This presentation is available as a google doc, easier to copy and paste from than the pdf

https://docs.google.com/presentation/d/1Z0R9mJtmgU2sQkJLNNszqraxcxDms17r9LfK1jDUXf8/pub?start=false&loop=false&delayms=3000

NOTE: on linux to paste to the terminal where R will be running:

Note to class: Middle Button, or Ctrl+Shift+V to paste into a terminal

*Fix for kubuntu pendrive machines

missing font adobe-helvetica
Open a terminal and paste in this script:

```
cd /usr/share/fonts/100dpi/
sudo mkfontdir
xset fp+ /usr/share/fonts/100dpi
cat >> ~/.xinitrc <<< FontPath /usr/share/fonts/100dpi
```

this will allow certain plots to work.

Bioconductor

Bioconductor "Bioc"

- What is bioconductor?
  - Like CRAN, bioconductor is one of the major repositories of R packages, in this case particularly focused on biology
  - In it you will 3 types of bioc packages:
    - Software
    - Annotation Data (meta-data)
    - Experimental Data
- Bioc community resources
  - tutorials, mailing lists etc.
- New version released twice a year. A dev version also available, which includes new packages under assessment.

Bioc objectives

- **Statistical and graphical methods**. The Bioconductor project provides access to powerful statistical and graphical methods for the analysis of genomic data. Analysis packages address workflows for analysis of oligonucleotide arrays, sequence analysis, flow cytometry and other high-throughput genomic data.
- **Documentation and reproducible research**. Each Bioconductor package contains one or more vignettes, documents that provide a textual, task-oriented description of the package's functionality.
- **Annotation**. The Bioconductor project provides software for associating microarray and other genomic data in real time with biological metadata from web databases such as GenBank, Entrez genes and PubMed (annotate package).
- **Bioconductor short courses**. The Bioconductor project has developed a program of short courses on software and statistical methods for the analysis of genomic data.
- **Open source**.
- **Open development**. Users are encouraged to become developers, either by contributing Bioconductor compliant packages or documentation.

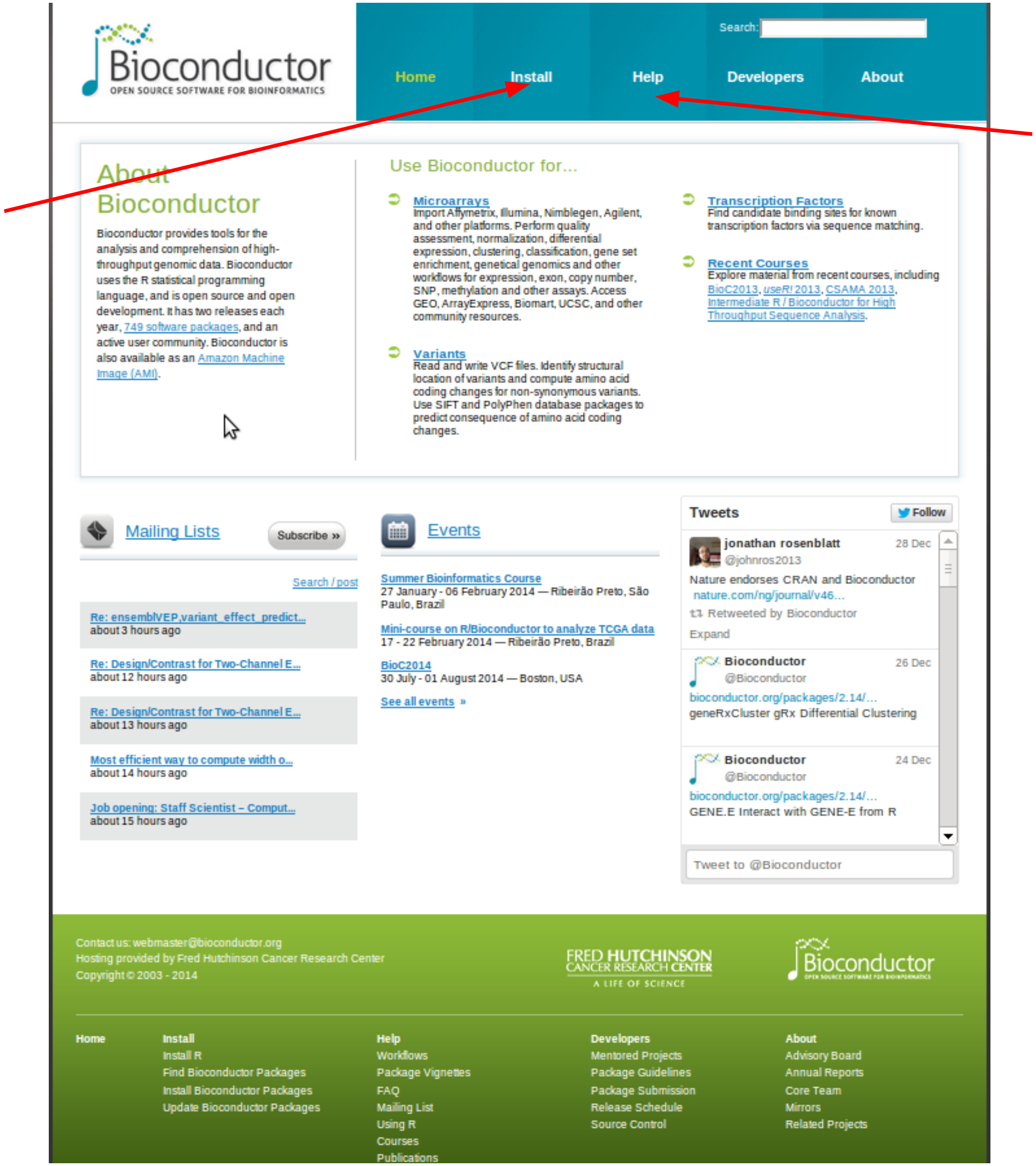
Bioconductor Website

PACKAGES

install bioc
Software packages
Annotation Data
(Genome, Array, etc.)
Experiment Data
Latest Release
Announcement
& prev. versions

WORKFLOWS

e.g.
Oligonucleotide Arrays
High-throughput
Sequencing
Annotation
Annotating Ranges
Variants
Flow Cytometry and
other assays
Finding Candidate
Binding Sites for Known
Transcription Factors
via Sequence Matching


Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS
Search:
Home
Install
Help
Developers
About
Subscribe »
Follow
FRED HUTCHINSON CANCER RESEARCH CENTER
A LIFE OF SCIENCE
Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

About Bioconductor

Bioconductor provides tools for the analysis and comprehension of high-throughput genomic data. Bioconductor uses the R statistical programming language, and is open source and open development. It has two releases each year, 749 software packages, and an active user community. Bioconductor is also available as an Amazon Machine Image (AMI).

Use Bioconductor for...

Microarrays
Import Affymetrix, Illumina, Nimblegen, Agilent, and other platforms. Perform quality assessment, normalization, differential expression, clustering, classification, gene set enrichment, genetical genomics and other workflows for expression, exon, copy number, SNP, methylation and other assays. Access GEO, ArrayExpress, Biomart, UCSC, and other community resources.

Variants
Read and write VCF files. Identify structural location of variants and compute amino acid coding changes for non-synonymous variants. Use SIFT and PolyPhen database packages to predict consequence of amino acid coding changes.

Transcription Factors
Find candidate binding sites for known transcription factors via sequence matching.

Recent Courses
Explore material from recent courses, including BioC2013, *useR!* 2013, CSAMA 2013, Intermediate R / Bioconductor for High Throughput Sequence Analysis.

Mailing Lists

Search / post

Re: ensemblVEP,variant_effect_predict...
about 3 hours ago

Re: Design/Contrast for Two-Channel E...
about 12 hours ago

Re: Design/Contrast for Two-Channel E...
about 13 hours ago

Most efficient way to compute width o...
about 14 hours ago

Job opening: Staff Scientist – Comput...
about 15 hours ago

Events

Summer Bioinformatics Course
27 January - 06 February 2014 — Ribeirão Preto, São Paulo, Brazil

Mini-course on R/Bioconductor to analyze TCGA data
17 - 22 February 2014 — Ribeirão Preto, Brazil

BioC2014
30 July - 01 August 2014 — Boston, USA

See all events »

Tweets

jonathan rosenblatt @johnros2013 28 Dec
Nature endorses CRAN and Bioconductor nature.com/ng/journal/v46...
Retweeted by Bioconductor
Expand

Bioconductor @Bioconductor 26 Dec
bioconductor.org/packages/2.14/...
geneRxCluster gRx Differential Clustering

Bioconductor @Bioconductor 24 Dec
bioconductor.org/packages/2.14/...
GENE.E Interact with GENE-E from R

Tweet to @Bioconductor

Contact us: webmaster@bioconductor.org
Hosting provided by Fred Hutchinson Cancer Research Center
Copyright © 2003 - 2014

Home

Install
Install R
Find Bioconductor Packages
Install Bioconductor Packages
Update Bioconductor Packages

Help
Workflows
Package Vignettes
FAQ
Mailing List
Using R
Courses
Publications

Developers
Mentored Projects
Package Guidelines
Package Submission
Release Schedule
Source Control

About
Advisory Board
Annual Reports
Core Team
Mirrors
Related Projects

Package Vignettes

Mailing Lists

Courses & Conferences

Community Help Resources

Installing bioconductor and its packages

BiocLite.R

Installation of Bioconductor in R is done using an R script provided here `http://www.bioconductor.org/biocLite.R` : You first source it to make it locally available.

- `source("http://www.bioconductor.org/biocLite.R ")`
-
- To install the main bioconductor package "Biobase", call the main function with no arguments:
- `biocLite()`

- To install a single package script e.g. **biomaRt** package,
- `biocLite("biomaRt")`

- ** **NOTE** ** this is different to how one might install a **CRAN** package i.e.
- `install.package("gplots")`
-
- `library("biomaRt") # loads the package biomaRt into the R environment`

Also see: http://www.bioconductor.org/install/ for further details on installing bioc, e.g. recompiling packages from source and troubleshooting installs

Package API, docs and help

Getting help in R

https://docs.google.com/document/d/1KSZi85XM6ryrEbESj3VzY3dEaSs2ImTVuUZmpnldcUg/pub

get the documentation for a function, package, operator

```
?<string or 'special char'>
?mean
```

do a fuzzy string search of the help pages

```
??<string>
??pairs
```

will list all the functions in a package

```
library(help="stats")
```

run in built examples in documentation pages (available for most functions):

```
example(pairs) #most functions in R will have a runnable e.g.
demo(graphics) #not all packages have a demo
```

Vignettes

All bioc packages come with one or more vignettes, which are runnable tutorials for that package -- you can think of them as package specific workflows.

They are very handy when getting to grips with a new package.

```
vignette()  # lists vignettes for all loaded packages.
vignette(package="Biobase") #list all vignettes on a specific package
```

Open* a vignette from listed set:

```
vignette("ExpressionSetIntroduction", package="Biobase")
```

To run the Vignette files, sometimes not all the R code is rendered in the pdf, so you can extract it:

```
rcode <- vignette("ExpressionSetIntroduction", package="Biobase")
edit(rcode)
:q  #to quit edit
```

* note may need to change the default pdf viewer R uses to, i.e.

```
options(pdfviewer="/usr/bin/evince")
```

Bioconductor Workflows

A number of workflows are available in bioconductor which often combine the use of several packages to solve a particular common task. These are similar to CRAN taskviews http://cran.ma.imperial.ac.uk/web/views/

http://www.bioconductor.org/help/workflows/

Common Bioconductor workflows include:

Oligonucleotide Arrays
High-throughput Sequencing
Annotation
Annotating Ranges
Variants
Flow Cytometry and other assays
Finding Candidate Binding Sites for Known Transcription Factors via Sequence Matching

Tutorial #1 -- Bioconductor

This is a simple list of tasks to familiarise yourselves with bioc.

1. Look through the bioc website and try to install a new package using the biocLite.R script. You should be able to install locally in your home dir. Use function: "library()" to see which packages are already installed on the cluster.

2. Take a look at the package of your newly installed package structure, API, documentation and vignettes (see prev. slides and the bioconductor website)

Tutorial 1 - an example

#one I installed earlier on *you should be able to replicate this*, but feel free to try a different package.

```
source("http://bioconductor.org/biocLite.R")  #provide the biocLite function in R env.
biocLite("flowPeaks")  #install the bioconduconductor package flowPeaks
```

to see where this is installed:

```
.libPaths()  #the path on the /home directory is where your stuff will be installed unless you launched R as root.
```

```
library("flowPeaks")  #load the package into the R environment
```

```
vignette(package="flowPeaks")  #lists the vignettes
vignette("flowPeaks-guide", package="flowPeaks")  #open the vignette pdf
```

```
rcode <- vignette("flowPeaks-guide", package="flowPeaks")  #get vignette R code
edit(rcode)
:q     #to quit the editor
```

look at some API & documentation:

```
library(help="flowPeaks")  #list functions in package
```

```
?adjust.flowPeaks     #browse the API for a function adjust.flowPeaks
```

```
examples(flowPeaks)    #run the examples for the peaks function docs
```

Biomart

various R annotation package types

Gene centric AnnotationDbi packages:

- Organism level: `org.Mm.eg.db`
- Platform level: `hgu133plus2.db`
- System-biology level: `GO.db or KEGG.db`

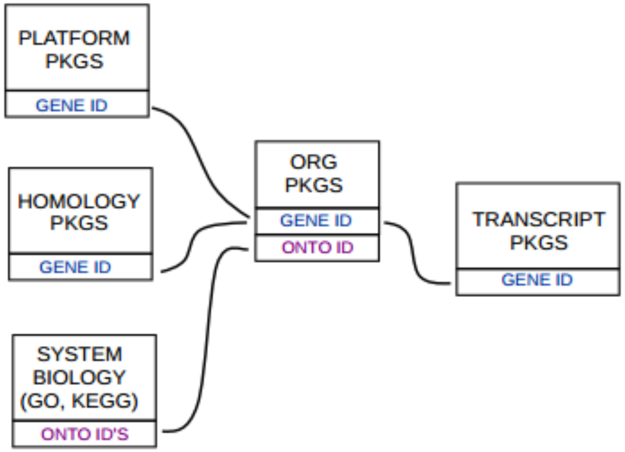
Genome centric GenomicFeatures packages:

- Transcriptome level: T`xDb.Hsapiens.UCSC.hg19.knownGene`
- Generic features: Can generate via GenomicFeatures

Not covering above, but information is available here http://www.bioconductor.org/help/course-materials/2011/BioC2011/LabStuff/AnnotationSlidesBioc2011.pdf

BiomaRt

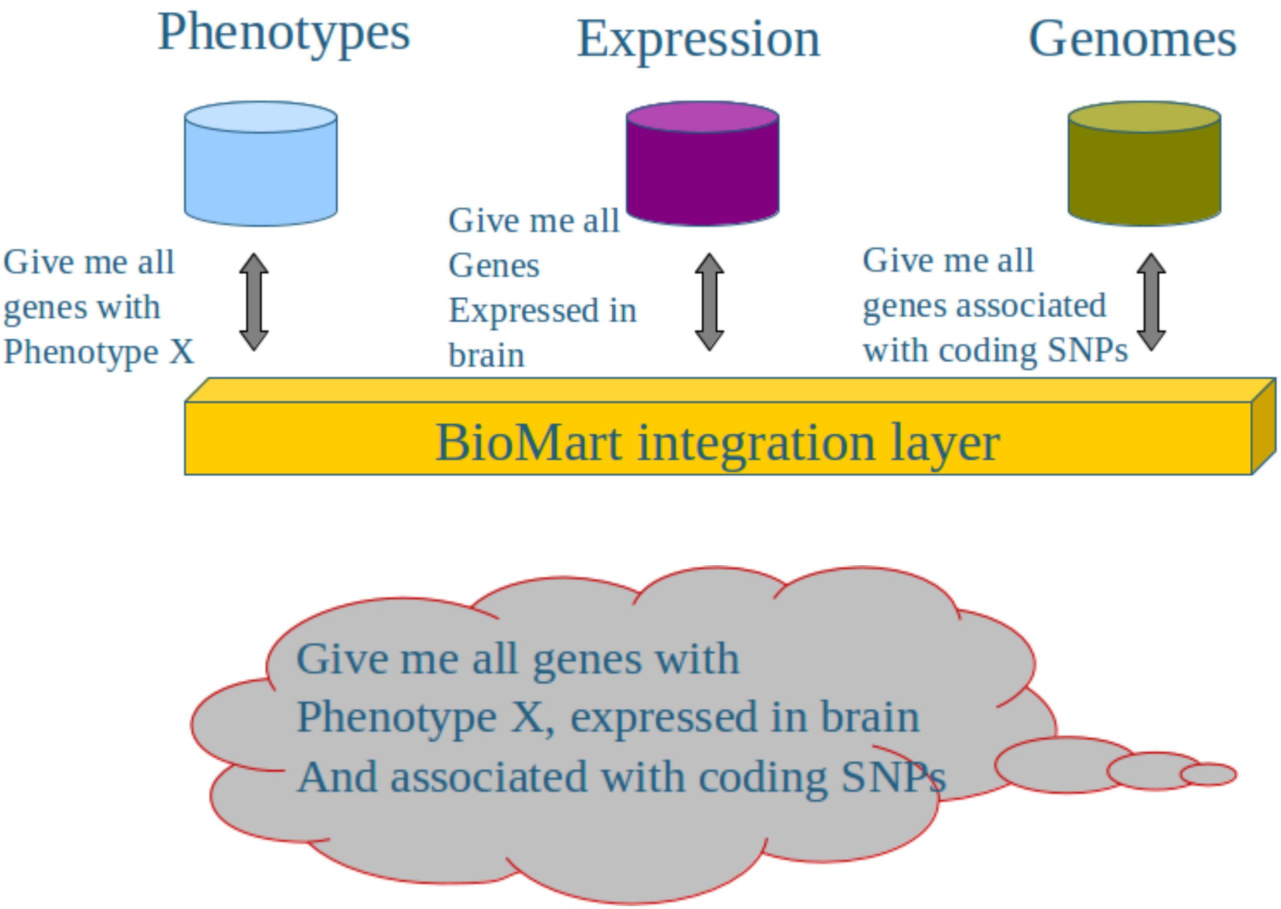
- an 'R' API into the biomart annotations



biological annotations are highly relational

Biomart

BioMart idea .....



Phenotypes

Expression

Genomes

...etc..

filters

Give me all genes with Phenotype X

Give me all Genes Expressed in brain

Give me all genes associated with coding SNPs

BioMart integration layer

Give me all genes with Phenotype X, expressed in brain And associated with coding SNPs

and list specific **attributes** of the returned genes

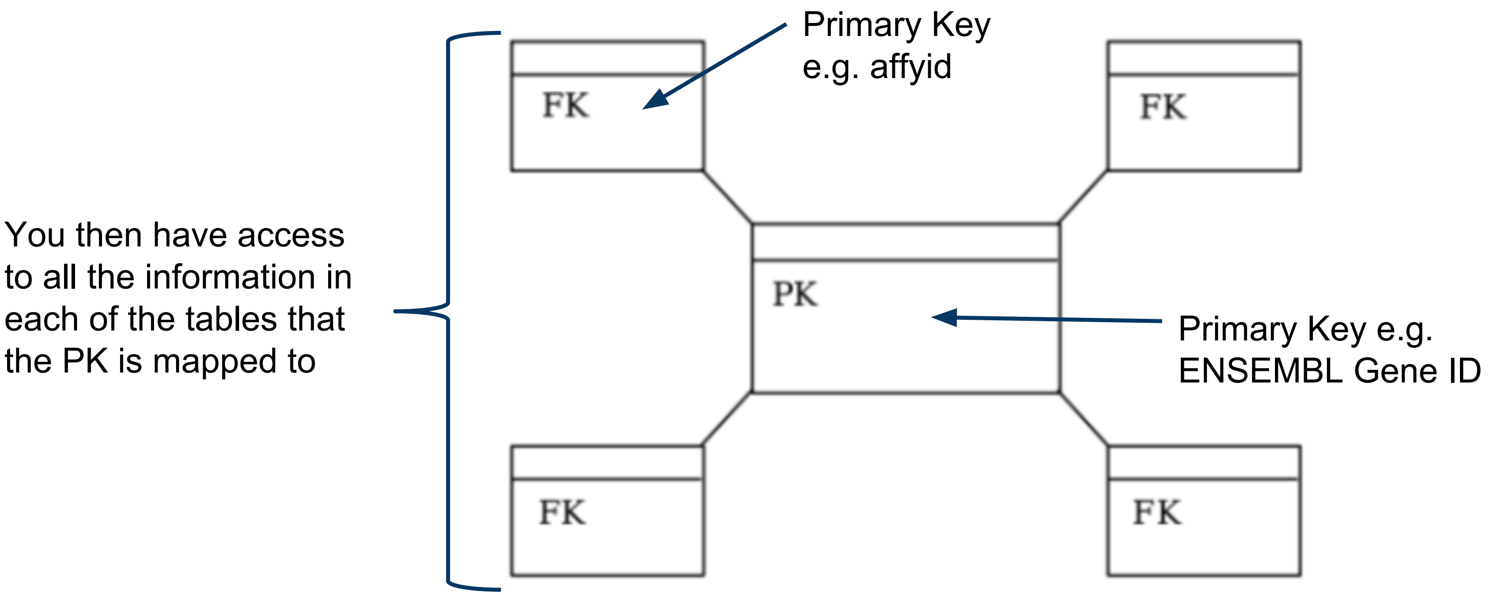
A Mart is a collection of datasets (~=Database).

Marts are optimised for querying.

A Dataset has a **main** table, with an entry and Primary Key (PK) for each of the items of interest in that dataset (eg PK → Mouse Transcripts ENSEMBL Gene ID).

Related bits of information about these items are hung off the table in **dimension** tables and are linked to the PK via the Foreign Key (FK) (eg. FK → Affy Id)

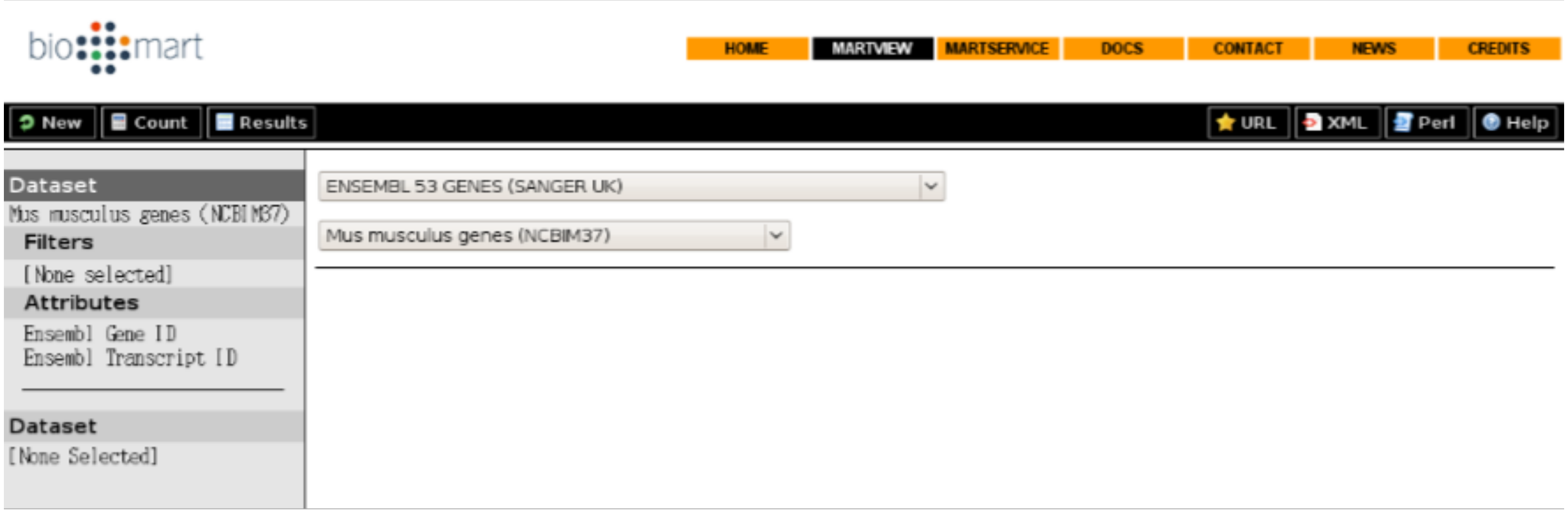
The Primary key maps to the Foreign Key i.e. **PK → FK → [linked info]**


Primary Key
e.g. affyid
FK
FK
You then have access to all the information in each of the tables that the PK is mapped to
PK
Primary Key e.g. ENSEMBL Gene ID
FK
FK

More Info: *http://www.biomart.org/user-docs.pdf*

Web Interface:

http://www.biomart.org/biomart/martview/



Dataset
Mus musculus genes (NCBIM37)
Filters
[None selected]
Attributes
Ensembl Gene ID
Ensembl Transcript ID

Dataset
[None Selected]

ENSEMBL 53 GENES (SANGER UK)

Mus musculus genes (NCBIM37)

Choose a Database (mart) to query (eg Ensembl)

Choose a Dataset from that mart to query (eg Mus Musculus Genes)

there are some tutorials available http://www.ensembl.org/info/website/tutorials/index.html

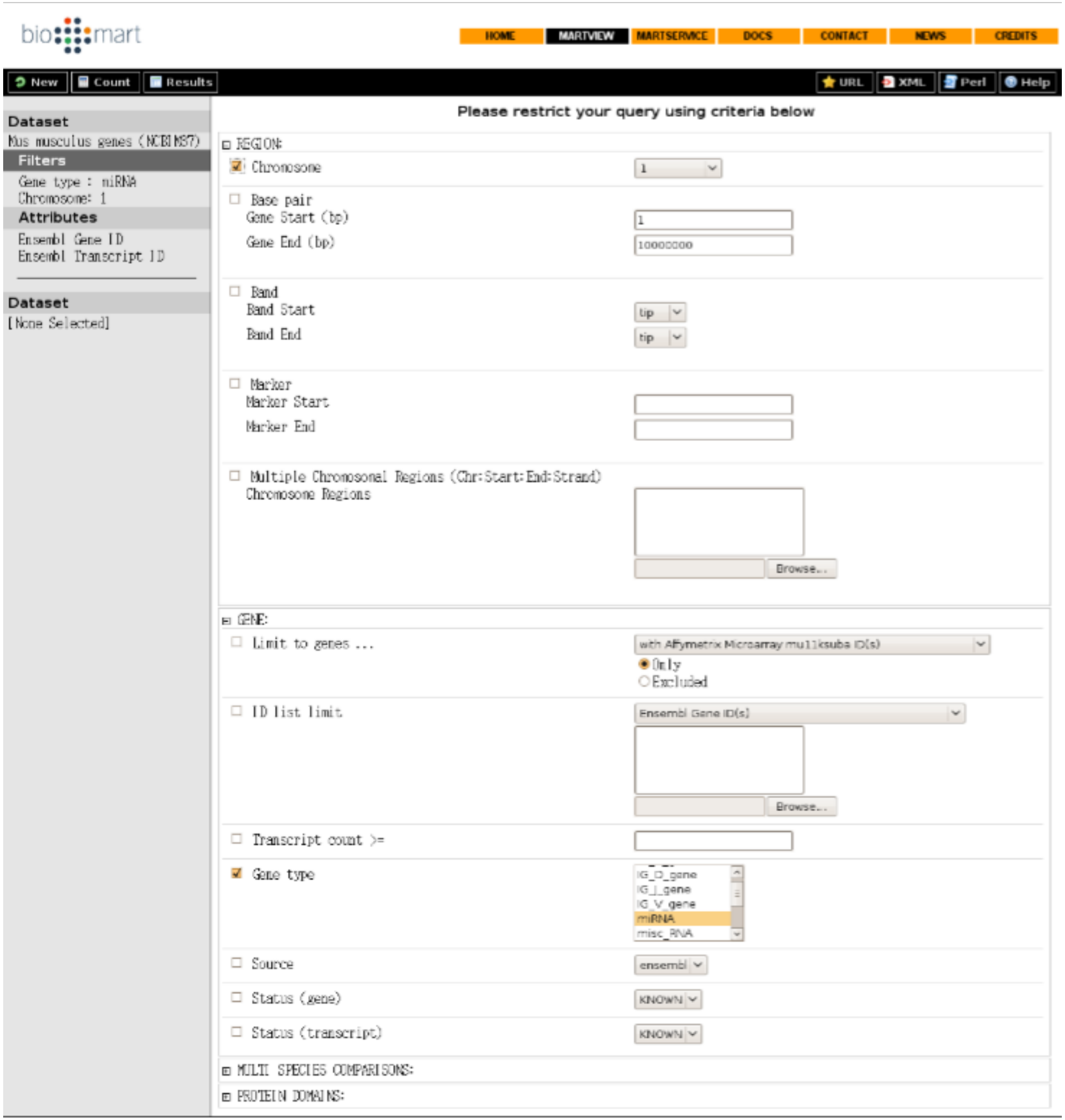
Filters

Use filters to select the members of the dataset in which you're interested

eg.

Limit to *miRNA* genes from *Chr1*

→



Dataset
Mus musculus genes (NCBIM37)

Filters
Gene type : miRNA
Chromosome: 1

Attributes
Ensembl Gene ID
Ensembl Transcript ID

Dataset
[None Selected]

Please restrict your query using criteria below

REGION:

☑ Chromosome: 1

□ Base pair
Gene Start (bp): 1
Gene End (bp): 10000000

□ Band
Band Start: tip
Band End: tip

□ Marker
Marker Start:
Marker End:

□ Multiple Chromosomal Regions (Chr:Start:End:Strand)
Chromosome Regions:
Browse...

GENE:

□ Limit to genes ... : with Affymetrix Microarray mu11ksuba ID(s)
● Only
○ Excluded

□ ID list limit: Ensembl Gene ID(s)
Browse...

□ Transcript count >=

☑ Gene type: IG_D_gene, IG_J_gene, IG_V_gene, miRNA, misc_RNA

□ Source: ensembl

□ Status (gene): KNOWN

□ Status (transcript): KNOWN

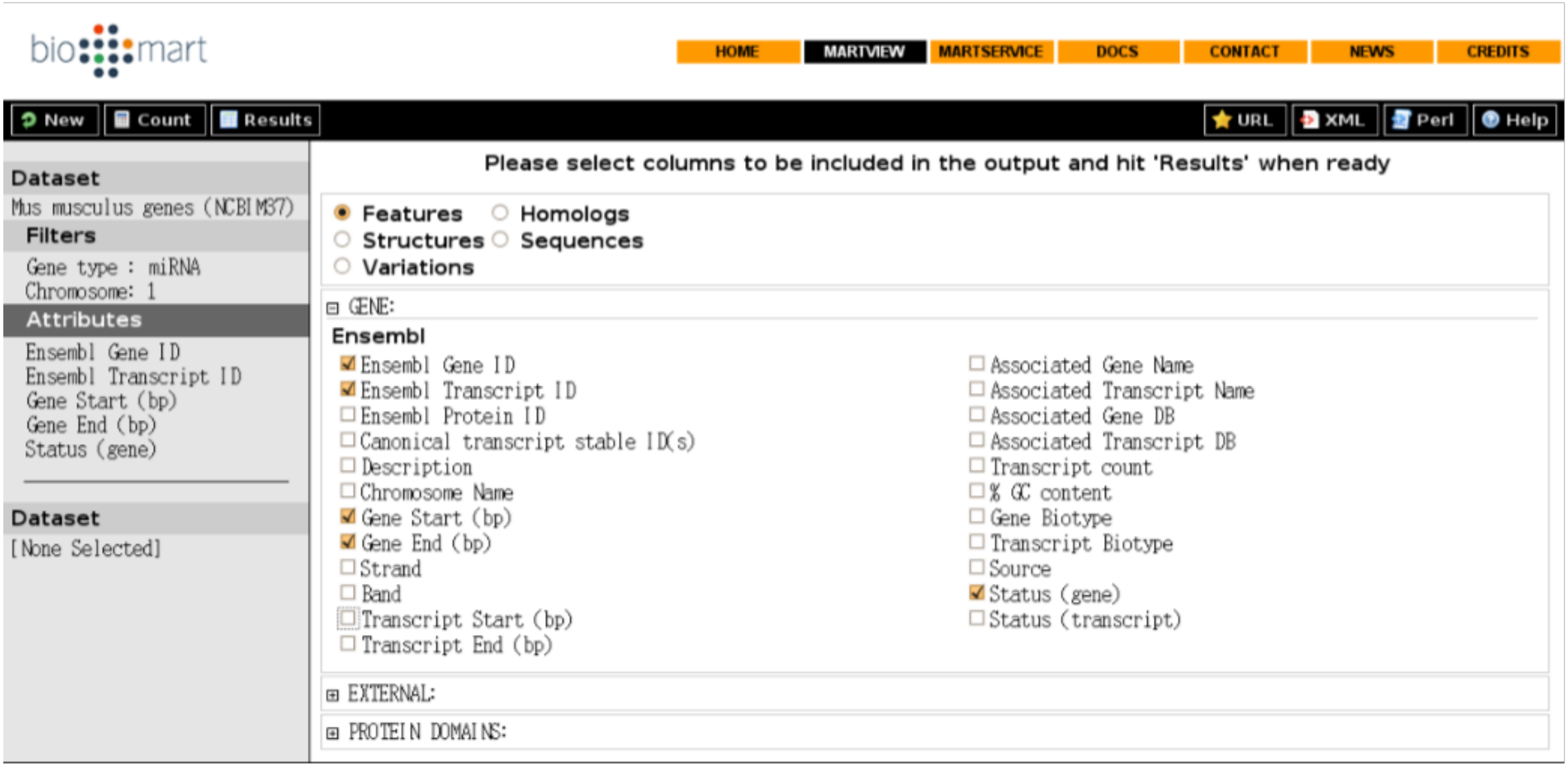
MULTI SPECIES COMPARISONS:

PROTEIN DOMAINS:

Attributes

Use attributes to define what bits of information you want to retrieve about the members of the dataset

eg. Gene ID, Transcript ID, Start, End and Status:



Dataset

Mus musculus genes (NCBIM37)

Filters

Gene type : miRNA
Chromosome: 1

Attributes

Ensembl Gene ID
Ensembl Transcript ID
Gene Start (bp)
Gene End (bp)
Status (gene)

Dataset

[None Selected]

Please select columns to be included in the output and hit 'Results' when ready

● Features ○ Homologs
○ Structures ○ Sequences
○ Variations

⊟ GENE:

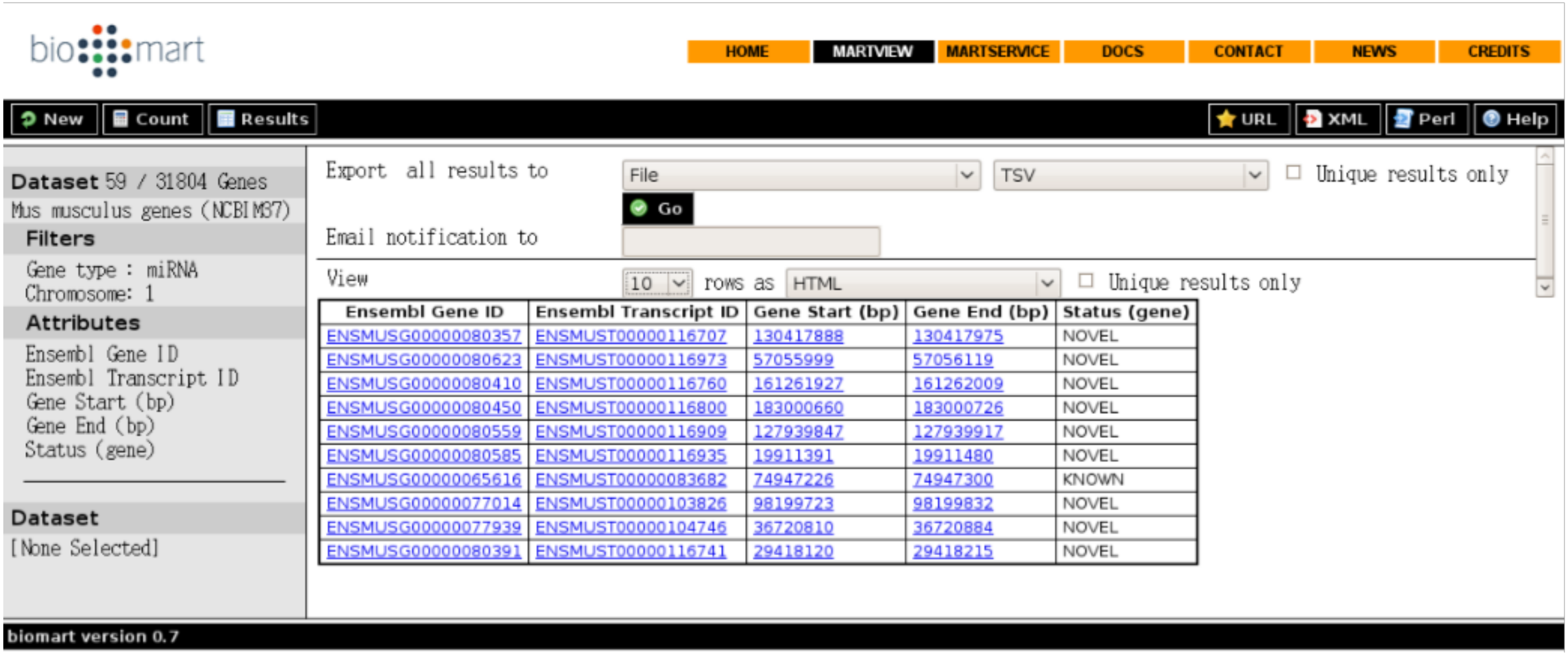
Ensembl

- ☑ Ensembl Gene ID
- ☑ Ensembl Transcript ID
- □ Ensembl Protein ID
- □ Canonical transcript stable ID(s)
- □ Description
- □ Chromosome Name
- ☑ Gene Start (bp)
- ☑ Gene End (bp)
- □ Strand
- □ Band
- □ Transcript Start (bp)
- □ Transcript End (bp)
- □ Associated Gene Name
- □ Associated Transcript Name
- □ Associated Gene DB
- □ Associated Transcript DB
- □ Transcript count
- □ % GC content
- □ Gene Biotype
- □ Transcript Biotype
- □ Source
- ☑ Status (gene)
- □ Status (transcript)

⊞ EXTERNAL:

⊞ PROTEIN DOMAINS:

Results:



Dataset 59 / 31804 Genes
Mus musculus genes (NCBIM37)

Filters
Gene type : miRNA
Chromosome: 1

Attributes
Ensembl Gene ID
Ensembl Transcript ID
Gene Start (bp)
Gene End (bp)
Status (gene)

Dataset
[None Selected]

Export all results to File TSV □ Unique results only

Email notification to

View 10 rows as HTML □ Unique results only

Ensembl Gene ID	Ensembl Transcript ID	Gene Start (bp)	Gene End (bp)	Status (gene)
ENSMUSG00000080357	ENSMUST00000116707	130417888	130417975	NOVEL
ENSMUSG00000080623	ENSMUST00000116973	57055999	57056119	NOVEL
ENSMUSG00000080410	ENSMUST00000116760	161261927	161262009	NOVEL
ENSMUSG00000080450	ENSMUST00000116800	183000660	183000726	NOVEL
ENSMUSG00000080559	ENSMUST00000116909	127939847	127939917	NOVEL
ENSMUSG00000080585	ENSMUST00000116935	19911391	19911480	NOVEL
ENSMUSG00000065616	ENSMUST00000083682	74947226	74947300	KNOWN
ENSMUSG00000077014	ENSMUST00000103826	98199723	98199832	NOVEL
ENSMUSG00000077939	ENSMUST00000104746	36720810	36720884	NOVEL
ENSMUSG00000080391	ENSMUST00000116741	29418120	29418215	NOVEL

biomart version 0.7

BiomaRt (R package)

biomaRt is an R interface to Biomart (http://www.biomart.org/), a system for integrating across a wide range of biological annotation databases.

biomaRt package:

- uses "marts" these are databases that have implemented the biomart interface
- Query web-based 'biomart' resource for genes, sequence, SNPs, and etc.

Creating a Biomart Query

Documentation:

http://www.bioconductor.org/packages/2.12/bioc/html/biomaRt.html

http://www.bioconductor.org/packages/2.12/bioc/vignettes/biomaRt/inst/doc/biomaRt.pdf

Two main components of biomart are:

marts which are a composition of **datasets**

e.g.

ensembl is a mart → *hsapiens_gene_ensembl* is a dataset in ensembl

```
listMarts()    # lists the available marts
listMarts(archive=TRUE)    #previous freezes of databases you can use
ensembl.m <- useMart(biomart="ensembl")    # select a mart to inspect
```

```
listDatasets(ensembl.m)    # list the datasets in the mart "ensembl"
```

# once you know the dataset you want, you can specify it when you create your mart

```
ensembl.m <- useMart(biomart="ensembl",dataset="hsapiens_gene_ensembl")
```

Now we have a mart for a specific dataset, we are ready to start building a query

Creating a Biomart Query

Like the web-interface there are **3 parts** to a query **Attributes** and **Filters and the filter Values.** This can be listed for our dataset mart, ensembl.m.

attributes: are what you want to retrieve. A vector of attributes e.g. ensembl_gene_id

Filters: are Property of the attribute. A vector of Filters that one that are used to qualify or constrain the attributes.

Values: values for the Filters. A list of vectors, where each position in the list corresponds to the position of the Filter in the Filter argument

(see examples below).

i.e. I want back a set of **Attributes**, which I will constrain by a set of **Filters** that take these **Values**

Information on Attributes

See the attributes available on your specific mart dataset:

```
listAttributes(ensembl.m)
```

or for easier browsing:

```
edit(listAttributes(ensembl.m))
```

or, search for a specific thing:

```
grep(pattern="text", listAttributes(ensembl.m)[ ,1] )  #e.g. pattern="snp"
```

Attributes are grouped by category of information in here:

```
attributePages(ensembl.m)
```

```
[1] "feature_page"     "structure"        "transcript_event" "homologs"
[5] "snp"              "sequences"
```

You can then display attributes of a particular page category:

```
listAttributes(ensembl.m, page="snp")
```

Information on Filters

See the filters available on your specific mart dataset:

```
listFilters(ensembl.m)
```

Provides the type of the filter e.g. (boolean, char, vector, text, etc..)

```
filterType("start", ensembl.m)
```

Provides the types of thing you can filter

```
filterOptions("chromosome_name", ensembl.m)
```

Make a query on the mart

The syntax for the main query function for bioMaRt:

```
getBM( attributes=c(,,), filters = c(,,), values =list(c
(,,),...), mart= )
```

A biomart query will involve one or more **attributes** and list of **filters + their values**.

```
affyids=c("202763_at","209310_s_at","207500_at")

getBM(attributes=c("affy_hg_u133_plus_2", "entrezgene", "uniprot_genename"), filters =
"affy_hg_u133_plus_2", values = affyids, mart = ensembl.m)
```

This query would give you the all the

Attributes: affy_hg_u133_plus_2 ids, entrezgene ids, uniprot genenames

restricted by the Filter: affy_hg_u133_plus_2

where the filter takes these Values: "202763_at","209310_s_at","207500_at"

EXERCISE: take a look at the available attributes with listAttributes(), and show some more attributes for the affymetrix probeset in the example code above.

An example -- Gene Ontology category annotation

Find all genes that match a particular Gene Ontology category:

We can browse http://amigo.geneontology.org to a GO category, get the id and enter the query:

Then construct our query:

```
getBM(c("entrezgene","hgnc_symbol"), filters="go_id",
values="GO:0004707", mart=ensembl.m)
```

EXERCISE: Try this with new GO categories and attributes.

"snp" biomart

We will use a new mart here.

load the "snp" biomart and look at its datasets:

```
snp.mart <- useMart("snp")
edit(listDatasets(snp.mart))
```

now rebuild the mart with the `hsapiens_snp` dataset:

```
snp.mart <- useMart(biomart="snp", dataset="hsapiens_snp")
```

take a look at the dataset's attributes & filters

```
edit(listAttributes(snp.mart))
edit(listFilters(snp.mart))
```

Get some annotations on a set of SNPs

```
snps <- c("rs769449", "rs514716", "rs514716", "rs9877502",
"rs514716", "rs6922617")
```

```
snp.q <- getBM(attributes=c("refsnp_id","allele","
chrom_start","ensembl_gene_stable_id"), filters=c
("snp_filter"), values=list(snps), mart=snp.mart)


snp.q
```

Get all the SNPs between two chromosome positions:

```
snp.q2 <- getBM(c("refsnp_id","allele","chrom_start","
chrom_strand"), filters = c("chr_name","chrom_start","
chrom_end"), values = list(8,148350,148612), mart =snp.
mart)


snp.q2
```

Retrieving Sequences:

# can get complicated with getBM. Use the getSequence wrapper
# Genome Sequences always 5'-3' but...
# Web-Services mode (default): Strand is context dependant
# MySQL mode: Always top strand

#eg...

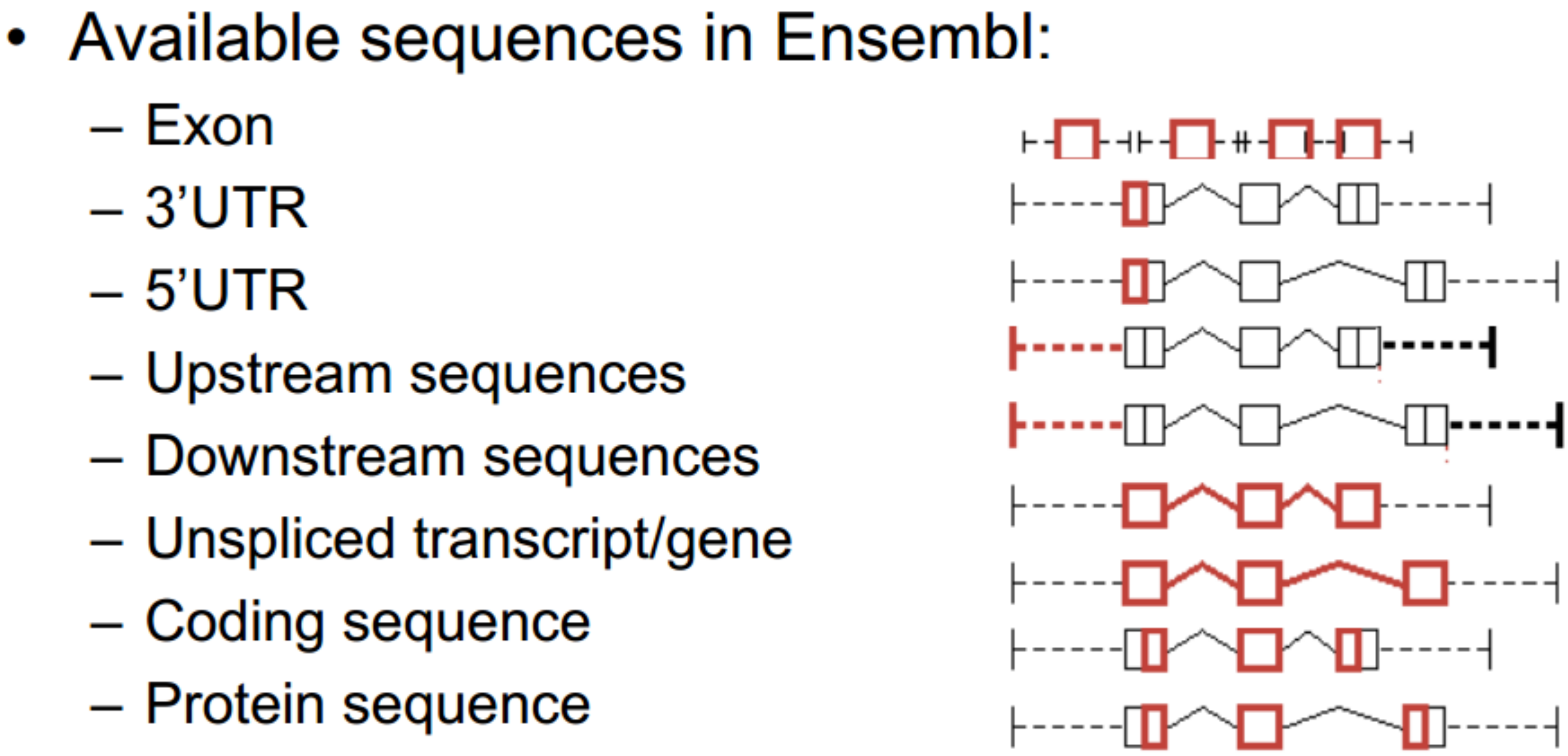
# BRCA1 peptide sequence from gene symbol

```
getSequence(id="BRCA1", type="mgi_symbol", seqType="peptide", mart =
snp.mart)
```

# REST transcript 20 bases upstream

```
getSequence(id='ENSMUST00000113448', type='ensembl_transcript_id',
seqType='transcript_flank', upstream=20, mart=snp.mart)
```

seqTypes:



- Available sequences in Ensembl:
  - Exon
  - 3’UTR
  - 5’UTR
  - Upstream sequences
  - Downstream sequences
  - Unspliced transcript/gene
  - Coding sequence
  - Protein sequence

Note that any of the _flank types need an 'upstream' *or* 'downstream' argument to determine the size of the flanking region. At the moment, you can't specify both.

Exporting Sequences to FASTA files:

```
# The exportFASTA function provides a quick way of saving
# sequences in FASTA format:

res <- getSequence(id="BRCA1", type="mgi_symbol", seqType="peptide", mart = mart)

exportFASTA(res, file='sequence.fa')
```

Linking Datasets...

```
# Make mart connections for each of the datasets:
mouse.mart<-useMart('ensembl', dataset="mmusculus_gene_ensembl")
people.mart<-useMart('ensembl', dataset='hsapiens_gene_ensembl')

# In Ensembl, datasets are made of transcripts
# from a single species.
# Linking datasets amounts to homology

#eg. Get pos of mouse homolog to human 'TP53' gene

getLDS(attributes = c("hgnc_symbol","chromosome_name", "start_position"),
filters = "hgnc_symbol",
values = "TP53",
mart = people.mart,
attributesL = c("chromosome_name","start_position"),
martL = mouse.mart)
}
```

```
V1 V2 V3 V4 V5
1 TP53 17 7512445 11 69393861
```

Tutorial #2 - BiomaRt

Worksheet*:
https://docs.google.com/document/d/1-NOpe6kGMTRWJvGTm6yOoHmciLgIZMk21K-m0bV_Www/pub

Answers are available, but please have a go first:
https://docs.google.com/document/d/1w2HIJ5BAeW3P4bAD_V7MDSE6Qppji7ub01MOsH3stBw/pub

Optional Further Tutorial -- Annotation packages

Work through the bioconductor Annotation Workflow. This will give examples of all annotation package types discussed here.

http://www.bioconductor.org/help/workflows/annotation/annotation/ **

Aim to attempt the first part: "Sample ChipDb Workflow", but if you finish this early, try the further exercises:

You may also want to take a look at the link on slide 16:

http://www.bioconductor.org/help/course-materials/2011/BioC2011/LabStuff/AnnotationSlidesBioc2011.pdf

*** Please note there are a couple of things that need correcting in the workflow -- I have listed them in the next slide*

Notes on Extended Tutorial -- there are one or two errors:

Some errors have crept into the Annotation Workflow, probably due to changes in the underlying packages not updated:

1) I didn't get the **hgu95av2.db** package installed**,** you will need to install the "hgu95av2.db" annotation package

2) There is an error in "Sample ChipDb Workflow"

columns(hgu95av2.db) #won't work

use instead

cols(hgu95av2.db)

3) similarly an error here,

- select(hgu95av2.db, keys = ids, **columns** = c("ENTREZID", "GENENAME", "SYMBOL"), keytype = "PROBEID")
- res <- select(hgu95av2.db, keys = ids[1], **columns** = "GO", keytype = "PROBEID")
- head(select(GO.db, keys = res$GO, **columns** = "TERM", keytype = "GOID"))

replace **columns** with **cols**

4) you will also need to install the **GO.db** package