

Practical: Annotation

Martin Morgan (mtmorgan@fhcrc.org)

27-28 February, 2014

Contents

1	Gene annotation	1
1.1	Data packages	1
1.2	Internet resources	3
2	Genome annotation	4
2.1	Transcript annotation packages	4
2.2	rtracklayer	4

1 Gene annotation

1.1 Data packages

Organism-level ('org') packages contain mappings between a central identifier (e.g., Entrez gene ids) and other identifiers (e.g. GenBank or Uniprot accession number, RefSeq id, etc.). The name of an org package is always of the form `org.<Sp>.<id>.db` (e.g. `org.Sc.sgd.db`) where `<Sp>` is a 2-letter abbreviation of the organism (e.g. `Sc` for *Saccharomyces cerevisiae*) and `<id>` is an abbreviation (in lower-case) describing the type of central identifier (e.g. `sgd` for gene identifiers assigned by the *Saccharomyces* Genome Database, or `eg` for Entrez gene ids). The "How to use the '.db' annotation packages" vignette in the `AnnotationDbi` package (org packages are only one type of ".db" annotation packages) is a key reference. The '.db' and most other *Bioconductor* annotation packages are updated every 6 months.

Annotation packages usually contain an object named after the package itself. These objects are collectively called `AnnotationDb` objects, with more specific classes named `OrgDb`, `ChipDb` or `TranscriptDb` objects. Methods that can be applied to these objects include `cols`, `keys`, `keytypes` and `select`. Common operations for retrieving annotations are summarized in Table~1.

Exercise 1

This exercise illustrates basic use of the 'select' interface to annotation packages.

- What is the name of the org package for Homo sapiens? Load it. Display the `OrgDb` object for the `org.Hs.eg.db` package. Use the `columns` method to discover which sorts of annotations can be extracted from it.
- Use the `keys` method to extract ENSEMBL identifiers and then pass those keys in to the `select` method in such a way that you extract the SYMBOL (gene symbol) and GENENAME information for each. Use the following ENSEMBL ids.

```
ensids <- c("ENSG00000130720", "ENSG00000103257", "ENSG00000156414",  
           "ENSG00000144644", "ENSG00000159307", "ENSG00000144485")
```

Table 1: Common operations for retrieving and manipulating annotations.

Category	Function	Description
Discover	columns	List the kinds of columns that can be returned
	keytypes	List columns that can be used as keys
	keys	List values that can be expected for a given keytype
	select	Retrieve annotations matching keys, keytype and columns
Manipulate	setdiff, union, intersect	Operations on sets
	duplicated, unique	Mark or remove duplicates
	%in%, match	Find matches
	any, all	Are any TRUE? Are all?
<i>GRanges*</i>	merge	Combine two different <code>data.frames</code> based on shared keys
	transcripts, exons, cds	Features (transcripts, exons, coding sequence) as <i>GRanges</i> .
	transcriptsBy, exonsBy, cdsBy	Features group by gene, transcript, etc., as <i>GRangesList</i> .

Solution: The *OrgDb* object is named `org.Hs.eg.db`.

```
library(org.Hs.eg.db)
keytypes(org.Hs.eg.db)

## [1] "ENTREZID"      "PFAM"          "IPI"           "PROSITE"       "ACCNUM"
## [6] "ALIAS"         "CHR"           "CHRLOC"        "CHRLOCEND"     "ENZYME"
## [11] "MAP"           "PATH"          "PMID"          "REFSEQ"        "SYMBOL"
## [16] "UNIGENE"       "ENSEMBL"       "ENSEMBLPROT"   "ENSEMBLTRANS"  "GENENAME"
## [21] "UNIPROT"       "GO"            "EVIDENCE"      "ONTOLOGY"      "GOALL"
## [26] "EVIDENCEALL"   "ONTOLOGYALL"   "OMIM"          "UCSCKG"

columns(org.Hs.eg.db)

## [1] "ENTREZID"      "PFAM"          "IPI"           "PROSITE"       "ACCNUM"
## [6] "ALIAS"         "CHR"           "CHRLOC"        "CHRLOCEND"     "ENZYME"
## [11] "MAP"           "PATH"          "PMID"          "REFSEQ"        "SYMBOL"
## [16] "UNIGENE"       "ENSEMBL"       "ENSEMBLPROT"   "ENSEMBLTRANS"  "GENENAME"
## [21] "UNIPROT"       "GO"            "EVIDENCE"      "ONTOLOGY"      "GOALL"
## [26] "EVIDENCEALL"   "ONTOLOGYALL"   "OMIM"          "UCSCKG"

cols <- c("SYMBOL", "GENENAME")
select(org.Hs.eg.db, keys=ensids, columns=cols, keytype="ENSEMBL")

##           ENSEMBL SYMBOL
## 1 ENSG00000130720 FIBCD1
## 2 ENSG00000103257 SLC7A5
## 3 ENSG00000156414 TDRD9
## 4 ENSG00000144644 GADL1
## 5 ENSG00000159307 SCUBE1
## 6 ENSG00000144485  HES6
##
##                                     GENENAME
## 1                                 fibrinogen C domain containing 1
## 2 solute carrier family 7 (amino acid transporter light chain, L system), member 5
## 3                                     tudor domain containing 9
## 4                                 glutamate decarboxylase-like 1
## 5                                 signal peptide, CUB domain, EGF-like 1
## 6                                 hairy and enhancer of split 6 (Drosophila)
```

Table 2: Selected packages querying web-based annotation services.

Package	Description
<i>AnnotationHub</i>	Ensembl, Encode, dbSNP, UCSC data objects
<i>biomaRt</i>	http://biomart.org , Ensembl and other annotations
<i>PSICQUIC</i>	https://code.google.com/p/psicquic.org , protein interactions
<i>uniprot.ws</i>	http://uniprot.org , protein annotations
<i>KEGGREST</i>	http://www.genome.jp/kegg , KEGG pathways
<i>SRadb</i>	http://www.ncbi.nlm.nih.gov/sra , sequencing experiments.
<i>rtracklayer</i>	http://genome.ucsc.edu , genome tracks.
<i>GEOquery</i>	http://www.ncbi.nlm.nih.gov/geo/ , array and other data
<i>ArrayExpress</i>	http://www.ebi.ac.uk/arrayexpress/ , array and other data

1.2 Internet resources

A short summary of select *Bioconductor* packages enabling web-based queries is in Table~2.

Using biomaRt The *biomaRt* package offers access to the online [biomart](http://biomart.org) resource. This consists of several data base resources, referred to as ‘marts’. Each mart allows access to multiple data sets; the *biomaRt* package provides methods for mart and data set discovery, and a standard method `getBM` to retrieve data.

Exercise 2

warning: This exercise requires INTERNET ACCESS

- Load the *biomaRt* package and list the available marts. Choose the *ensembl* mart and list the datasets for that mart. Set up a mart to use the *ensembl* mart and the *hsapiens_gene_ensembl* dataset.
- A *biomaRt* dataset can be accessed via `getBM`. In addition to the mart to be accessed, this function takes filters and attributes as arguments. Use `filterOptions` and `listAttributes` to discover values for these arguments. Call `getBM` using filters and attributes of your choosing.

Solution:

```
## NEEDS INTERNET ACCESS !!
library(biomaRt)
head(listMarts(), 3) ## list the marts
head(listDatasets(useMart("ensembl")), 3) ## mart datasets
ensembl <- ## fully specified mart
  useMart("ensembl", dataset = "hsapiens_gene_ensembl")

head(listFilters(ensembl), 3) ## filters
myFilter <- "chromosome_name"
head(filterOptions(myFilter, ensembl), 3) ## return values
myValues <- c("21", "22")
head(listAttributes(ensembl), 3) ## attributes
myAttributes <- c("ensembl_gene_id", "chromosome_name")

## assemble and query the mart
res <- getBM(attributes = myAttributes, filters = myFilter,
             values = myValues, mart = ensembl)
```

Use `head(res)` to see the results.

Exercise 3

As an optional exercise, annotate the genes that are differentially expressed in the DESeq2 laboratory, e.g., find the `GENENAME` associated with the five most differentially expressed genes. Do these make biological sense? Can you merge the annotation results with the 'top table' results to provide a statistically and biologically informative summary?

2 Genome annotation

There are a diversity of packages and classes available for representing large genomes. Several include:

`TxDb.*` For transcript and other genome / coordinate annotation.

`BSgenome` For whole-genome representation. See `available.packages` for pre-packaged genomes, and the vignette 'How to forge a BSgenome data package' in the

`Homo.sapiens` For integrating `TxDb.*` and `org.*` packages.

`SNPlocs.*` For model organism SNP locations derived from dbSNP.

`FaFile` (**`Rsamtools`**) for accessing indexed FASTA files.

`SIFT.*`, `PolyPhen`, `ensemblVEP` Variant effect scores.

2.1 Transcript annotation packages

Genome-centric packages are very useful for annotations involving genomic coordinates. It is straight-forward, for instance, to discover the coordinates of coding sequences in regions of interest, and from these retrieve corresponding DNA or protein coding sequences. Other examples of the types of operations that are easy to perform with genome-centric annotations include defining regions of interest for counting aligned reads in RNA-seq experiments and retrieving DNA sequences underlying regions of interest in ChIP-seq analysis, e.g., for motif characterization.

2.2 `rtracklayer`

The `rtracklayer` package allows us to query the UCSC genome browser, as well as providing `import` and `export` functions for common annotation file formats like GFF, GTF, and BED.

Exercise 4

warning: This exercise requires INTERNET ACCESS

Here we use `rtracklayer` to retrieve estrogen receptor binding sites identified across cell lines in the ENCODE project. We focus on binding sites in the vicinity of a particularly interesting region of interest.

- Define our region of interest by creating a `GRanges` instance with appropriate genomic coordinates. Our region corresponds to 10Mb up- and down-stream of a particular gene.
- Create a session for the UCSC genome browser
- Query the UCSC genome browser for ENCODE estrogen receptor $ER\alpha$ transcription marks; identifying the appropriate track, table, and transcription factor requires biological knowledge and detective work.
- Visualize the location of the binding sites and their scores; annotate the mid-point of the region of interest.

Solution: Define the region of interest

```
roi <- GRanges("chr10", IRanges(92106877, 112106876, names="ENSG00000099194"))
```

Create a session

```
library(rtracklayer)
session <- browserSession()
```

Query the UCSC for a particular track, table, and transcription factor, in our region of interest

```
trackName <- "wgEncodeRegTfbsClusteredV2"
tableName <- "wgEncodeRegTfbsClusteredV2"
trFactor <- "ERalpha_a"
ucscTable <- getTable(ucscTableQuery(session, track=trackName,
  range=roi, table=tableName, name=trFactor))
```

Visualize the result

```
plot(score ~ chromStart, ucscTable, pch="+")
abline(v=start(roi) + (end(roi) - start(roi) + 1) / 2, col="blue")
```

