

Extending ensembl¹db: MySQL backend and protein annotations

Johannes Rainer (EURAC research, Italy), Sebastian Gibb, Laurent Gatto (CPU Cambridge, UK)

December 7, 2016

Introduction

- ensemblDb: retrieve gene & transcript annotations.
- ensemblDb package defines the EnsDb class:
 - Wrapper to access annotations from an SQLite database.
 - Same functionality than the GenomicFeatures package (TxDb object).
 - Filter framework to enable specific and fast queries.
 - **NEW**: *MySQL* backend support.
 - **NEW**: provide protein annotations.

Basic usage

- Available methods to extract data: genes, transcripts, transcriptsBy, exons, exonsBy, cdsBy, fiveUTRsByTranscripts, threeUTRsByTranscripts, proteins (**NEW**).
- Example: get all lincRNA genes encoded on chromosome Y.

```
## Load an EnsDb package matching Ensembl version 86
library(EnsDb.Hsapiens.v86)
edb <- EnsDb.Hsapiens.v86

## Retrieve all lincRNAs encoded on chromosome Y.
## Create the filter objects
sf <- SeqnameFilter("Y")
gbf <- GenebiotypeFilter("lincRNA")
```

Basic usage

- Example: (continued)

```
## Retrieve the data.
```

```
genes(edb, filter = list(sf, gbf))
```

GRanges object with 52 ranges and 6 metadata columns:

seqnames	ranges	strand	gene_id
<Rle>	<IRanges>	<Rle>	<character>
ENSG00000278847	Y [2934406, 2934771]	-	ENSG00000278847
ENSG00000231535	Y [3002912, 3102272]	+	ENSG00000231535
ENSG00000229308	Y [4036497, 4100320]	+	ENSG00000229308
...
ENSG00000228786	Y [25378300, 25394719]	-	ENSG00000228786
ENSG00000240450	Y [25482908, 25486705]	+	ENSG00000240450
ENSG00000231141	Y [25728490, 25733388]	+	ENSG00000231141

gene_name	entrezid	gene_biotype	seq_coord_system	
<character>	<character>	<character>	<character>	
ENSG00000278847	RP11-414C23.1	lincRNA	chromosome	
ENSG00000231535	LINC00278	100873962	lincRNA	chromosome
ENSG00000229308	AC010084.1	lincRNA	chromosome	
...	
ENSG00000228786	LINC00266-4P	lincRNA	chromosome	
ENSG00000240450	CSPG4P1Y	114758	lincRNA	chromosome
ENSG00000231141	TTY3	474148;114760	lincRNA	chromosome

symbol	
<character>	
ENSG00000278847	RP11-414C23.1
ENSG00000231535	LINC00278
ENSG00000229308	AC010084.1

Available filters

- For genes: GeneidFilter, GenenameFilter, EntrezidFilter, GenebiotypeFilter, (SymbolFilter).
- For transcripts: TxidFilter, TxbiotypeFilter.
- For exons: ExonidFilter, ExonrankFilter.
- **NEW**: for proteins: ProteinidFilter, UniprotidFilter, UniprotddbFilter, UniprotmappingtypeFilter, ProtodomidFilter.
- Based on chromosomal coordinates: SeqnameFilter, SeqstrandFilter, SeqstartFilter, SeqendFilter, GRangesFilter: condition can be *within* or *overlapping*.
- Multiple filters are combined with a logical *AND*.
- Each filter supports 1:n values, =, != and also a *like* condition.
- Filters speed up queries.

Building annotation databases

- **Option A)**: from GTF/GFF files or AnnotationHub.
- Example: create an EnsDb using AnnotationHub.

```
library(AnnotationHub)
ah <- AnnotationHub()
## Query for available Ensembl gtf files for release 83.
query(ah, pattern=c("ensembl", "release-83", "gtf"))

## Select one; in this case: Anolis carolinensis (lizard)
edbSql83 <- ensDbFromAH(ah=ah["AH50353"])
## BUT: DB lacks NCBI Entrezgene IDs and protein annotation.

## Load the database.
db <- EnsDb(edbSql83)

## Optional, make a package.
makeEnsemblDbPackage(ensdb=edbSql83, version="1.0.0", author="J Rainer",
                     maintainer="Johannes Rainer <johannes.rainer@eurac.edu>")
```

- **Option B)** (preferred): using the Ensembl Perl API:
 - fetchTablesFromEnsembl and makeEnsemblSQLiteFromTables.
 - Fetches also **protein annotations**.
 - Requirements: Perl, Bioperl, Ensembl Perl API.

MySQL backend

- Example: `listEnsDb` list all available databases, `useMySQL` to insert an `EnsDb` to a MySQL server.

```
library(RMySQL)
dbc <- dbConnect(MySQL(), host = "localhost", user = "anonuser", pass = "")

## list all available EnsDb databases.
listEnsDbs(dbc)
```

	dbname	organism	ensembl_version
1	ensdb_acarolinensis_v83	acarolinensis	83
2	ensdb_bvulgaris_v86	bvulgaris	86
3	ensdb_dmelanogaster_v86	dmelanogaster	86
4	ensdb_hsapiens_v75	hsapiens	75

```
## Connect to a database.
dbc <- dbConnect(MySQL(), host = "localhost", user = "anonuser", pass = "",
                  dbname = "ensdb_dmelanogaster_v86")
edb_2 <- EnsDb(dbc)

## To insert an EnsDb to a MySQL: useMySQL
db_my <- useMySQL(edb, host = "localhost", user = "anonuser", pass = "")
```

- Enables a central, MySQL-based annotation server.

Protein data: fetch protein data from an EnsDb

- Example: add protein columns to the columns parameter.

```
## Get all genes with a C2H2 Zinc finger domain and
## return all of their Uniprot IDs
pfam <- ProtDomidFilter("PF13912")
genes(edb, filter = pfam, return.type = "DataFrame",
      columns = c("gene_name", "uniprot_id"))
```

DataFrame with 583 rows and 4 columns

	gene_name	uniprot_id	gene_id	protein_domain_id
	<character>	<character>	<character>	<character>
1	AC002310.11	B7Z5R0	ENSG00000261459	PF13912
2	AC092835.2	A0A087WUV0	ENSG00000233757	PF13912
3	CTD-2006C1.13	F5H0A9	ENSG00000267179	PF13912
...
581	ZSCAN9	O15535	ENSG00000137185	PF13912
582	ZSCAN9	A0A024RCK9	ENSG00000137185	PF13912
583	ZSCAN9	E9PQL7	ENSG00000137185	PF13912

Protein data: fetch protein data from an EnsDb

- Example: use `proteins` method to specifically fetch protein data.

```
## Return the protein annotation as a AAStringSet:
prts <- proteins(edb, filter = GenenameFilter("ZBTB16"),
                 columns = c("tx_id", "tx_biotype"),
                 return.type = "AAStringSet")

prts
```

```
A AAStringSet instance of length 5
width seq                                     names
[1] 673 MDLTKMGMIQLQNPSHTGLLCK...GHKPEEIPPDWRIEKTYLYLCYV ENSP00000338157
[2] 115 MDLTKMGMIQLQNPSHTGLLCK...QAKAEDLDDLLYAAEILEIEYLE ENSP00000437716
[3] 148 MDLTKMGMIQLQNPSHTGLLCK...QASDDNDTEATMADGGAEEDR ENSP00000443013
[4] 673 MDLTKMGMIQLQNPSHTGLLCK...GHKPEEIPPDWRIEKTYLYLCYV ENSP00000376721
[5] 55 XGGLLPQGF IQRELF SKLGELAV...GEQCSVCVELPDNEAVEQHRVF ENSP00000445047
```

```
## Additional columns are available as mcols:
mcols(prts)
```

DataFrame with 5 rows and 4 columns

	tx_id <character>	tx_biotype <character>	protein_id <character>	gene_name <character>
1	ENST00000335953	protein_coding	ENSP00000338157	ZBTB16
2	ENST00000544220	protein_coding	ENSP00000437716	ZBTB16
3	ENST00000535700	protein_coding	ENSP00000443013	ZBTB16
4	ENST00000392996	protein_coding	ENSP00000376721	ZBTB16
5	ENST00000539918	nonsense_mediated_decay	ENSP00000445047	ZBTB16

Protein data: use ensemblDb with Pbase

- Pbase: (Laurent Gatto and Sebastian Gibb): provides classes and functions for the analysis of protein sequence data in proteomics experiments.
- The Proteins object: container for proteins and peptide ranges within the AA sequences.
- Example: fetch a Proteins object for all ZBTB16 proteins including their protein domains.

```
## load Pbase - we need the "ensemldb" branch.  
library(Pbase)  
  
## Fetch proteins including protein domains for ZBTB16  
prts <- Proteins(edb, filter = GenenameFilter("ZBTB16"))  
  
## Amino acid sequence:  
aa(prts)
```

```
A AAStringSet instance of length 5  
width seq names  
[1] 673 MDLTKMGMIQLQNPSHTGLLCK...GHKPEEIPPDWRIEKTYLYLCYV ENSP00000338157  
[2] 115 MDLTKMGMIQLQNPSHTGLLCK...QAKAEDLDDLLYAAEILEIEYLE ENSP00000437716  
[3] 148 MDLTKMGMIQLQNPSHTGLLCK...QASDDNDTEATMADGGAEEEEEDR ENSP00000443013  
[4] 673 MDLTKMGMIQLQNPSHTGLLCK...GHKPEEIPPDWRIEKTYLYLCYV ENSP00000376721  
[5] 55 XGGLLPQGFIIQRELF SKLGELAV...GEQCSVCGVELPDNEAVEQHRVF ENSP00000445047
```

Protein data: use ensemblDb with Pbase

- Example: fetch a Proteins object for all ZBTB16 proteins including their protein domains (continued).

```
## Peptide features:  
pranges(prts)
```

```
IRangesList of length 5
```

```
$ENSP00000338157
```

```
IRanges object with 36 ranges and 3 metadata columns:
```

start	end	width	protein_id	protein_domain_source
<integer>	<integer>	<integer>	<character>	<character>
PS50157	602	629	28 ENSP00000338157	pfscan
PS50157	490	517	28 ENSP00000338157	pfscan
PS50157	630	657	28 ENSP00000338157	pfscan
...
SM00355	432	454	23 ENSP00000338157	smart
SM00355	574	596	23 ENSP00000338157	smart
SM00225	34	126	93 ENSP00000338157	smart

```
interpro_accession
```

```
<character>
```

PS50157	IPR007087
PS50157	IPR007087
PS50157	IPR007087
...	...
SM00355	IPR015880
SM00355	IPR015880
SM00225	IPR000210

```
...  
<4 more elements>
```

Protein data: use ensemblDb with Pbase

- Example: use ensemblDb to map peptide features to the genome.

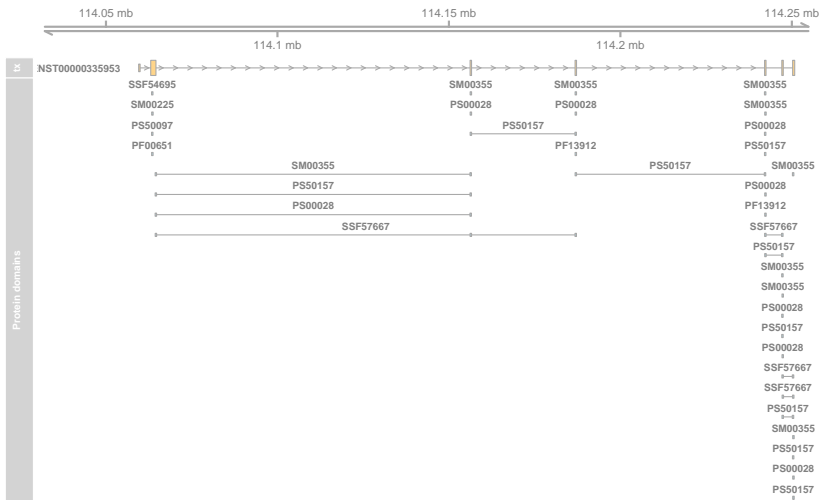
```
## Map all protein domains to the genome
gen_map <- mapToGenome(prts, edb)

## Plot the results for the first protein (transcript)
txid <- gen_map[[1]]$tx_id
## Get the gene region track for the first transcript
tx <- getGeneRegionTrackForGviz(edb, filter = TxidFilter(txid))

## Add a protein ID column
map_1 <- gen_map[[1]]
map_1$id <- names(map_1)

## Plot using Gviz
library(Gviz)
plotTracks(list(GenomeAxisTrack(),
               GeneRegionTrack(tx, name = "tx"),
               AnnotationTrack(map_1, groupAnnotation = "id",
                              just.group = "above",
                              name = "Protein domains")),
           transcriptAnnotation = "transcript")
```

Protein data: use ensemblldb with Phage



Things not covered

- ensemblDb provides full AnnotationDbi support.
- Example: use AnnotationDbi's select method to fetch annotations: keys can be a character list of IDs or a list of filter objects.

```
## Get all data for the gene SKA2
Res <- select(edb, keys="SKA2", keytype="GENENAME")
head(Res, n=3)
```

	ENTREZID	EXONID	EXONIDX	EXONSEQEND	EXONSEQSTART	GENEBIOTYPE
1	348235	ENSE00001324111	1	59155269	59155131	protein_coding
2	348235	ENSE00003636954	2	59131367	59131281	protein_coding
3	348235	ENSE00003478713	3	59119495	59119319	protein_coding

	GENEID	GENENAME	GENESEQEND	GENESEQSTART	INTERPROACCESSION	ISCIRCULAR
1	ENSG00000182628	SKA2	59155269	59109951	IPR026762	0
2	ENSG00000182628	SKA2	59155269	59109951	IPR026762	0
3	ENSG00000182628	SKA2	59155269	59109951	IPR026762	0

	PROTDOMEND	PROTDOMSTART	PROTEINDOMAINID	PROTEINDOMAINSOURCE	PROTEINID
1	115	2	PF16740	pfam ENSP00000333433	
2	115	2	PF16740	pfam ENSP00000333433	
3	115	2	PF16740	pfam ENSP00000333433	

	PROTEINSEQUENCE
1	MEAEVDKLELMFQKAESLDYIQYRLEYEIKTNHPDASEKNPVTLLKELSVIKSRYQTLYARFKPVAVEQKESKSRICATVKKTMNM
2	MEAEVDKLELMFQKAESLDYIQYRLEYEIKTNHPDASEKNPVTLLKELSVIKSRYQTLYARFKPVAVEQKESKSRICATVKKTMNM
3	MEAEVDKLELMFQKAESLDYIQYRLEYEIKTNHPDASEKNPVTLLKELSVIKSRYQTLYARFKPVAVEQKESKSRICATVKKTMNM

	SEQCOORDSYSTEM	SEQLENGTH	SEQNAME	SEQSTRAND	SYMBOL	TXBIOTYPE	TXCDSSEQEND
1	chromosome	83257441	17	-1	SKA2	protein_coding	59155163
2	chromosome	83257441	17	-1	SKA2	protein_coding	59155163
3	chromosome	83257441	17	-1	SKA2	protein_coding	59155163

Things not covered

- Easy integration of UCSC and Ensembl annotations: use `seqlevelsStyle` to change chromosome naming scheme in `EnsDb`.
- Example: How to integrate Ensembl based annotation with UCSC data?

```
## Get chromosome names, they are "Ensembl-formatted"  
head(seqlevels(edb))
```

```
[1] "1" "10" "11" "12" "13" "14"
```

```
## Get genes on chromosome Y, UCSC style.  
genes(edb, filter=SeqnameFilter("chrY"))
```

GRanges object with 0 ranges and 6 metadata columns:

seqnames	ranges	strand	gene_id	gene_name	entrezid	gene_biotype
<Rle>	<IRanges>	<Rle>	<character>	<character>	<character>	<character>
seq_coord_system	symbol					
<character>	<character>					

seqinfo: no sequences

```
## Solution: change the chromosome naming style:  
seqlevelsStyle(edb) <- "UCSC"  
## Get chromosome names  
head(seqlevels(edb))
```

```
[1] "chr1" "chr10" "chr11" "chr12" "chr13" "chr14"
```

Warning message:

Finally

Thank you for your attention!

<https://github.com/jotsetung/EuroBioC2016-ensemldb.git>